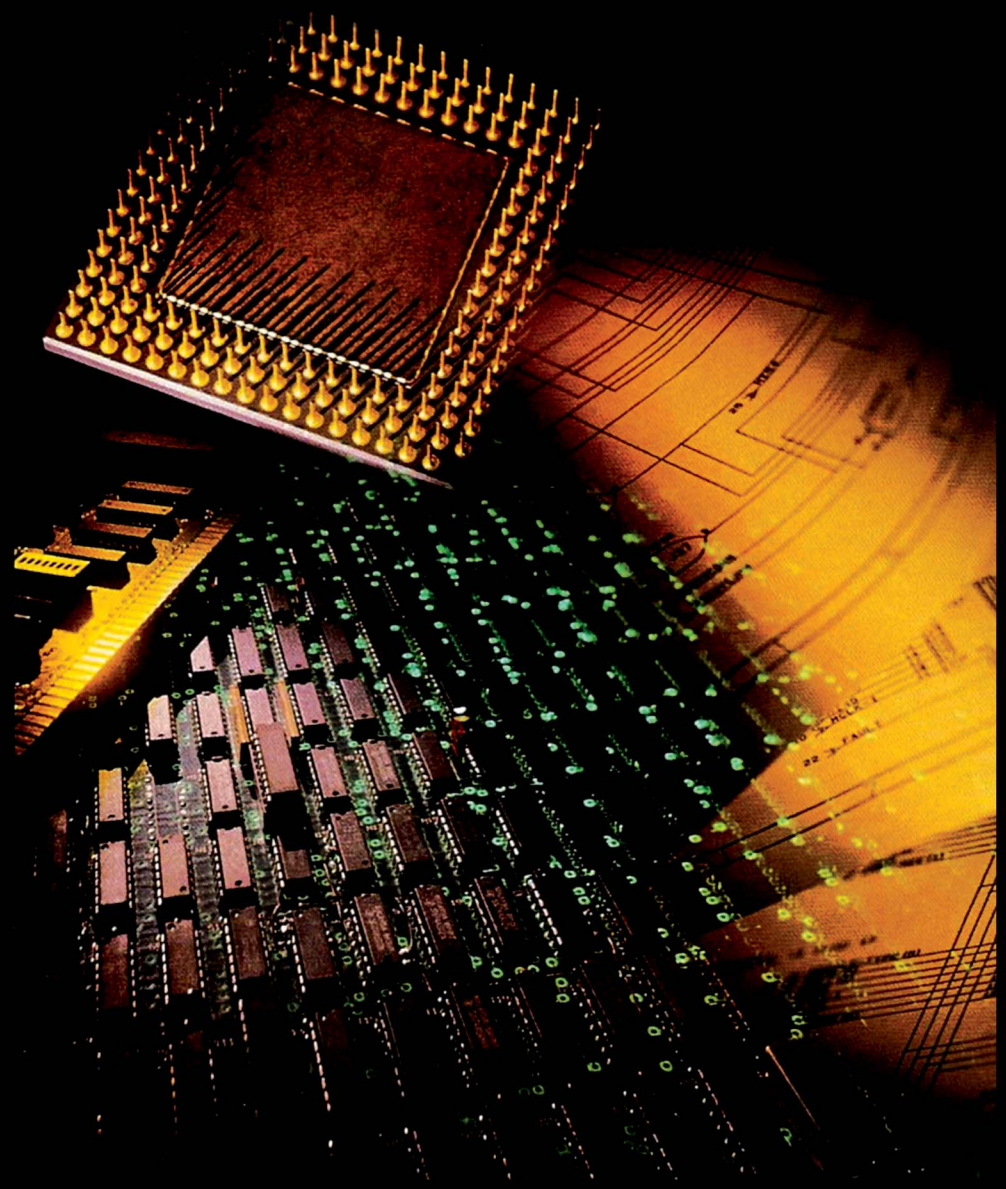
CD com arquivos MultiSIM (inglês)

SISTEMAS DIGITAIS

FUNDAMENTOS E APLICAÇÕES

9ª edição



FLOYD

Bookman®

SISTEMAS DIGITAIS

Sobre o Autor

Tom Floyd formou-se em engenharia elétrica na University of Florida e fez mestrado em engenharia elétrica na Southern Methodist University. Foi membro do Mayland Community College em Spruce Pine, Carolina do Norte, e foi professor no Electronics Technology Program por cinco anos. Em 1977 lançou a primeira edição de Sistemas Digitais e desde 1983 dedica-se exclusivamente à atividade de escrever livros.

F645s Floyd, Thomas L.
Sistemas digitais [recurso eletrônico] : fundamentos e aplicações / Thomas L. Floyd ; tradução José Lucimar do Nascimento. – Dados eletrônicos. – 9. ed. – Porto Alegre : Bookman, 2007.

Editado também como livro impresso em 2007.
ISBN 978-85-7780-107-7

1. Computador – Estrutura. 2. Eletrônica digital. I. Título.

CDU 004

Catalogação na publicação: Juliana Lagôas Coelho – CRB 10/1798

THOMAS L. FLOYD

SISTEMAS DIGITAIS

FUNDAMENTOS E APLICAÇÕES

9ª edição

Tradução:

José Lucimar do Nascimento
Professor e coordenador do CETEL
Engenheiro de Telecomunicações (PUCMG)
Especialista em Engenharia de Sistemas (UFMG)

Consultoria, supervisão e revisão técnica desta edição:

Antonio Pertence Júnior
Engenheiro eletrônico e de telecomunicações
Especialista em Processamento de Sinais (Ryerson University - Canadá)
Professor de Telecomunicações da FUMEC/MG
Professor titular da Faculdade de Sabará/MG

Versão impressa
desta obra: 2007

Bookman®

2007

Obra originalmente publicada sob o título *Digital Fundamentals, 9th Edition*

ISBN 0131946099

Capa: *Gustavo Demarchi,* arte sobre capa original

Leitura final: *Rachel Garcia Valdez*

Supervisão editorial: *Arysinha Jacques Affonso e Denise Weber Nowaczyk*

Editoração eletrônica: *Laser House*

As fotos de abertura de capítulo e inseridas ao longo do texto são de Getty Images.

Av. Jerônimo de Ornelas, 670 - Santana
90040-340 Porto Alegre RS
Fone (51) 3027-7000 Fax (51) 3027-7070

SÃO PAULO
Av. Angélica, 1.091 - Higienópolis
01227-100 São Paulo SP
Fone (11) 3665-1100 Fax (11) 3667-1333

SAC 0800 703-3444

IMPRESSO NO BRASIL
PRINTED IN BRAZIL

Prefácio

Bem-vindos à nona edição de *Sistemas Digitais: Fundamentos e Aplicações**. Uma abordagem consistente dos fundamentos essenciais da tecnologia digital é vital para qualquer pessoa que procura trilhar uma carreira nesta empolgante área que evolui rapidamente. Este livro foi cuidadosamente organizado para incluir uma abordagem atualizada de tópicos que podem ser abordados em sua totalidade, de forma condensada ou ainda suprimindo alguns, de acordo com a ênfase da disciplina.

A abordagem dos tópicos é clara, direta e bem ilustrada, seguindo o formato bem-sucedido das edições anteriores. Muitos tópicos foram enriquecidos ou ampliados e diversas melhorias podem ser encontradas ao longo deste livro.

Provavelmente o leitor encontrará mais tópicos do que pode ser abordado em apenas uma disciplina. Essa extensão proporciona uma flexibilidade para se adequar a uma variedade de programas. Por exemplo, alguns dos projetos orientados ou tópicos de aplicação de sistemas podem não ser apropriados para determinados cursos. Outros programas podem não abordar a lógica programável e alguns podem não ter carga horária disponível para tratar de, por exemplo, computadores, microprocessadores ou processamento de sinais digitais. Além disso, alguns cursos podem não ter a necessidade de entrar em detalhes internos aos circuitos dos chips. Esses e outros tópicos podem ser omitidos ou abordados de forma sintética sem afetar a abordagem dos tópicos fundamentais. O conhecimento prévio de circuitos transistorizados não é um pré-requisito para o estudo deste livro mesmo que a abordagem de tecnologia de circuitos integrados (circuito interno aos chips) esteja incluída no "capítulo flutuante", o qual é opcional.

No Sumário, consta um código de cores que indica uma variedade de abordagens para se adequar aos requisitos da maioria dos cursos. Este livro é organizado em módulos que permite a inclusão ou omissão de vários tópicos sem prejudicar outros que forem abordados no seu curso. Devido ao contínuo crescimento do uso de lógica programável, foi dedicado um capítulo inteiro (Capítulo 11) a esse tópico, incluindo PALs, GALs, CPLDs e FPGAs; dispositivos específicos dos fabricantes Altera e Xilinx são apresentados. É abordado também, como uma introdução geral, o software usado em lógica programável e a lógica *boundary scan*.

Novidades desta edição

- O código Hamming de detecção e correção de erros
- Somadores com carry antecipado
- Uma introdução sintética de VHDL
- Abordagem expandida e melhorada de instrumentos de teste
- Abordagem expandida e reorganizada de lógica programável
- Abordagem melhorada de análise de defeito
- Nova abordagem de Aplicações em Sistemas Digitais

Características

- Apresentação em duas cores.
- Notas que fornecem informações de forma bastante condensada.
- Termos importantes estão listados no início de cada capítulo, aparecem em negrito ao longo do texto e estão definidos ao final dos capítulos. No final do livro há um glossário com todos esses termos.

---

* N. de T.: O nome original em inglês é *Digital Fundamentals*.

- O Capítulo 14 fala sobre tecnologia de CIs (circuito interno ao chip) e foi projetado como um "capítulo flutuante" porque pode ser estudado em qualquer momento do seu curso.
- Considerações gerais e objetivos em cada início de capítulo.
- Introdução e objetivos no início de cada seção dentro de um capítulo.
- Questões para revisão e exercícios no final de cada seção de um capítulo.
- Um Problema Relacionado em cada exemplo resolvido.
- Notas de computação entremeiam o texto fornecendo informações interessantes sobre a tecnologia de computadores conforme estejam relacionadas com o texto abordado.
- Dicas Práticas entremeiam o texto fornecendo informações práticas e úteis.
- O tópico Aplicações em Sistemas Digitais é uma característica no final de muitos capítulos que fornece aplicações práticas e interessantes dos fundamentos dos circuitos lógicos.
- Resumo do capítulo no final de cada capítulo.
- Autoteste de múltipla escolha no final de cada capítulo.
- Conjunto de problemas, divididos em seções, extensivos no final de cada capítulo incluem problemas básicos, de análise de defeito, aplicações de sistemas e projetos especiais.
- Abordagem do uso e aplicação de instrumentos de teste, incluindo osciloscópio, analisador lógico, gerador de funções e DMM (multímetro digital).
- O Capítulo 12 fornece uma introdução aos computadores digitais.
- O Capítulo 13 introduz o processamento de sinais digitais, incluindo a conversão analógico-digital e digital-analógico.
- Os conceitos de lógica programável começam a ser introduzidos no Capítulo 1.
- Dispositivos na forma de CIs de função fixa são apresentados ao longo do livro.
- O Capítulo 11 fornece uma abordagem de PALs, GALs, CPLDs e FPGAs bem como uma abordagem geral da programação de PLDs.

- Os diagramas no texto, identificados pelo ícone especial ao lado, são implementados no Multisim® 2001 e Multisim® 7, e os arquivos desses circuitos estão no CD-ROM que acompanha este livro. Esses arquivos (disponíveis também no Companion Website www.prenhall.com/floyd) são fornecidos sem custo extra para o consumidor e são para uso de qualquer pessoa que escolha o software Multisim. Esse software é considerado uma excelente ferramenta de simulação para uso em sala ou laboratório. Entretanto, o sucesso no uso deste livro independe do uso desses arquivos de circuito.
- A lógica *boundary scan* associada aos dispositivos programáveis é introduzida no Capítulo 11.
- Além da *boundary scan*, a abordagem de análise de defeito inclui métodos de teste de lógica programável, tal como o tradicional, *bed-of-nails* e *flying probe*. O *boundary scan* e esses outros métodos são importantes na indústria.
- Para aqueles que desejarem incluir a programação ABEL, uma introdução é fornecida no site **www.prenhall.com/floyd**.

Recursos Complementares para o Estudante (em inglês)

- *Experiments in Digital Fundamentals*, um manual de laboratório de autoria de David M. Buchla.*
- Dois CD-ROMs que acompanham esse livro:
  Arquivos de circuitos para uso com o software Multisim
  Folhas de dados (*data sheets*) de dispositivos digitais da Texas Instruments

Recursos para o Professor**

- *Slides em PowerPoint*®. Apresentação com Notas de Aula e figuras do livro.
- *Companion Website* (www.prenhall.com/floyd). Esse site oferece para o professor a possibilidade de colocar o seu programa on-line com o nosso Syllabus Manager™. Essa é uma formidável solução para estudos a distância ou para uso em atividades assistidas por computador.
- Manual do professor. Incluem o desenvolvimento das soluções dos problemas do capítulo, soluções para Aplicações em Sistemas Digitais, um resumo dos resultados de simulações com o Multisim e a solução das questões de laboratório com manual de laboratório de David M, Buchla.
- *Test Item File*. Essa edição de Test Item File tem mais de 900 questões.
- *TestGen*.® Essa é uma versão eletrônica de Test Item File, que possibilita ao professor personalizar os testes para a classe.

Para acessar materiais suplementares on-line, os professores precisam solicitar um código de acesso do professor. Acesse www.prenhall.com, clique no link **Instructor Resource Center** e em seguida clique em **Register Today** para obter um código de acesso do professor. Dentro de 48 horas após o registro você receberá uma confirmação por e-mail incluindo o código de acesso do professor. Uma vez recebido seu código, acesse o website e faça o download do material que você deseja usar.

Características das Ilustrações nos Capítulos

Início de Capítulo Cada capítulo começa com as informações características nas duas primeiras páginas, como mostra a Figura P–1. A página da esquerda inclui uma lista de seções do capítulo e uma lista de objetivos a serem alcançados no estudo do capítulo. Um conteúdo típico da página à direita inclui considerações gerais do capítulo, uma lista de dispositivos específicos apresentados no capítulo (cada novo dispositivo é indicado por um logo de um CI no ponto onde ele é introduzido), uma abordagem prévia resumida de Aplicações em Sistemas Digitais, uma lista de termos importantes e uma referência de um website para ajudar no estudo do capítulo.

Início de seção Cada seção de um capítulo inicia com uma breve introdução que inclui considerações gerais e objetivos da seção. Uma ilustração é mostrada na Figura P–2.

Revisão da seção Cada seção termina com uma revisão que consiste de questões ou exercícios que enfatizam os principais conceitos apresentados na seção. Essa característica é mostrada na Figura P–2. As respostas para as Revisões das Seções são apresentadas no final do capítulo.

---

* Esse material está disponível apenas no mercado norte-americano.

** Professores interessados em receber material complementar (em inglês e em português) devem entrar em contato com a Bookman Editora pelo endereço secretariaeditorial@artmed.com.br e anexar comprovante de docência.

6

FUNÇÕES DE LÓGICA COMBINACIONAL

TÓPICOS DO CAPÍTULO

6-1 Somadores Básicos
6-2 Somadores Binários Paralelos
6-3 Somadores com Carry Ondulante *versus* Somadores com Carry Antecipado
6-4 Comparadores
6-5 Decodificadores
6-6 Codificadores
6-7 Conversores de Códigos
6-8 Multiplexadores (Seletores de Dados)
6-9 Demultiplexadores
6-10 Geradores/Verificadores de Paridade
6-11 Análise de Defeito
Aplicações em Sistemas Digitais

OBJETIVOS DO CAPÍTULO

- Fazer distinção entre meio-somadores e somadores-completos
- Usar somadores-completos para implementar somadores binários em paralelo
- Explicar as diferenças entre somadores em paralelo com carry ondulante e com carry antecipado
- Usar o comparador de magnitude para determinar a relação entre dois números binários e usar comparadores em cascata para conseguir realizar comparações de números com maior número de bits
- Implementar um decodificador binário básico
- Usar decodificadores de BCD para 7 segmentos em sistemas com display
- Usar um codificador de decimal para BCD com prioridade numa aplicação com um teclado simples
- Converter de binário para código Gray e vice-versa usando dispositivos lógicos
- Usar multiplexadores em seleção de dados, displays multiplexados, geração de funções lógicas e sistemas de comunicação simples
- Usar decodificadores como demultiplexadores
- Explicar o significado de paridade
- Usar geradores e verificadores de paridade para detectar erros de bit em sistemas digitais
- Implementar um sistema de comunicação de dados simples
- Identificar glitches, que são problemas comuns em sistemas digitais

TERMOS IMPORTANTES

- Meio-somador
- Somador-completo
- Conexão em cascata
- Carry ondulante
- Carry antecipado
- Decodificador
- Codificador
- Codificador com prioridade
- Multiplexador (MUX)
- Demultiplexador (DEMUX)
- Bit de paridade
- Glitch

INTRODUÇÃO

Neste capítulo, diversos tipos de circuitos lógicos combinacionais são apresentados incluindo somadores, comparadores, decodificadores, codificadores, conversores de código, multiplexadores (seletores de dados), demultiplexadores e geradores/verificadores de paridade. São incluídos também exemplos de circuitos integrados (CIs) de função fixa.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

74XX42	74XX47	74XX85
74XX138	74XX139	74XX147
74XX148	74XX151	74XX154
74XX157	74XX280	74XX283

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico Aplicações em Sistemas Digitais ilustra conceitos abordados neste capítulo e lida com uma parte de um sistema de controle de tráfego de veículos. Esse mesmo tópico nos Capítulos 6, 7 e 8 tem como foco as diversas partes de um sistema de controle de tráfego de veículos. Basicamente, esse sistema controla o trânsito no cruzamento de uma rua movimentada com uma rua secundária de pouco movimento. Esse sistema inclui uma seção lógica combinacional, para a qual os tópicos desse capítulo se aplicam, um circuito de temporização, para o qual se aplica o Capítulo 7 e um circuito lógico seqüencial para o qual se aplica o Capítulo 8.

WWW. ACESSE O SITE
Recursos que o ajudarão no estudo deste capítulo estão disponíveis em
http://www.prenhall.com/floyd

313

▲ **FIGURA P–1**

Início de capítulo.

Exemplos resolvidos e problemas propostos Uma grande quantidade de exemplos resolvidos ajudam a ilustrar e esclarecer os conceitos básicos ou os procedimentos específicos. Cada exemplo termina com um Problema Relacionado que reforça ou amplia o exemplo, propondo ao estudante a busca da solução de um problema similar ao exemplo. Um exemplo resolvido típico juntamente com um Problema Relacionado é mostrado na Figura P–3.

Seção de análise de defeito Muitos capítulos incluem uma seção de análise de defeito relacionada aos tópicos abordados no capítulo que enfatiza a técnica de análise de defeito e o uso de instrumentos de teste. Uma parte de uma seção de análise de defeito é ilustrada na Figura P–4.

Aplicações em sistemas digitais Tópico que aparece no final de diversos capítulos apresentando uma aplicação prática dos conceitos abordados no capítulo. Ele apresenta um sistema do "mundo real" no qual a análise de funcionamento, a análise de defeito e os elementos do projeto são implementados usando procedimentos abordados no capítulo. Alguns tópicos de Aplicações em Sistemas Digitais se limitam a um único capítulo e outros se estendem por dois ou mais capítulos. Os temas específicos de Aplicações em Sistemas Digitais são os seguintes:

- Sistema de controle e contagem de comprimidos: Capítulo 1
- Display digital: Capítulos 4 e 11
- Sistema de controle de um tanque de armazenamento: Capítulo 5

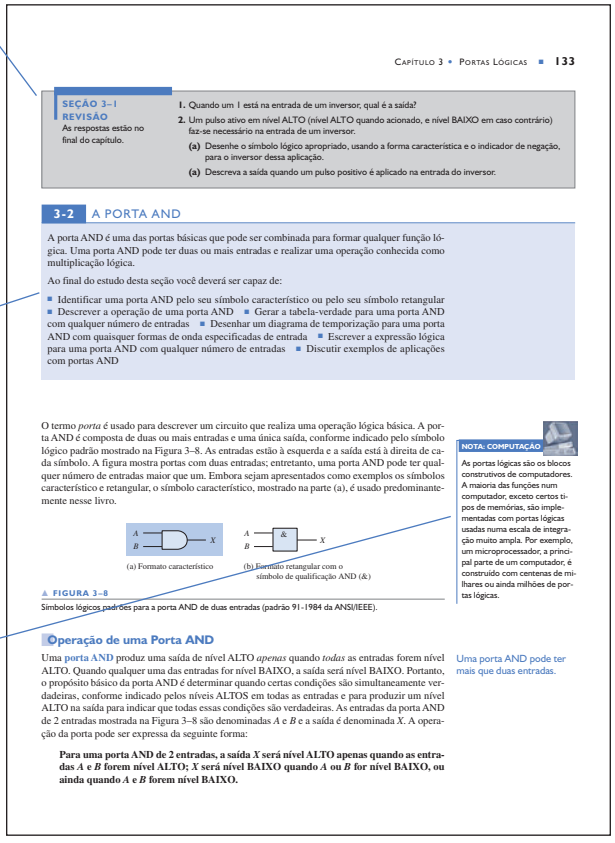
▶ FIGURA P–2

Início de seção e revisão da seção.

Exercício de revisão no final de cada seção.

No início de cada seção existe um parágrafo introdutório e uma lista de objetivos da seção orientados pelo desempenho esperado.

Notas relativas à área de computação são encontradas ao longo do livro.



CAPÍTULO 3 • PORTAS LÓGICAS ■ 133

SEÇÃO 3–1 REVISÃO
As respostas estão no final do capítulo.

1. Quando um 1 está na entrada de um inversor, qual é a saída?
2. Um pulso ativo em nível ALTO (nível ALTO quando acionado, e nível BAIXO em caso contrário) faz-se necessário na entrada de um inversor.
   (a) Desenhe o símbolo lógico apropriado, usando a forma característica e o indicador de negação, para o inversor dessa aplicação.
   (a) Descreva a saída quando um pulso positivo é aplicado na entrada do inversor.

3-2 A PORTA AND

A porta AND é uma das portas básicas que pode ser combinada para formar qualquer função lógica. Uma porta AND pode ter duas ou mais entradas e realizar uma operação conhecida como multiplicação lógica.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma porta AND pelo seu símbolo característico ou pelo seu símbolo retangular ■ Descrever a operação de uma porta AND ■ Gerar a tabela-verdade para uma porta AND com qualquer número de entradas ■ Desenhar um diagrama de temporização para uma porta AND com quaisquer formas de onda especificadas de entrada ■ Escrever a expressão lógica para uma porta AND com qualquer número de entradas ■ Discutir exemplos de aplicações com portas AND

O termo *porta* é usado para descrever um circuito que realiza uma operação lógica básica. A porta AND é composta de duas ou mais entradas e uma única saída, conforme indicado pelo símbolo lógico padrão mostrado na Figura 3–8. As entradas estão à esquerda e a saída está à direita de cada símbolo. A figura mostra portas com duas entradas; entretanto, uma porta AND pode ter qualquer número de entradas maior que um. Embora sejam apresentados como exemplos os símbolos característico e retangular, o símbolo característico, mostrado na parte (a), é usado predominantemente nesse livro.

NOTA: COMPUTAÇÃO
As portas lógicas são os blocos construtivos de computadores. A maioria das funções num computador, exceto certos tipos de memórias, são implementadas com portas lógicas usadas numa escala de integração muito ampla. Por exemplo, um microprocessador, a principal parte de um computador, é construído com centenas de milhares ou ainda milhões de portas lógicas.

(a) Formato característico (b) Formato retangular com o símbolo de qualificação AND (&)

▲ FIGURA 3–8
Símbolos lógicos padrões para a porta AND de duas entradas (padrão 91-1984 da ANSI/IEEE).

Operação de uma Porta AND

Uma **porta AND** produz uma saída de nível ALTO *apenas* quando *todas* as entradas forem nível ALTO. Quando qualquer uma das entradas for nível BAIXO, a saída será nível BAIXO. Portanto, o propósito básico da porta AND é determinar quando certas condições são simultaneamente verdadeiras, conforme indicado pelos níveis ALTOS em todas as entradas e para produzir um nível ALTO na saída para indicar que todas essas condições são verdadeiras. As entradas da porta AND de 2 entradas mostrada na Figura 3–8 são denominadas *A* e *B* e a saída é denominada *X*. A operação da porta pode ser expressa da seguinte forma:

Uma porta AND pode ter mais que duas entradas.

Para uma porta AND de 2 entradas, a saída *X* será nível ALTO apenas quando as entradas *A* e *B* forem nível ALTO; *X* será nível BAIXO quando *A* ou *B* for nível BAIXO, ou ainda quando *A* e *B* forem nível BAIXO.

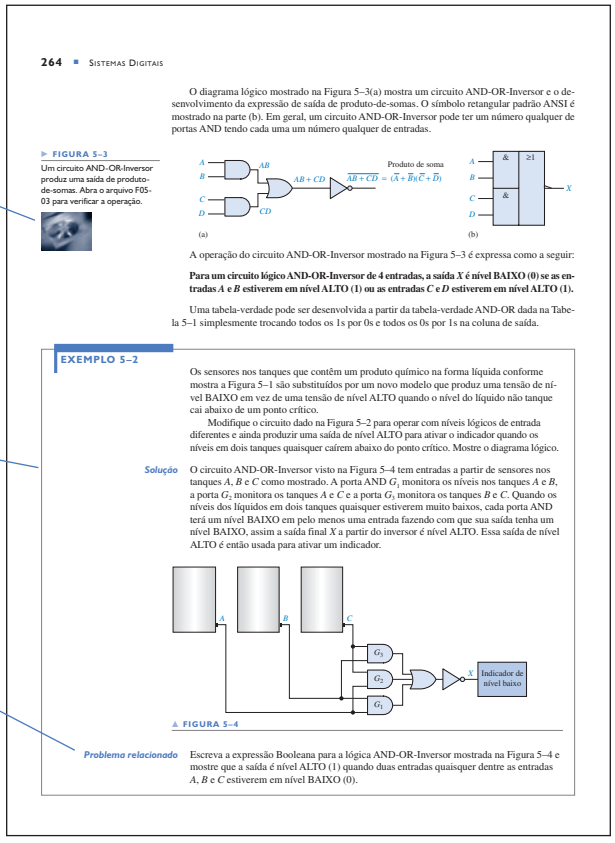
▶ FIGURA P–3

Exemplo e problema relacionado.

Um ícone especial indica os circuitos selecionados que estão no CD-ROM que acompanha este livro.

Os exemplos estão realçados do texto.

Cada exemplo contém um problema relacionado ao exemplo.



264 ■ SISTEMAS DIGITAIS

O diagrama lógico mostrado na Figura 5–3(a) mostra um circuito AND-OR-Inversor e o desenvolvimento da expressão de saída de produto-de-somas. O símbolo retangular padrão ANSI é mostrado na parte (b). Em geral, um circuito AND-OR-Inversor pode ter um número qualquer de portas AND tendo cada uma um número qualquer de entradas.

▶ FIGURA 5–3
Um circuito AND-OR-Inversor produz uma saída de produto-de-somas. Abra o arquivo F05-03 para verificar a operação.

Produto de soma

(a) (b)

A operação do circuito AND-OR-Inversor mostrado na Figura 5–3 é expressa como a seguir:

Para um circuito lógico AND-OR-Inversor de 4 entradas, a saída *X* é nível BAIXO (0) se as entradas *A* e *B* estiverem em nível ALTO (1) ou as entradas *C* e *D* estiverem em nível ALTO (1).

Uma tabela-verdade pode ser desenvolvida a partir da tabela-verdade AND-OR dada na Tabela 5–1 simplesmente trocando todos os 1s por 0s e todos os 0s por 1s na coluna de saída.

EXEMPLO 5–2

Os sensores nos tanques que contêm um produto químico na forma líquida conforme mostra a Figura 5–1 são substituídos por um novo modelo que produz uma tensão de nível BAIXO em vez de uma tensão de nível ALTO quando o nível do líquido não tanque cai abaixo de um ponto crítico.

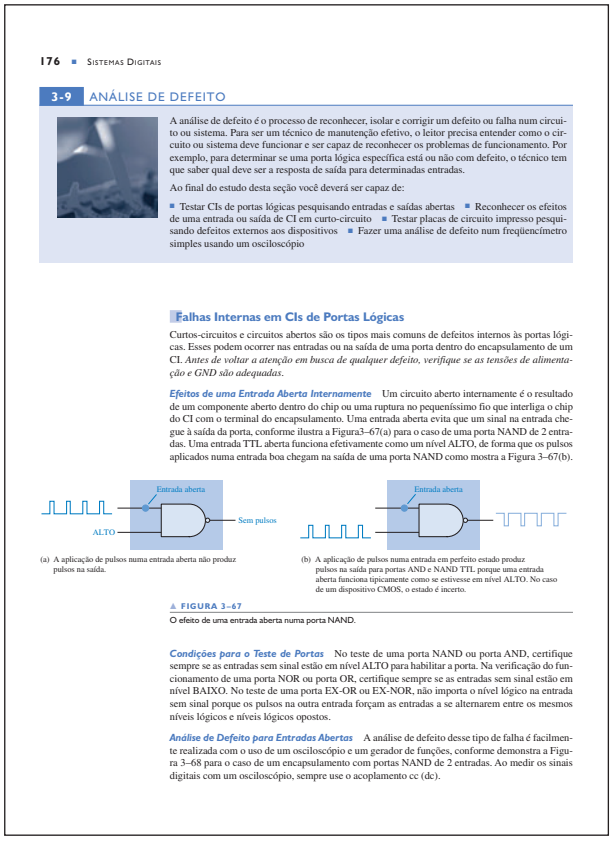
Modifique o circuito dado na Figura 5–2 para operar com níveis lógicos de entrada diferentes e ainda produzir uma saída de nível ALTO para ativar o indicador quando os níveis em dois tanques quaisquer caírem abaixo do ponto crítico. Mostre o diagrama lógico.

Solução O circuito AND-OR-Inversor visto na Figura 5–4 tem entradas a partir de sensores nos tanques *A*, *B* e *C* como mostrado. A porta AND G_3 monitora os níveis nos tanques *A* e *B*, a porta G_2 monitora os tanques *A* e *C* e a porta G_1 monitora os tanques *B* e *C*. Quando os níveis dos líquidos em dois tanques quaisquer estiverem muito baixos, cada porta AND terá um nível BAIXO em pelo menos uma entrada fazendo com que sua saída tenha um nível BAIXO, assim a saída final *X* a partir do inversor é nível ALTO. Essa saída de nível ALTO é então usada para ativar um indicador.

Indicador de nível baixo

▲ FIGURA 5–4

Problema relacionado Escreva a expressão Booleana para a lógica AND-OR-Inversor mostrada na Figura 5–4 e mostre que a saída é nível ALTO (1) quando duas entradas quaisquer dentre as entradas *A*, *B* e *C* estiverem em nível BAIXO (0).



176 ■ SISTEMAS DIGITAIS

3-9 ANÁLISE DE DEFEITO

A análise de defeito é o processo de reconhecer, isolar e corrigir um defeito ou falha num circuito ou sistema. Para ser um técnico de manutenção efetivo, o leitor precisa entender como o circuito ou sistema deve funcionar e ser capaz de reconhecer os problemas de funcionamento. Por exemplo, para determinar se uma porta lógica específica está ou não com defeito, o técnico tem que saber qual deve ser a resposta de saída para determinadas entradas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Testar CIs de portas lógicas pesquisando entradas e saídas abertas ■ Reconhecer os efeitos de uma entrada ou saída de CI em curto-circuito ■ Testar placas de circuito impresso pesquisando defeitos externos aos dispositivos ■ Fazer uma análise de defeito num freqüencímetro simples usando um osciloscópio

Falhas Internas em CIs de Portas Lógicas

Curtos-circuitos e circuitos abertos são os tipos mais comuns de defeitos internos às portas lógicas. Esses podem ocorrer nas entradas ou na saída de uma porta dentro do encapsulamento de um CI. *Antes de voltar a atenção em busca de qualquer defeito, verifique se as tensões de alimentação e GND são adequadas.*

Efeitos de uma Entrada Aberta Internamente Um circuito aberto internamente é o resultado de um componente aberto dentro do chip ou uma ruptura no pequeníssimo fio que interliga o chip do CI com o terminal do encapsulamento. Uma entrada aberta evita que um sinal na entrada chegue à saída da porta, conforme ilustra a Figura3–67(a) para o caso de uma porta NAND de 2 entradas. Uma entrada TTL aberta funciona efetivamente como um nível ALTO, de forma que os pulsos aplicados numa entrada boa chegam na saída de uma porta NAND como mostra a Figura 3–67(b).

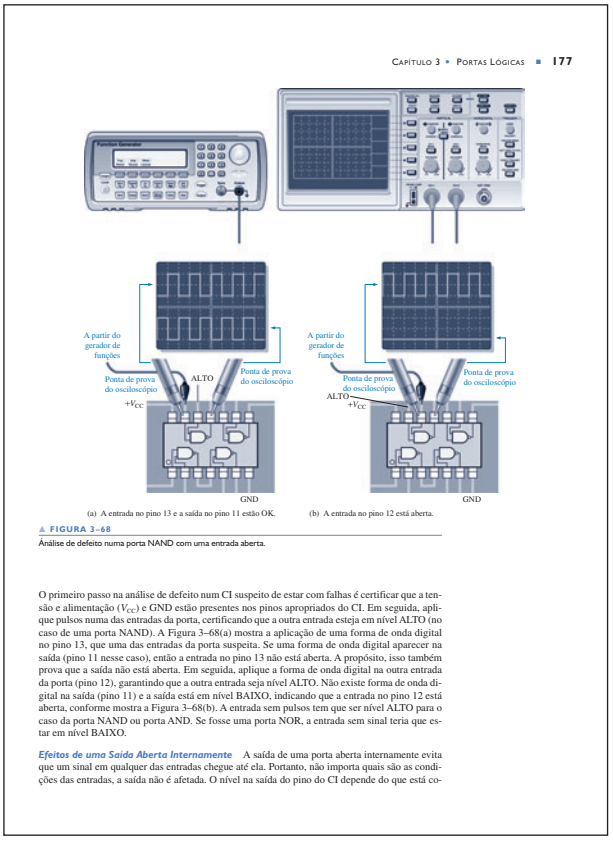
(a) A aplicação de pulsos numa entrada aberta não produz pulsos na saída.

(b) A aplicação de pulsos numa entrada em perfeito estado produz pulsos na saída para portas AND e NAND TTL porque uma entrada aberta funciona tipicamente como se estivesse em nível ALTO. No caso de um dispositivo CMOS, o estado é incerto.

▲ FIGURA 3–67
O efeito de uma entrada aberta numa porta NAND.

Condições para o Teste de Portas No teste de uma porta NAND ou porta AND, certifique sempre se as entradas sem sinal estão em nível ALTO para habilitar a porta. Na verificação do funcionamento de uma porta NOR ou porta OR, certifique sempre se as entradas sem sinal estão em nível BAIXO. No teste de uma porta EX-OR ou EX-NOR, não importa o nível lógico na entrada sem sinal porque os pulsos na outra entrada forçam as entradas a se alternarem entre os mesmos níveis lógicos e níveis lógicos opostos.

Análise de Defeito para Entradas Abertas A análise de defeito desse tipo de falha é facilmente realizada com o uso de um osciloscópio e um gerador de funções, conforme demonstra a Figura 3–68 para o caso de um encapsulamento com portas NAND de 2 entradas. Ao medir os sinais digitais com um osciloscópio, sempre use o acoplamento cc (dc).



CAPÍTULO 3 ■ PORTAS LÓGICAS ■ 177

(a) A entrada no pino 13 e a saída no pino 11 estão OK.

(b) A entrada no pino 12 está aberta.

▲ FIGURA 3–68
Análise de defeito numa porta NAND com uma entrada aberta.

O primeiro passo na análise de defeito num CI suspeito de estar com falhas é certificar que a tensão e alimentação (V_{CC}) e GND estão presentes nos pinos apropriados do CI. Em seguida, aplique pulsos numa das entradas da porta, certificando que a outra entrada esteja em nível ALTO (no caso de uma porta NAND). A Figura 3–68(a) mostra a aplicação de uma forma de onda digital no pino 13, que uma das entradas da porta suspeita. Se uma forma de onda digital aparecer na saída (pino 11 nesse caso), então a entrada no pino 13 não está aberta. A propósito, isso também prova que a saída não está aberta. Em seguida, aplique a forma de onda digital na outra entrada da porta (pino 12), garantindo que a outra entrada seja nível ALTO. Não existe forma de onda digital na saída (pino 11) e a saída está em nível BAIXO, indicando que a entrada no pino 12 está aberta, conforme mostra a Figura 3–68(b). A entrada sem pulsos tem que ser nível ALTO para o caso da porta NAND ou porta AND. Se fosse uma porta NOR, a entrada sem sinal teria que estar em nível BAIXO.

Efeitos de uma Saída Aberta Internamente A saída de uma porta aberta internamente evita que um sinal em qualquer das entradas chegue até ela. Portanto, não importa quais são as condições das entradas, a saída não é afetada. O nível na saída do pino do CI depende do que está co-

▲ **FIGURA P–4**

Páginas representativas da seção Análise de Defeito.

- Sistema de controle de um semáforo: Capítulos 6, 7 e 8
- Sistema de segurança: Capítulos 9 e 10

O tópico Aplicações em Sistemas Digitais pode ser tratado como opcional porque a sua omissão não afeta nenhum outro material neste livro. A Figura P–5 mostra uma parte de um tópico de Aplicações em Sistemas Digitais.

Final de capítulo Os seguintes itens, que ajudam no estudo, aparecem no final de cada capítulo:

- Resumo
- Glossário de termos importantes
- Autoteste
- Conjunto de problemas que incluem alguns ou todas as seguintes categorias: Básicos, Análise de Defeito, Aplicações em Sistemas Digitais, Projeto e Prática de Análise de Defeito Usando o Multisim
- Respostas das Revisões das Seções
- Respostas dos Problemas Relacionados aos Exemplos
- Respostas do Autoteste

Final do livro

- Apêndices: Conversão de códigos e tabelas de potências de dois (Apêndice A) e circuitos de interface do semáforo (Apêndice B)


364 ■ SISTEMAS DIGITAIS

APLICAÇÕES EM SISTEMAS DIGITAIS

Nesta seção de aplicações em sistemas digitais, começamos a trabalhar com um sistema de controle de semáforo de trânsito. Estabelecemos aqui os requisitos do sistema, desenvolvemos um diagrama em bloco e criamos um diagrama de estados para definir a seqüência de operação. Faremos o projeto da parte do sistema que envolve lógica combinacional e consideraremos os métodos de teste. A temporização e as partes seqüenciais do sistema serão tratadas nos Capítulos 7 e 8.

Requisitos Gerais de Sistema

Um controlador digital é necessário para controlar um semáforo de trânsito na interseção de uma via principal e uma via secundária. A via principal terá um sinal verde de pelo menos 25 s ou continuará verde enquanto não houver veículos na via secundária. A via secundária terá um sinal verde enquanto não existir veículos na via principal ou por um máximo de 25 s. Teremos um sinal amarelo de atenção durante 4 s entre a mudança do verde para o vermelho nas vias principal e secundária. Esses requisitos estão ilustrados no diagrama ilustrado na Figura 6–65.

Desenvolvimento de um Diagrama em Bloco do Sistema

A partir dos requisitos podemos desenvolver um diagrama em bloco do sistema. Primeiro, sabemos que o sistema tem que controlar seis diferentes pares de luz. Essas luzes são vermelho, amarelo e verde para as duas direções da via principal e vermelho, amarelo e verde nas duas direções da via secundária. Além disso, sabemos que existe uma entrada externa (além da alimentação) a partir do sensor de veículos na via secundária. A Figura 6–66 mostra um diagrama em bloco mínimo com esses requisitos.

Usando o diagrama em bloco mínimo do sistema, podemos começar a detalhá-lo. O sistema tem quatro estados, conforme indicado na Figura 6–65, assim é necessário um circuito lógico para controlar a seqüência de estados (lógica seqüencial). Além disso, são necessários circuitos para gerar os intervalos de tempo adequados de 25 s e 4 s que são necessários no sistema e para gerar um sinal de clock para a operação cíclica do sistema (circuitos de temporização). Os intervalos de tempo (longo e curto) e o sensor de veículo são entradas para a lógica seqüencial porque o seqüenciamento dos estados é uma função dessas variáveis. Os circuitos lógicos também são necessários para determinar em qual dos quatro estados o sistema está em qualquer momento especificado, para gerar as saídas adequadas para as luzes (decodificador de estados e circuito lógico de acionamento das luzes) e para iniciar os intervalos de tempo longo e curto. Os circuitos de interface são incluídos no semáforo e a unidade de interface para converter os níveis de saída do circuito de acionamento das luzes para as tensões e correntes necessárias para ligar cada uma das luzes. A Figura 6–67 mostra um diagrama em bloco mais detalhado mostrando esses elementos essenciais.

Diagrama de Estados

Um diagrama de estados mostra graficamente a seqüência de estados num sistema e as condições para cada estado e para as transições de um estado para o próximo. Na realidade, a Figura 6–65 é uma forma de diagrama de estados porque mostra a seqüência de estados e as condições.

Definição de Variáveis Antes que um diagrama de estados tradicional possa ser desenvolvido, as variáveis é que determinam como as seqüências do sistema através dos estados tem que ser definidas. Essas variáveis e os seus símbolos são apresentadas a seguir:

- Presença de veículo na via secundária = V_s
- Temporizador de 25 s (temporizador longo) é *ligado* = T_L
- Temporizador de 4 s (temporizador curto) é ligado = T_s

O uso de variáveis complementadas indica as condições opostas. Por exemplo, $\overline{V}_s$ indica que não existe veículo na via secundária, $\overline{T}_L$ indica que o temporizador longo está desligado (*off*), T_s indica que o temporizador curto está desligado (*off*).

Descrição do Diagrama de Estados Um diagrama de estados é mostrado na Figura 6–68. Cada um dos quatro estados é indicado com uma seqüência de código

Via principal / Via secundária

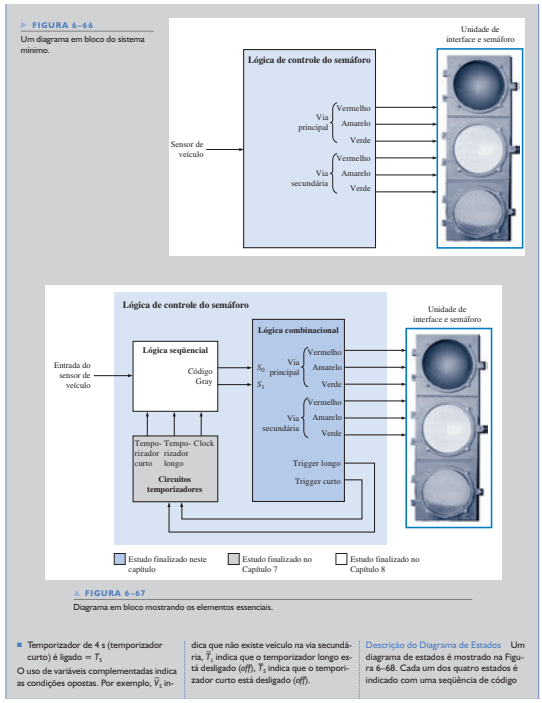
Primeiro estado: pelo menos 25 segundos ou enquanto não existir veículo na via secundária

Segundo estado: 4 segundos

Terceiro estado: máximo de 25 segundos ou até que não tenha veículo na via secundária

Quarto estado: 4 segundos

▲ FIGURA 6–65
Requisito para a seqüência do semáforo de trânsito.



▲ FIGURA 6–67
Diagrama em bloco mostrando os elementos essenciais.

CAPÍTULO 6 • FUNÇÕES DE LÓGICA COMBINACIONAL ■ 365

▲ FIGURA 6–66
Um diagrama em bloco do sistema mínimo.

▲ FIGURA P–5

Páginas representativas da seção Aplicações em Sistemas Digitais.

- Respostas dos problemas de número ímpar
- Glossário
- Índice

Para o estudante

A tecnologia digital é fascinante! A maioria dos equipamentos/serviços já é digital ou passará a ser num futuro próximo. Por exemplo, os telefones celulares e outros tipos de comunicações sem fio, televisão, rádio, controle de processos, eletrônica automotiva, equipamentos de eletrônicos de consumo, navegação global, sistemas militares, para citar apenas algumas aplicações, dependem intensamente da eletrônica digital.

Uma abordagem consistente dos fundamentos da tecnologia digital irá prepará-lo para atividades altamente qualificadas e bem remuneradas no futuro próximo. O mais importante é entender os seus fundamentos essenciais. Com isso, você poderá se desenvolver em qualquer área.

Além disso, a lógica programável está se tornando extremamente importante na tecnologia atual sendo que tópicos relacionados são apresentados nesse livro. É claro que, uma análise de defeito eficiente é uma habilidade que também é bastante desejada. Este livro também aborda métodos de análise de defeito e teste a partir de testes tradicionais a técnicas de fabricação, tais como *bed-of-nails*, *flying probe* e *boundary scan*. Esses são exemplos de experiências que você pode adquirir com um esforço aplicado ao estudo dos conceitos apresentados.

Os CD-ROMs Dois CDs acompanham este livro. Um contém folhas de dados (*data sheets*) de circuitos integrados digitais. O outro contém arquivos de circuitos para uso com o software Multisim Versão 2001 ou 7 (os arquivos dos circuitos para essas versões — bem como para uso com o Multisim 8 — também estão disponíveis no site **www.prenhall.com/floyd**).

"Guia do usuário" para professores

Geralmente o tempo ou a ênfase do programa do curso determina quais tópicos serão abordados. Não é incomum omitir ou condensar tópicos ou alterar sua seqüência para personalizar o material para um curso em particular. O autor reconhecendo isso projetou este livro especificamente para prover uma grande flexibilidade na abordagem dos tópicos.

O livro é organizado em torno de um núcleo de tópicos fundamentais que são, em sua maioria, essenciais em qualquer curso de eletrônica. Inclui outros tópicos que podem ser incluídos ou omitidos dependendo da ênfase do curso ou de outros fatores. A Figura P–6 ilustra esse conceito de modularidade.

Fundamentos essenciais Os tópicos fundamentais da lógica digital, em sua maior parte, devem ser abordados em todos os programas. Em conexão com esse núcleo estão alguns tópicos "satélites" que podem ser considerados para omissão ou inclusão, dependendo dos objetivos do curso. Qualquer bloco em torno do núcleo pode ser omitido sem afetar os fundamentos essenciais.

Lógica programável Esse tópico pode ser omitido, mas é recomendável a sua abordagem se possível. A lógica programável pode ser abordada de forma superficial ou à medida que for considerada prática, de acordo com o programa.

Análise de defeito As seções de análise de defeito aparecem em muitos capítulos.

Aplicações em sistemas digitais As aplicações de sistemas aparecem em vários capítulos.

Tecnologias de circuitos integrados Alguns ou todos os tópicos do Capítulo 14 podem ser abordados caso seja desejado discutir detalhes dos circuitos internos de CIs digitais.

Tópicos especiais Esses tópicos são: Introdução aos Computadores e Processamento de Sinais Digitais nos Capítulos 12 e 13 respectivamente. Esses são tópicos especiais e podem não ser essenciais ao seu curso.

A partir do diagrama da Figura P–6, podemos omitir ou enfatizar tópicos em função de restrições de tempo ou outras prioridades. Por exemplo, nos fundamentos essenciais, códigos de correção de erros, somadores com carry antecipado, projeto lógico seqüencial e outros tópicos selecionados podem ser omitidos.

Personalização do sumário Você pode "percorrer" os tópicos abordados neste livro de várias maneiras, dependendo dos objetivos do seu programa. Caso você escolha uma abordagem mínima apenas com os fundamentos essenciais, uma abordagem completa de todos os tópicos ou qualquer abordagem intermediária, este livro se adapta às suas necessidades. O sumário é codi-

FIGURA P–6

ficado em cores as quais coincidem com as cores dos blocos na Figura P–6. Isso permite identificação dos tópicos para omissão ou inclusão personalizando o seu curso.

Algumas opções para o uso deste livro são mostradas a seguir em termos dos tópicos codificados em cores conforme a Figura P–6. Outras opções também são possíveis incluindo abordagem parcial de alguns tópicos.

Opção 1 ■
Opção 2 ■■
Opção 3 ■■□
Opção 4 ■■□■
Opção 5 ■■□■■■

Agradecimentos

Este livro inovador é o resultado dos esforços e das habilidades de muitas pessoas. Eu penso que alcançamos o nosso objetivo, que foi a produção de um livro insuperável. Na Prentice Hall, Kate Linsner e Rex Davidsom contribuíram bastante com tempo, talento e esforço para conduzir esse projeto em suas diversas fases para produzir esse livro conforme você pode constatar. Lois Porter fez um trabalho fantástico de edição dos manuscritos. Ela desvendou os mistérios das observações e das anotações freqüentemente quase ilegíveis desse autor e, a partir dessa bagunça, extraiu um manuscrito editado inacreditavelmente organizado e esplêndido. Além disso, Jane Lopez fez outro belo trabalho com os gráficos. Outra pessoa que contribuiu significativamente para este livro foi Gary Snyder, fornecendo todos os arquivos de circuitos Multisim (nas versões 2001, 7 e 8 que se encontram no Companion Website **www.prenhall.com/floyd**). Eu estendo os meus agradecimentos e reconhecimento a todas essas pessoas e também àquelas que indiretamente estiveram envolvidas nesse projeto.

Na revisão deste e de todos os outros livros, dependo dos comentários de muitos usuários e não-usuários. Quero oferecer os meus sinceros agradecimentos aos seguintes revisores, aos quais submeti muitas sugestões valiosas e forneceram muitas críticas construtivas: Bo Barry, University of North Carolina-Charlotte; Chuck McGlumphy, Belmont Thecnical College; e Amy Ray, Mitchell Community College.

Minhas apreciações para David Buchla pelo seu esforço em garantir que o manual de laboratório ficasse em conformidade com o livro-texto e pelas suas observações valiosas. Gostaria também de mencionar Muhammed Arif Shabir pelas suas sugestões concernentes a registradores de deslocamento.

Agradeço a todos os membros da equipe de vendas da Prentice Hall cujos esforços têm ajudado a tornar este livro acessível a um grande número de leitores pelo mundo. Além disso, agradeço a todos que adotam este livro em suas escolas ou para uso próprio. Sem vocês, não estaríamos neste negócio. Espero que este livro seja uma valiosa ferramenta de aprendizagem e uma referência para os estudantes.

Tom Floyd

Sumário

♦ Tópicos que podem ser considerados opcionais.

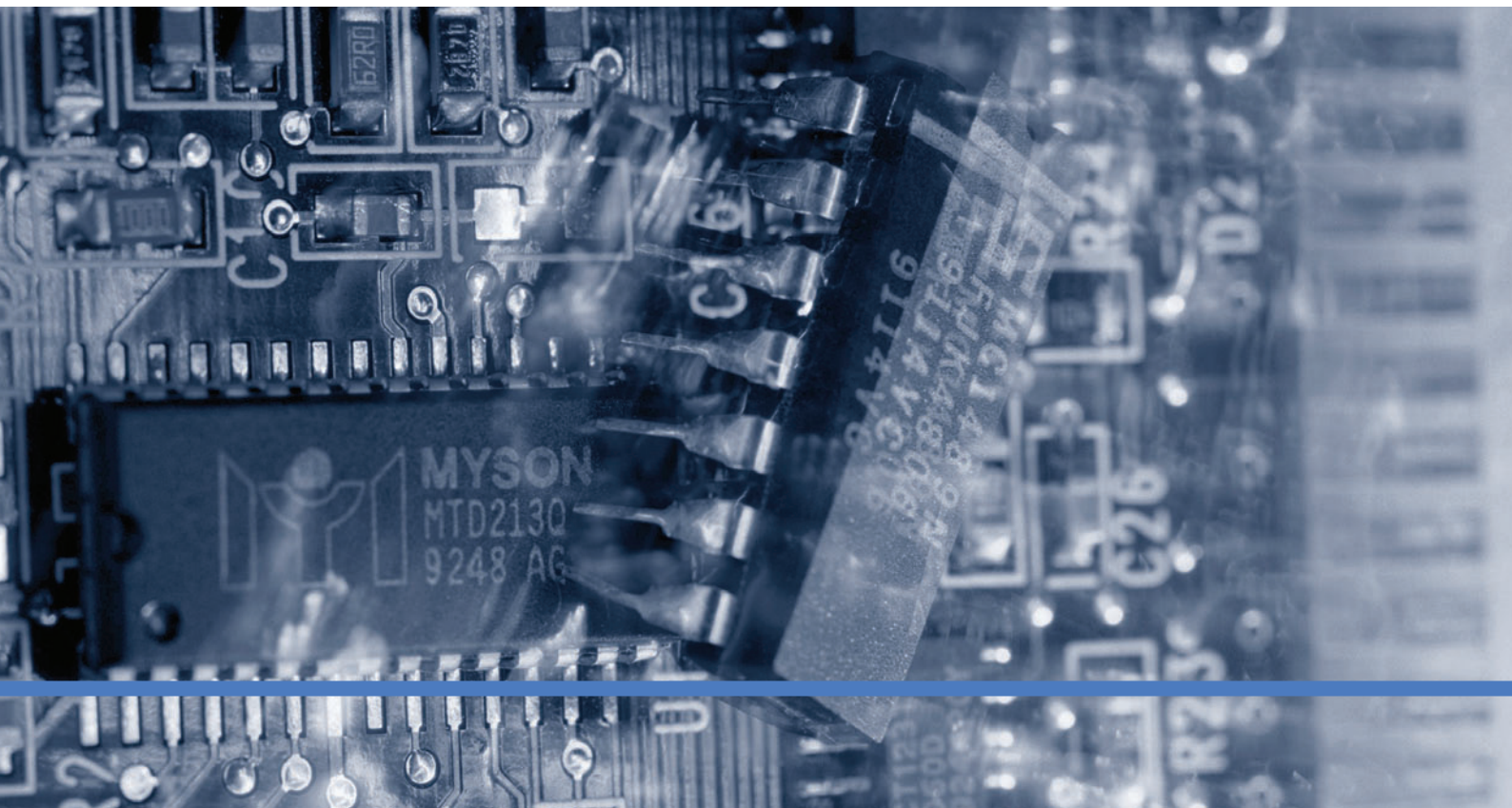
1

Eletrônica Digital – Conceitos

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Explicar as diferenças básicas entre grandezas analógicas e digitais
- Mostrar como os níveis de tensão são usados para representar grandezas digitais
- Descrever os diversos parâmetros da forma de onda de um pulso tais como tempo de subida, tempo de descida, largura de pulso, freqüência, período e ciclo de trabalho
- Explicar as operações lógicas básicas AND, OR e NOT
- Descrever as funções lógicas dos circuitos comparador, somador, conversor de código, codificador, decodificador, multiplexador, demultiplexador, contador e registrador



- Identificar circuitos integrados digitais de funções fixas, de acordo com a complexidade deles, e os tipos de encapsulamentos de CIs
- Identificar a numeração de pinos nos encapsulamentos dos circuitos integrados
- Descrever a lógica de programação, discutir os diversos tipos e descrever como são programados os PLDs
- Reconhecer os diversos instrumentos e compreender como eles são usados em medições e análise de defeito em sistemas e circuitos digitais
- Mostrar como um sistema digital completo é formado combinando as funções básicas em aplicações práticas

TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- Analógico
- Digital
- Binário
- Bit
- Pulso
- Clock
- Diagrama de temporização
- Dados
- Serial
- Paralelo
- Lógica
- Entrada
- Saída
- Porta
- NOT
- Inversor
- AND
- OR
- Circuito integrado (CI)
- SPLD
- CPLD
- FPGA
- Compilador
- Análise de defeito

INTRODUÇÃO

O termo *digital* é derivado da forma com que os computadores realizam operações, contando dígitos. Durante muitos anos, as aplicações da eletrônica digital ficaram confinadas aos sistemas computacionais. Hoje em dia, a tecnologia digital é aplicada em diversas áreas além da área computacional. Aplicações como televisão, sistemas de comunicação, radar, sistemas de navegação e direcionamento, sistemas militares, instrumentação médica, controle de processos industriais e equipamentos eletrônicos de consumo usam técnicas digitais. Ao longo dos anos a tecnologia digital tem progredido desde os circuitos com válvulas, passando pelos circuitos com transistores discretos, até os circuitos integrados complexos, alguns dos quais contêm milhões de transistores.

Esse capítulo fornece uma introdução à eletrônica digital e propicia uma ampla visão dos diversos conceitos, componentes e ferramentas importantes.

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

A abordagem adotada no final de vários capítulos deste livro faz uso de aplicações em sistemas para explorar os principais assuntos do capítulo. Cada sistema é projetado de forma a ser adequado aos assuntos tratados para ilustrar como a teoria e os dispositivos podem ser usados. Cinco sistemas diferentes são apresentados ao longo do livro, sendo que alguns abordam dois ou mais capítulos.

Todos os sistemas são simplificados para torná-los adequados ao contexto do material do capítulo. Embora esses sistemas sejam baseados em requisitos de sistemas reais, eles são projetados de forma a se ajustarem aos tópicos desenvolvidos sem a intenção de representar a abordagem mais eficiente ou definitiva para uma determinada aplicação.

Este capítulo introduz o primeiro sistema, o qual se enquadra na área de controle de processos industriais e que faz a contagem e o controle de itens no empacotamento numa linha transportadora. Este foi projetado para incorporar todas as funções lógicas que são introduzidas neste capítulo de forma que o leitor possa ver como elas são usadas e como funcionam em conjunto para alcançar um objetivo.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

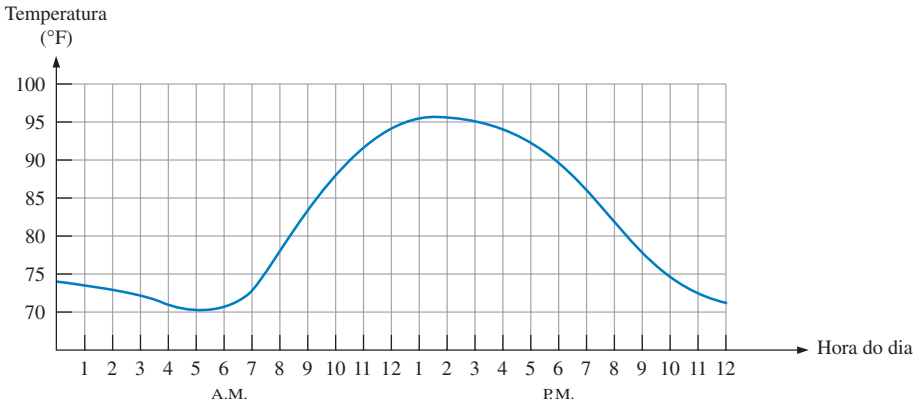
1-1 GRANDEZAS ANALÓGICAS E DIGITAIS

Os circuitos eletrônicos podem ser divididos em duas grandes categorias, digitais e analógicos. A eletrônica digital envolve grandezas com valores discretos e a eletrônica analógica envolve grandezas com valores contínuos. Ainda que o leitor estude os fundamentos da eletrônica digital neste livro, deve conhecer também algo sobre eletrônica analógica, pois muitas aplicações requerem conhecimentos das duas áreas; são igualmente importantes os conhecimentos relativos ao interfaceamento entre essas áreas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir grandeza *analógica* ■ Definir grandeza *digital* ■ Explicar a diferença entre grandezas analógicas e digitais ■ Enunciar as vantagens dos sistemas digitais sobre os analógicos ■ Apresentar exemplos de como as grandezas digitais e analógicas são usadas em eletrônica

Uma grandeza **analógica*** é aquela que apresenta valores contínuos. Uma grandeza **digital** é aquela que apresenta valores discretos. A maioria daquilo que se pode medir quantitativamente na natureza se encontra na forma analógica. Por exemplo, a temperatura do ar varia numa faixa contínua de valores. Durante um determinado dia, a temperatura não passa, digamos, de 71° F para 72° F (~21,7° C para ~22,2° C) instantaneamente; ela passa por toda uma infinidade de valores intermediários. Se fizermos um gráfico da temperatura em um dia de verão típico, teremos uma curva contínua e de variação suave similar à curva mostrada na Figura 1–1. Outros exemplos de grandezas analógicas são tempo, pressão, distância e som.

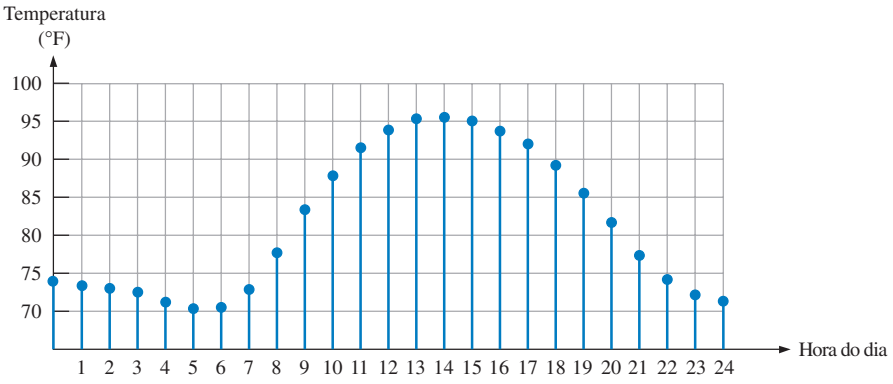


► **FIGURA 1–1**
Gráfico de uma grandeza analógica (temperatura *versus* tempo).

Em vez de fazer um gráfico da temperatura em uma base contínua, suponha que façamos a leitura da temperatura apenas a cada hora. Agora temos valores amostrados que representam a temperatura em pontos discretos no tempo (de hora em hora) ao longo de um período de 24 horas, conforme indicado na Figura 1–2. Efetivamente convertemos uma grandeza analógica em um formato que podemos agora digitalizar representando cada valor amostrado por um código digital. É importante perceber que a Figura 1–2 não é propriamente uma representação digital de uma grandeza analógica.

Vantagens dos Sistemas Digitais A representação digital tem certas vantagens sobre a representação analógica em aplicações eletrônicas. Para citar uma, dados digitais podem ser processados e transmitidos de forma mais eficiente e confiável que dados analógicos. Além disso, dados digitais possuem uma grande vantagem quando é necessário armazenamento (memorização). Por exemplo, a música quando convertida para o formato digital pode ser armazenada de forma mais compacta e reproduzida com maior precisão e pureza que quando está no formato analógico. O ruído (flutuações indesejadas na tensão) quase não afeta os dados digitais tanto quanto afeta os sinais analógicos.

* Todos os termos destacados em negrito são importantes e estão definidos no Glossário que se encontra no final do livro. Aqueles termos em negrito e em cor são termos-chave e estão incluídos nos Termos Importantes no final de cada capítulo.

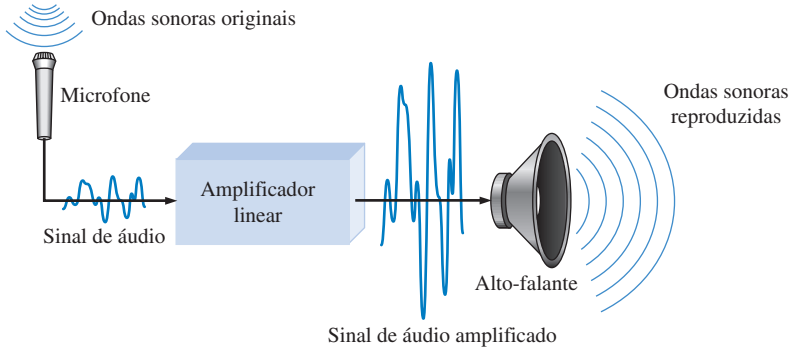


◄ **FIGURA 1–2**

Representação dos valores amostrados (quantização) da grandeza analógica mostrada na Figura 1–1. Cada valor representado por um ponto pode ser digitalizado sendo representado por um código digital que consiste de uma série de 1s e 0s.

Um Sistema Eletrônico Analógico

Um sistema de amplificação de som que pode ser ouvido por uma grande quantidade de pessoas é um exemplo simples de uma aplicação da eletrônica analógica. O diagrama básico na Figura 1–3 ilustra as ondas sonoras, que são de natureza analógica, sendo captadas por um microfone e convertidas em uma pequena tensão analógica denominada sinal de áudio. Essa tensão varia continuamente de acordo com as variações no volume e na freqüência do som e é aplicada na entrada de um amplificador linear. A saída do amplificador, que é uma reprodução ampliada da tensão de entrada, é enviada para o(s) alto-falante(s). O alto-falante converte o sinal de áudio amplificado de volta para o formato de ondas sonoras com um volume muito maior que as ondas sonoras originais captadas pelo microfone.

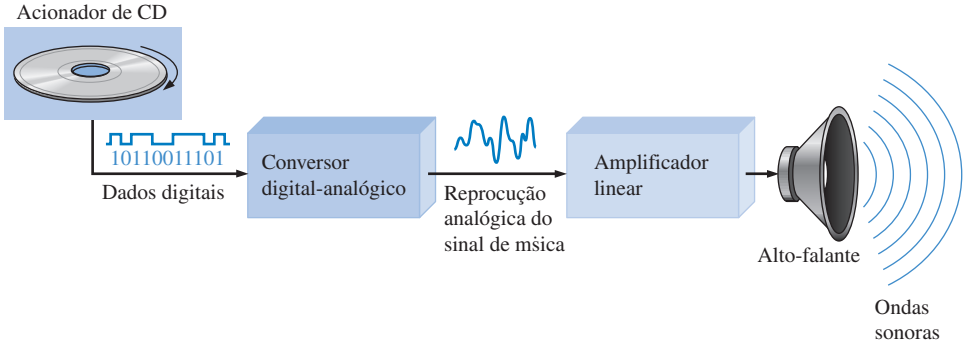


◄ **FIGURA 1–3**

Um sistema básico de amplificação de áudio.

Um Sistema que Usa Métodos Analógicos e Digitais

O aparelho de CD (*compact disk*) é um exemplo de um sistema no qual são usados tanto circuitos digitais quanto analógicos. O diagrama em bloco simplificado que é visto na Figura 1–4 ilustra o princípio básico. A música no formato digital é armazenada no CD. Um sistema óptico com diodo laser capta os dados digitais a partir do disco girante e os transfere para um **conversor digital-analógico** (**DAC** – *digital-to-analog converter*).



◄ **FIGURA 1–4**

Diagrama em bloco básico de um aparelho de CD. Apenas um canal é mostrado.

O DAC converte os dados digitais em um sinal analógico que é uma reprodução elétrica da música original. Esse sinal é amplificado e enviado ao auto-falante para que as pessoas apreciem. Quando a música é gravada originalmente no CD, um processo essencialmente contrário ao que descrevemos aqui acontece usando um **conversor analógico-digital** (**ADC** – *analog-to-digital converter*).

SEÇÃO 1–1 REVISÃO
As respostas estão no final do capítulo.

1. Escreva o significado de *analógico*.
2. Escreva o significado de *digital*.
3. Explique a diferença entre uma grandeza digital e uma grandeza analógica.
4. Cite um exemplo de um sistema analógico e de um outro sistema que combina técnicas analógicas e digitais. Cite também um sistema totalmente digital.

1-2 DÍGITOS BINÁRIOS, NÍVEIS LÓGICOS E FORMAS DE ONDA DIGITAIS

A eletrônica digital envolve circuitos e sistemas nos quais existem apenas dois estados possíveis. Esses estados são representados por dois níveis de tensão diferentes: um ALTO e um BAIXO. Os dois estados também podem ser representados por níveis de corrente, bits e ressaltos num CD ou DVD, etc. Em sistemas digitais tais como computadores, as combinações de dois estados, denominadas *códigos*, são usadas para representar números, símbolos, caracteres alfabéticos e outros tipos de informações. O sistema de numeração de dois estados é denominado de *binário* e os seus dois dígitos são 0 e 1. Um dígito binário é denominado de *bit*.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir *binário* ■ Definir *bit* ■ Especificar os bits num sistema binário ■ Explicar como os níveis de tensão são usados para representar bits ■ Explicar como os níveis de tensão são interpretados por um circuito digital ■ Descrever as características gerais de um pulso ■ Determinar a amplitude, o tempo de subida, o tempo de descida e a largura de um pulso ■ Identificar e descrever as características de uma forma de onda digital ■ Determinar a amplitude, o período, a freqüência e o ciclo de trabalho de uma forma de onda digital ■ Explicar o que é um diagrama de temporização e dizer qual a finalidade dele ■ Explicar a transferência serial e paralela de dados e dizer quais são as vantagens e desvantagens de cada uma

Dígitos Binários

NOTA: COMPUTAÇÃO

O conceito de um computador digital pode nos levar de volta a Charles Babage, quem desenvolveu um dispositivo computacional mecânico primitivo nos anos de 1830. John Atanasoff foi o primeiro a aplicar o processamento eletrônico à computação digital em 1939. Em 1946, um computador digital eletrônico chamado ENIAC foi implementado com circuitos à válvula. Mesmo ocupando toda uma sala, o ENIAC não tinha o poder computacional das nossas calculadoras portáteis.

Cada um dos dois dígitos de um sistema **binário**, 1 e 0, é denominado **bit**, uma contração das palavras ***bi****nary digi****t*** (dígito binário). Em circuitos digitais, dois níveis de tensão diferentes são usados para representar os dois bits. Geralmente, 1 é representado pela tensão maior, a qual chamamos de nível ALTO, e o 0 é representado pelo nível de tensão menor, o nível BAIXO. Essa forma de representação é denominada lógica positiva e é usada ao longo desse livro.

ALTO = 1 e BAIXO = 0

Um outro sistema no qual o 1 é representado por um nível BAIXO e o 0 é representado por um nível ALTO é chamado de *lógica negativa*.

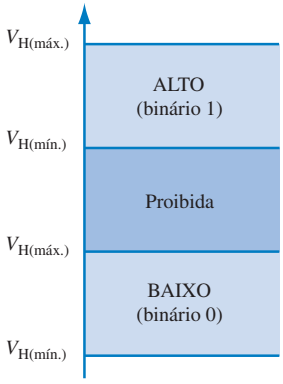
Grupos de bits (combinação de 1s e 0s), denominados *códigos*, são usados para representar números, letras, símbolos, instruções e qualquer outro tipo de grupo necessário para uma determinada aplicação.

Níveis Lógicos

As tensões usadas para representar 1 e 0 são denominados *níveis lógicos*. Teoricamente, um nível de tensão representa um nível ALTO e o outro representa um nível BAIXO. Entretanto, em um circuito digital prático, um nível ALTO pode ser qualquer tensão entre um valor mínimo e um valor

máximo especificados. Da mesma forma, um nível BAIXO pode ser qualquer valor de tensão entre um valor mínimo e máximo especificados. Não existe sobreposição entre as faixas aceitáveis para os níveis ALTO e BAIXO.

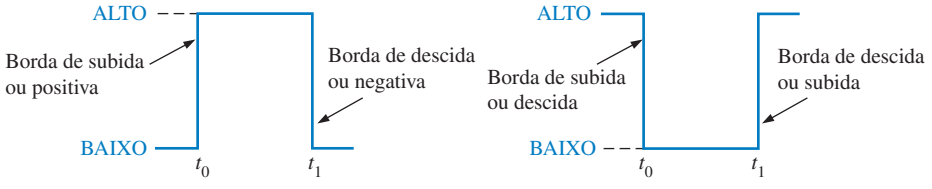
A Figura 1–5 ilustra as faixas dos níveis ALTO e BAIXO para um circuito digital. A variável $V_{H(máx)}$ representa o valor máximo de tensão para o nível ALTO e $V_{H(mín)}$ representa o valor mínimo de tensão para o nível ALTO. O valor máximo de tensão para o nível BAIXO é representado por $V_{L(máx)}$ e o valor mínimo de tensão para o nível BAIXO é representado por $V_{L(mín)}$. Os valores de tensão entre $V_{L(máx)}$ e $V_{H(mín)}$ são inaceitáveis para uma operação adequada. Uma tensão na faixa proibida pode ser interpretada tanto como um nível ALTO quanto um nível BAIXO por um determinado circuito sendo, portanto, valores inaceitáveis. Por exemplo, os valores referentes ao nível ALTO para um determinado circuito digital chamado de CMOS pode variar de 2 V a 3,3 V e os valores referentes ao nível BAIXO podem varia de 0 a 0,8 V. Assim, por exemplo, se uma tensão de 2,5 V for aplicada, o circuito interpretará como um nível BAIXO ou binário 0. Para esse tipo de circuito, as tensões entre 0,8 V e 2 V não são permitidas.



◄ FIGURA 1–5
Faixas de níveis lógicos de tensão para um circuito digital.

Formas de Onda Digitais

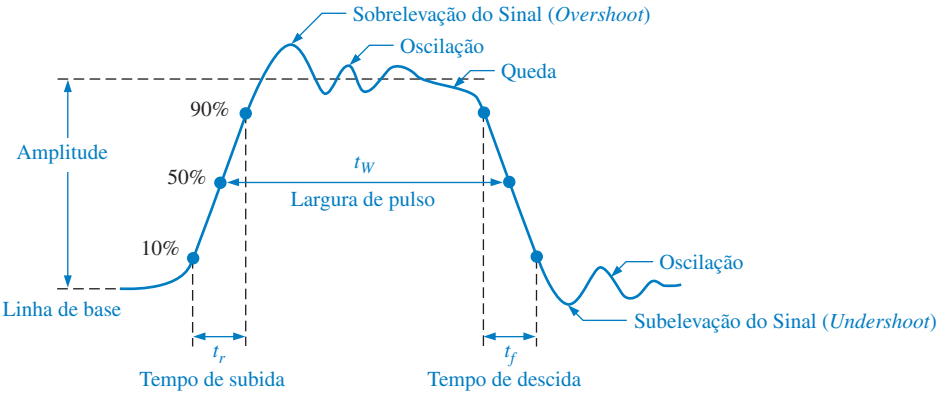
Formas de onda digitais consistem em níveis de tensão que comutam entre os níveis, ou estados, lógicos ALTO e BAIXO. A Figura 1–6(a) mostra que um único **pulso** positivo é gerado quando a tensão (ou corrente) passa do nível BAIXO normal para o nível ALTO e em seguida retorna para o nível BAIXO. O pulso negativo, visto na Figura 1–6(b), é gerado quando a tensão passa do nível ALTO normal para o nível BAIXO e retorna para o nível ALTO. Uma forma de onda digital é constituída de uma série de pulsos.



◄ FIGURA 1–6
Pulsos ideais.

O Pulso Conforme indicado na Figura 1–6, um pulso tem duas bordas: a **borda positiva**, que ocorre primeiro no instante t_0, e uma **borda negativa**, que ocorre depois no instante t_1. Para um pulso positivo, a borda positiva é uma borda de subida e a borda negativa é uma borda de descida. Os pulsos vistos na Figura 1–6 são ideais porque se considera que as bordas de subida e descida comutam num tempo zero (instantaneamente). Na prática, essas transições nunca ocorrem instantaneamente, embora para a maioria dos circuitos digitais funcionarem consideramos pulsos ideais.

A Figura 1–7 mostra um pulso não-ideal. Na realidade, todos os pulsos exibem algumas, ou todas, essas características. A sobrelevação do sinal (*overshoot*) e oscilações são produzidas al-



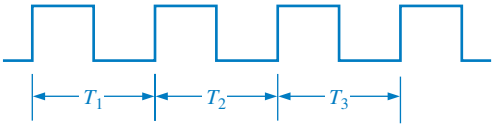
▶ **FIGURA 1–7**

Características de um pulso não ideal.

gumas vezes por efeitos de indutância e capacitância parasitas. A inclinação pode ser causada por capacitância parasita e circuitos resistivos que formam um circuito *RC* com uma pequena constante de tempo.

O tempo necessário para um pulso passar do nível BAIXO para o nível ALTO é denominado **tempo de subida** (t_r – *rise time*) e o tempo necessário para a transição do nível ALTO para o nível BAIXO é denominado **tempo de descida** (t_f – *fall time*). Na prática, é comum medir o tempo de subida a partir de 10% da amplitude do pulso (altura a partir da linha de base) até 90% da amplitude do pulso e para medir o tempo de descida consideramos o tempo de 90% a 10% da amplitude do pulso, conforme indicado na Figura 1–7. Os 10% da parte inferior e os 10% da parte superior não são incluídos nos tempos de subida e descida devido a não-linearidade da forma de onda nessas áreas. A **largura de pulso** (t_W – *pulse width*) é a medida da duração do pulso e é freqüentemente definida como o intervalo de tempo entre os pontos de 50% das bordas de subida e descida, conforme indicado na Figura 1–7.

Características de uma Forma de Onda A maioria das formas de onda encontradas em sistemas digitais são compostas de uma série de pulsos, algumas vezes denominados *trem de pulsos*, podendo ser classificadas como periódicas ou não-periódicas. Uma forma de onda **periódica** é aquela que se repete num intervalo fixo, denominado de **período** (***T***). A **freqüência** (***f***) é a taxa com que ela se repete e é medida em hertz (Hz). Uma forma de onda não-periódica, é claro, não se repete em intervalos fixos e pode ser composta de pulsos com larguras aleatórias e/ou intervalos aleatórios de tempo entre os pulsos. Um exemplo de cada tipo é mostrado na Figura 1–8.



Período = $T_1 = T_2 = T_3 = \ldots = T_n$

Freqüência = $\frac{1}{T}$

(a) Periódica (onda quadrada)

(b) Não-periódica

▲ **FIGURA 1–8**

Exemplos de formas de onda digitais.

A freqüência (*f*) de uma forma de onda digital é o inverso do período. A relação entre freqüência e período é expressa como:

Equação 1–1

$$f = \frac{1}{T}$$

Equação 1–2

$$T = \frac{1}{f}$$

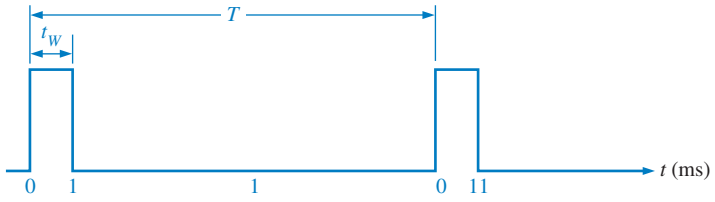
Uma característica importante de uma forma de onda digital periódica é o **ciclo de trabalho**, a razão entre a largura de pulso (t_W) e o período (T). O ciclo de trabalho pode ser expresso em porcentagem.

$$\text{Ciclo de trabalho} = \left(\frac{t_W}{T}\right)100\% \qquad \textbf{Equação 1–3}$$

EXEMPLO 1–1

Uma parte de uma forma de onda digital periódica é mostrada na Figura 1–9. As medidas estão em milissegundos. Determine:

(a) período **(b)** freqüência **(c)** ciclo de trabalho



▲ **FIGURA 1–9**

Solução

(a) O período é medido a partir da borda de um pulso até a borda correspondente do próximo pulso. Nesse caso T é medido entre duas bordas positivas, conforme indicado. T é igual a **10 ms**.

(b) $f = \dfrac{1}{T} = \dfrac{1}{10\text{ ms}} = \mathbf{100\ Hz}$

(c) $\text{Ciclo de trabalho} = \left(\dfrac{t_W}{T}\right)100\% = \left(\dfrac{1\text{ ms}}{10\text{ ms}}\right)100\% = \mathbf{10\%}$

Problema relacionado* Uma forma de onda digital periódica tem uma largura de pulso de 25 μs e um período de 150 μs. Determine a freqüência e o ciclo de trabalho.

* As respostas estão no final do capítulo.

Uma Forma de Onda Digital Transporta Informação Binária

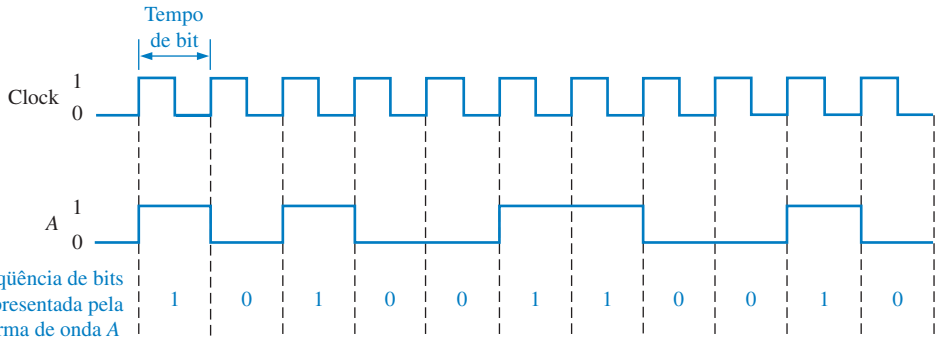
Uma informação binária manipulada por sistemas digitais aparece como formas de onda que representam seqüências de bits. Quando a forma de onda está em nível ALTO, um número binário 1 está presente; quando a forma de onda está em nível BAIXO, um binário 0 está presente. Cada bit na seqüência ocupa um intervalo de tempo definido denominado **tempo de bit**.

O Clock Em sistemas, todas as formas de onda são sincronizadas com uma forma de onda de temporização de referência denominada **clock**. O clock é uma forma de onda periódica na qual cada intervalo entre os pulsos (período) é igual ao tempo de um bit.

Um exemplo de uma forma de onda de clock é mostrado na Figura 1–10. Observe que, nesse caso, cada mudança de nível na forma de onda A ocorre na borda positiva da forma de onda do clock. De outra forma, as mudanças de nível ocorreriam na borda negativa do clock. Durante cada tempo de bit do clock, a forma de onda A é nível ALTO ou nível BAIXO. Esses ALTOs e BAIXOs representam uma seqüência de bits conforme indicado. Um grupo de vários bits pode ser usado como parte de uma informação binária, tal como um número ou uma letra. A forma de onda do clock por si só não transporta informação.

NOTA: COMPUTAÇÃO

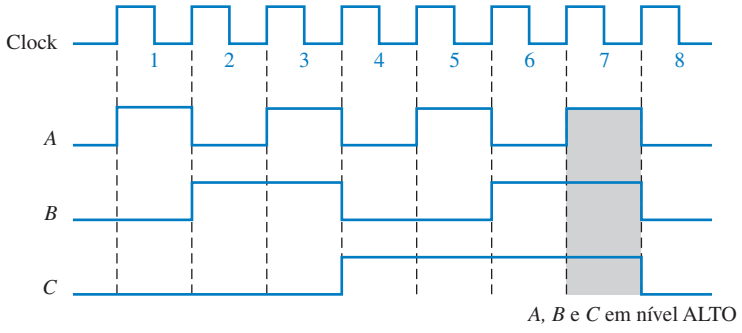
A velocidade na qual um computador pode operar depende do tipo do microprocessador usado no sistema. A especificação de velocidade, por exemplo 3,5 GHz, de um computador é a máxima freqüência de clock na qual o microprocessador pode operar.



▶ **FIGURA 1–10**

Exemplo de uma forma de onda de clock sincronizada com uma forma de onda que representa uma seqüência de bits.

Diagramas de Temporização Um **diagrama de temporização** é um gráfico de formas de onda digitais que mostra a relação atual de tempo de duas ou mais formas de onda e como cada forma de onda muda em relação às outras. Observando um diagrama de temporização, podemos determinar os estados (ALTO ou BAIXO) de todas as formas de onda em qualquer instante especificado e o momento exato que uma forma de onda muda de estado em relação às outras formas de onda. A Figura 1–11 é um exemplo de um diagrama de temporização composto de quatro formas de onda. A partir desse diagrama de temporização podemos ver, por exemplo, que as três formas de onda *A*, *B* e *C* são nível ALTO apenas durante o tempo de bit 7 e todas elas retornam para o nível BAIXO no final do tempo de bit 7 (área sombreada).



▶ **FIGURA 1–11**

Exemplo de um diagrama de temporização.

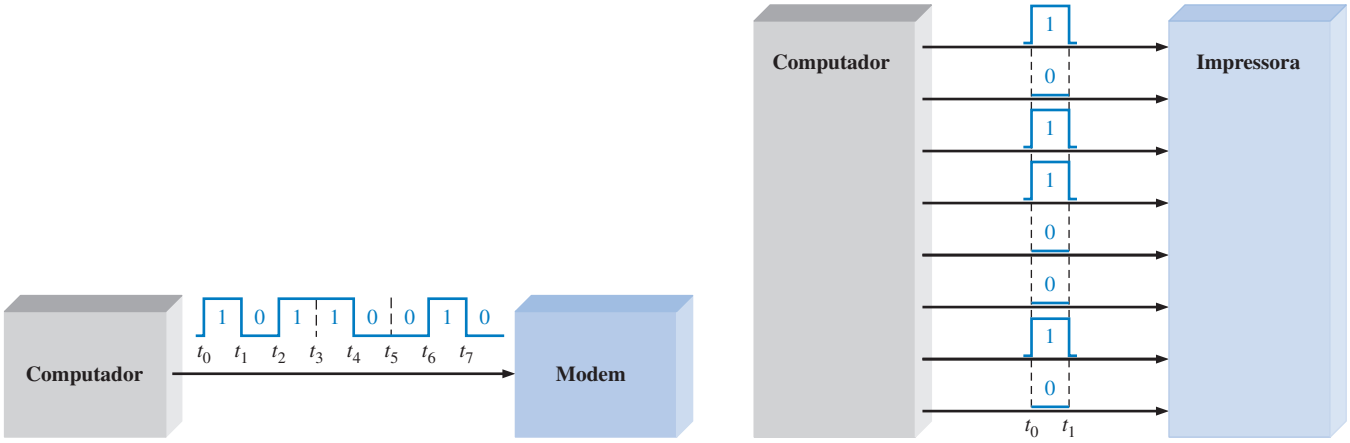
Tranferência de Dados

Dados se referem a grupos de bits que transportam algum tipo de informação. Dados em binário, que são representados por formas de onda digitais, têm que ser transferidos de um circuito para outro dentro de um sistema digital ou de um sistema para outro para cumprir um determinado propósito. Por exemplo, números armazenados em binário a partir de uma memória de computador têm que ser transferidos para a unidade central de processamento do computador para serem somados. O resultado da adição tem que ser transferido para um monitor e/ou transferido de volta para a memória. Em sistemas computacionais, conforme ilustrado na Figura 1–12, dados em binário são transferidos de duas formas: em série e em paralelo.

Quando bits são transferidos na forma **serial** de um ponto para outro, eles são enviados um bit de cada vez ao longo de uma única linha, conforme ilustrado na Figura 1–12(a) para o caso da transferência de um computador para um modem. Durante o intervalo de tempo de t_0 a t_1, o primeiro bit é transferido. Durante o intervalo de tempo de t_1 a t_2, o segundo bit é transferido, e assim por diante. Para transferir oito bits em série, se gastam oito intervalos de tempo.

Quando bits são transferidos no formato **paralelo**, todos os bits de um grupo são enviados em linhas separadas ao mesmo tempo. Existe uma linha para cada bit, conforme mostra a Figura 1–12(b) para o exemplo de oito bits sendo transferidos de um computador para uma impressora. Para transferir oito bits em paralelo, se gasta um intervalo de tempo comparado aos oito intervalos de tempo gastos na transferência serial.

Resumindo, uma vantagem da transferência serial de dados em binário é que um número mínimo de linhas é necessário. Na transferência em paralelo, é necessário um número de linhas igual



(a) Transferência serial de um dado Binário de 8 bits a partir de um Computador para um modem. O intervalo de t_0 a t_1 é enviado primeiro.

(b) Transferência paralela de um dado binário de 8 bits a partir de um computador para uma impressora. O instante inicial é t_0.

▲ **FIGURA 1–12**

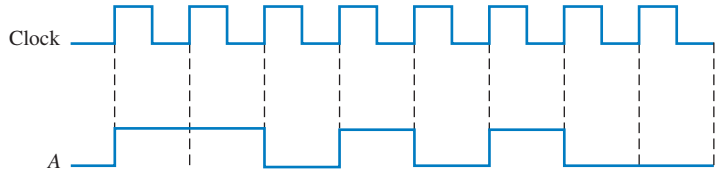
Ilustração de transferências serial e paralela de dados binários. Apenas as linhas de dados são mostradas.

ao número de bits. Uma desvantagem da transferência serial é que ela gasta um tempo maior, para transferir um determinado número de bits, que a transferência paralela. Por exemplo, se um bit pode ser transferido em 1 μs, então a transferência serial gasta 8 μs para transferir oito bits, porém gasta apenas 1 μs para a transferência paralela de oito bits. Uma desvantagem da transferência paralela é que ela necessita de mais linhas que a transferência serial.

EXEMPLO 1–2

(a) Determine o tempo total necessário para a transferência serial de oito bits contidos na forma de onda *A* vista na Figura 1–13 e indique a seqüência de bits. O bit mais à esquerda é o primeiro a ser transferido. Um clock de 100 kHz é usado como referência.

(b) Qual é o tempo total de transferência dos mesmos oito bits em paralelo?



▲ **FIGURA 1–13**

Solução **(a)** Como a freqüência do clock é 100 kHz, o período é

$$T = \frac{1}{f} = \frac{1}{100\text{ kHz}} = 10\ \mu\text{s}$$

Se gasta 10 μs para transferir cada bit da forma de onda. O tempo total de transferência para 8 bits é

$$8 \times 10\ \mu\text{s} = \mathbf{80\ \mu s}$$

Para determinar a seqüência de bits, examine a forma de onda apresentada na Figura 1–13 durante cada tempo de bit. Se a forma de onda *A* for nível ALTO durante o tempo

de bit, um 1 é transferido. Se a forma de onda for nível BAIXO durante o tempo de bit, um 0 é transferido. A seqüência de bits é ilustrada na Figura 1–14. O bit mais à esquerda é o primeiro a ser transferido.

▲ **FIGURA 1–14**

(b) Uma transferência paralela gastaria **10 μs** para todos os oito bits.

Problema relacionado Se dados em binário são transferidos a uma taxa de 10 milhões de bits por segundo (10 Mbits/s), quanto tempo é gasto para uma transferência paralela de 16 bits em 16 linhas? E para uma transferência serial de 16 bits?

SEÇÃO 1–2 REVISÃO

1. Defina *binário*.
2. O que significa *bit*?
3. O que são os bits em um sistema binário?
4. O que é o tempo de subida e o tempo de descida de um pulso medido?
5. Conhecendo o período de uma forma de onda, como se determina a freqüência?
6. Explique o que é uma forma de onda de clock.
7. Qual é a finalidade de um diagrama de temporização?
8. Qual é a principal vantagem da transferência paralela sobre a transferência de dados em binário?

1-3 OPERAÇÕES LÓGICAS BÁSICAS

Em sua forma básica, a lógica é o campo do raciocínio humano que nos diz que uma certa proposição (declaração) é verdadeira se certas condições forem verdadeiras. Proposições podem ser classificadas como verdadeiras ou falsas. Muitas situações e processos que encontramos em nossas vidas diariamente podem ser expressos na forma de funções proposicionais ou lógicas. Como tais funções são declarações verdadeiro/falso ou sim/não, os circuitos digitais com suas características de dois estados são aplicáveis.

Ao final do estudo desta seção você deverá ser capaz de:

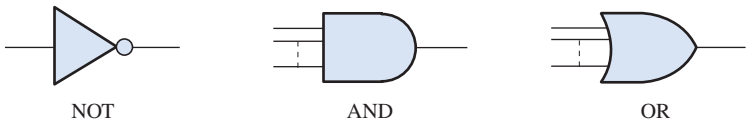
■ Fazer uma lista com as três operações lógicas básicas ■ Definir a operação NOT ■ Definir a operação AND ■ Definir a operação OR

Diversas proposições, quando combinadas, formam funções proposicionais ou lógicas. Por exemplo, a declaração proposicional "A luz está ligada" será verdadeira se "A lâmpada não está queimada" for verdadeira e se "A chave está ligada" for verdadeira. Portanto, a declaração lógica a seguir pode ser feita: *A luz está ligada apenas se a lâmpada não está queimada e a chave está ligada*. Nesse exemplo, a primeira declaração é verdadeira apenas se as duas últimas forem verdadeiras. A primeira declaração ("A luz está ligada") é a proposição básica e as outras duas declarações são as condições das quais a proposição depende.

Em 1850, o matemático e logicista irlandês Georg Boole desenvolveu um sistema matemático para formulação de declarações lógicas com símbolos de forma que pudessem ser resolvidos de uma forma similar à álgebra comum. A álgebra Booleana, como é conhecida hoje em dia, é aplicada no projeto e análise de sistemas digitais e será abordada em detalhes no Capítulo 4.

O termo **lógica** é aplicado a circuitos digitais usados para implementar funções lógicas. Diversos tipos de **circuitos** lógicos digitais são os elementos básicos que formam os blocos construtivos de sistemas digitais complexos como o computador. Agora estudaremos esses elementos e discutiremos as funções deles de uma forma bem geral. Os capítulos posteriores abordarão esses circuitos em detalhes.

As três operações lógicas básicas (NOT, AND e OR) estão indicadas pelos seus símbolos padrão na Figura 1–15. Outros símbolos padrão para essas operações lógicas serão apresentados no Capítulo 3. As linhas conectadas em cada símbolo são as **entradas** e **saídas**. As entradas estão do lado esquerdo de cada símbolo e a saída está do lado direito. Um circuito que executa uma operação lógica especificada (AND, OR) é denominado de **porta** lógica. As portas AND e OR podem ter um numero qualquer (duas no mínimo) de entradas conforme indicado pela linha pontilhada na figura.



◀ **FIGURA 1–15**
As operações lógicas básicas e os respectivos símbolos.

Em operações lógicas, as condições verdadeiro/falso mencionadas anteriormente são representadas por ALTO (verdadeiro) e BAIXO (falso). Cada uma das três operações lógicas básicas gera uma única resposta para um determinado conjunto de condições.

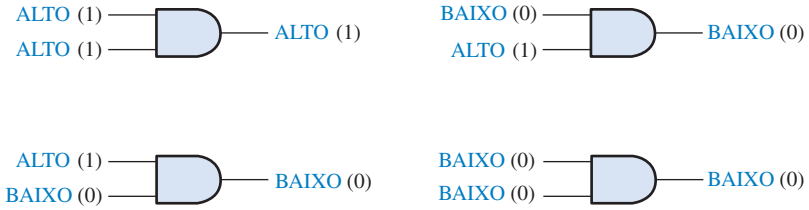
NOT

A operação **NOT** comuta de um nível lógico para o nível lógico oposto, conforme indicado na Figura 1–16. Quando a entrada for nível ALTO (1), a saída será nível BAIXO (0). Quando a entrada for nível BAIXO, a saída será nível ALTO. Nos dois casos, a saída *não* é o mesmo nível lógico que a entrada. A operação NOT é implementada por um circuito lógico conhecido como **inversor**.

◀ **FIGURA 1–16**
A operação NOT(inversão).

AND

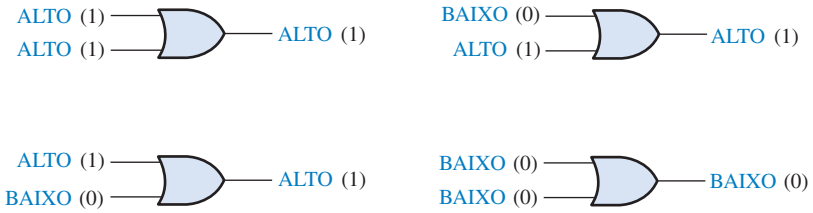
A operação **AND** gera uma saída de nível ALTO apenas quando todas as entradas forem nível ALTO, conforme indicado na Figura 1–17 para o caso de duas entradas. Quando um entrada for nível ALTO *e* a outra entrada for nível ALTO, a saída será nível ALTO. Quando qualquer uma, ou todas, as entradas forem nível BAIXO, a saída será nível BAIXO. A operação AND é implementada por um circuito lógico conhecido como *porta* AND.



◀ **FIGURA 1–17**
A operação AND.

OR

A operação **OR** gera uma saída de nível ALTO quando uma ou mais entradas forem nível ALTO, conforme indicado na Figura 1–18 para o caso de duas entradas. Quando uma entrada for nível ALTO *ou* a outra entrada for nível ALTO *ou* ambas forem nível ALTO, a saída será nível ALTO. Quando as duas entradas forem nível BAIXO, a saída será nível BAIXO. A operação OR é implementada por um circuito lógico conhecido como *porta* OR.



◀ FIGURA 1–18
A operação OR.

SEÇÃO 1–3 REVISÃO

1. Quando a operação NOT gera uma saída de nível ALTO?
2. Quando a operação AND gera uma saída de nível ALTO?
3. Quando a operação OR gera uma saída de nível ALTO?
4. O que é um inversor?
5. O que é uma porta lógica?

1-4 VISÃO GERAL DAS FUNÇÕES LÓGICAS BÁSICAS

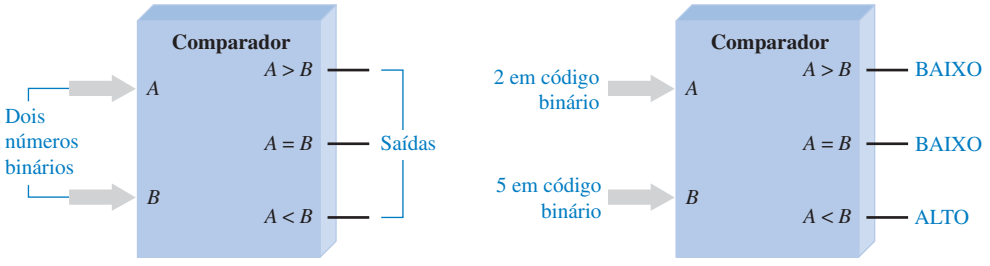
Os três elementos lógicos básicos AND, OR e NOT podem ser combinados para formar circuitos lógicos mais complexos que realizam diversas operações úteis e que são usados para construir sistemas digitais completos. Algumas das funções lógicas comuns são: comparação, aritmética, conversão de código, codificação, decodificação, seleção de dados, armazenamento e contagem. Esta seção apresenta uma visão geral dessas funções importantes de forma que possamos iniciar o estudo de como elas formam os blocos construtivos dos sistemas digitais tais como os computadores. Cada uma das funções lógicas básicas será abordada em detalhes em capítulos posteriores.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar nove tipos básicos de funções lógicas ■ Descrever um comparador de magnitude básico ■ Fazer uma lista das quatro funções aritméticas ■ Descrever um somador básico ■ Descrever um codificador básico ■ Descrever um decodificador básico ■ Definir multiplexação e demultiplexação ■ Dizer como é realizado o armazenamento de dados ■ Descrever a função de um contador básico

A Função de Comparação

A comparação de **magnitude** é realizada por um circuito lógico denominado **comparador**, abordado no Capítulo 6. Um comparador compara dois números e indica se eles são iguais ou não. Por exemplo, suponha que temos dois números e desejamos saber se eles são iguais ou não e, caso não sejam iguais, qual deles é maior. A função de comparação é representada na Figura 1–19. Um nú-



▶ FIGURA 1–19
A função de comparação.

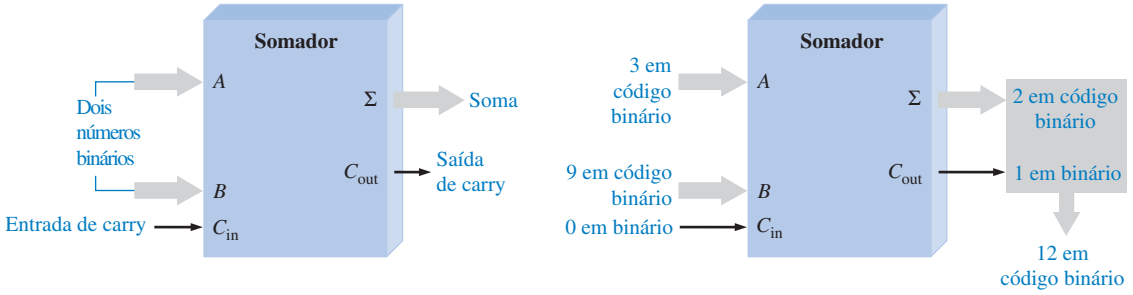
(a) Comparador de magnitude básico

(b) Exemplo: A é menor que B ($2 < 5$) conforme indicado pelo nível ALTO na saída ($A < B$)

mero na forma binária (representado por níveis lógicos) é aplicado na entrada *A*, e o outro número na forma binária (representado por níveis lógicos) é aplicado na entrada *B*. A saída indica a relação entre os dois números gerando um nível ALTO na linha de saída apropriada. Suponha que uma representação binárias do número 2 seja aplicada na entrada *A* e uma representação binária do número 5 seja aplicada na entrada *B*, (discutiremos a representação binária de números e símbolos no Capítulo 2). Um nível ALTO aparecerá na saída $A < B$ (*A* é menor que *B*), indicando a relação entre os dois números (2 é menor que 5). As setas maiores representam um grupo de linhas em paralelo pelas quais os bits são transferidos.

As Funções Aritméticas

Adição A adição é realizada por um circuito lógico denominado **somador**, abordado no Capítulo 6. Um somador soma dois números binários (nas entradas *A* e *B* com um carry na entrada C_{in}) e gera uma soma (Σ) e um carry (vai um) de saída (C_{out}), conforme mostra a Figura 1–20(a). A Figura 1–20(b) ilustra a soma de 3 com 9. Sabemos que a soma é 12; o somador indica esse resultado gerando um 2 na saída 'soma' e um 1 na saída de carry. Considere que a entrada de carry nesse exemplo seja 0.



(a) Somador básico (b) Exemplo: *A* mais *B* (3 + 9 = 12)

▲ **FIGURA 1–20**

A função de soma.

Subtração A subtração também é realizada por um circuito lógico. Um **subtrator** necessita de três entradas: duas para os números a serem subtraídos e uma para o borrow (empréstimo). As duas saídas são: a saída da diferença e a saída de borrow. Quando, por exemplo, 5 é subtraído de 8 sem borrow na entrada, a diferença é 3 sem borrow na saída. Veremos no Capítulo 2 como a subtração pode ser realizada por um somador porque a subtração é simplesmente um caso especial da adição.

Multiplicação A multiplicação é realizada por um circuito lógico denominado *multiplicador*. Os números são multiplicados sempre dois de cada vez, assim são necessárias duas entradas. A saída do multiplicador é o produto. Devido a multiplicação ser uma série de adições com deslocamentos nas posições dos produtos parciais, ela pode ser realizada usando um somador associado a outros circuitos.

Divisão A divisão pode ser realizada por meio de uma série de subtrações, comparações e deslocamentos, sendo que dessa forma ela pode ser feita usando um somador associado a outros circuitos. São necessárias duas entradas no divisor e as saídas geradas são o quociente e o resto.

NOTA: COMPUTAÇÃO

Em um microprocessador, a unidade lógica e aritmética (ALU – *arithmetic logic unit*) realiza as operações de soma, subtração, multiplicação e divisão bem como as operações lógicas sobre os dados digitais conforme determinado por uma série de instruções. Uma ALU típica é construída com várias centenas de portas lógicas.

A Função de Conversão de Código

Um **código** é um conjunto de bits organizados em um padrão único e usado para representar uma informação específica. Um conversor de código converte uma informação codificada de uma forma em uma outra forma de código. Como exemplos disso temos as conversões entre binário e outros códigos, como decimal codificado em binário (BCD – *binary coded decimal*) e código Gray. Vários tipos de códigos são abordados no Capítulo 2, e as conversões de código são abordadas no Capítulo 6.

A Função de Codificação

A função é realizada por um circuito lógico denominado **codificador**, abordado no Capítulo 6. Um codificador converte informação, tal como um número decimal ou um caractere do alfabeto, em alguma forma codificada. Por exemplo, um certo tipo de codificador converte cada um dos dígitos decimais, de 0 a 9, em um código binário. Um nível ALTO na entrada correspondente a um digito decimal gera níveis lógicos que representam o código binário apropriado nas linhas de saída.

A Figura 1–21 é uma ilustração simples de um codificador usado para converter (codificar) as teclas acionadas de uma calculadora em um código binário que pode ser processado pelos circuitos da calculadora.

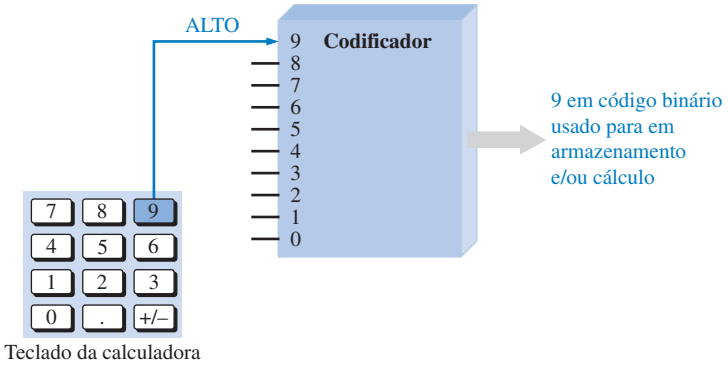


FIGURA 1–21
Um codificador usado para codificar as teclas acionadas de uma calculadora em um código binário sendo usado para armazenamento ou cálculo.

A Função de Decodificação

A função de decodificação é realizada por um circuito lógico denominado **decodificador**, abordado no Capítulo 6. Um decodificador converte uma informação codificada, como um número binário, numa forma não-codificada, como a forma de um número decimal. Por exemplo, um tipo particular de decodificador converte um código binário de 4 bits em um dígito decimal apropriado.

A Figura 1–22 é uma ilustração simples de um tipo de decodificador que é usado para ativar um display de 7 segmentos. Cada um dos sete segmentos do display é conectado a uma linha de saída do decodificador. Quando um determinado código binário aparece nas entradas do decodificador, as linhas de saída apropriadas são ativadas fazendo com que os segmentos apropriados sejam acesos mostrando o dígito decimal correspondente ao código binário.

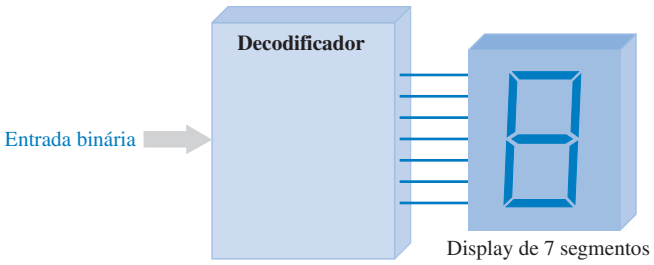


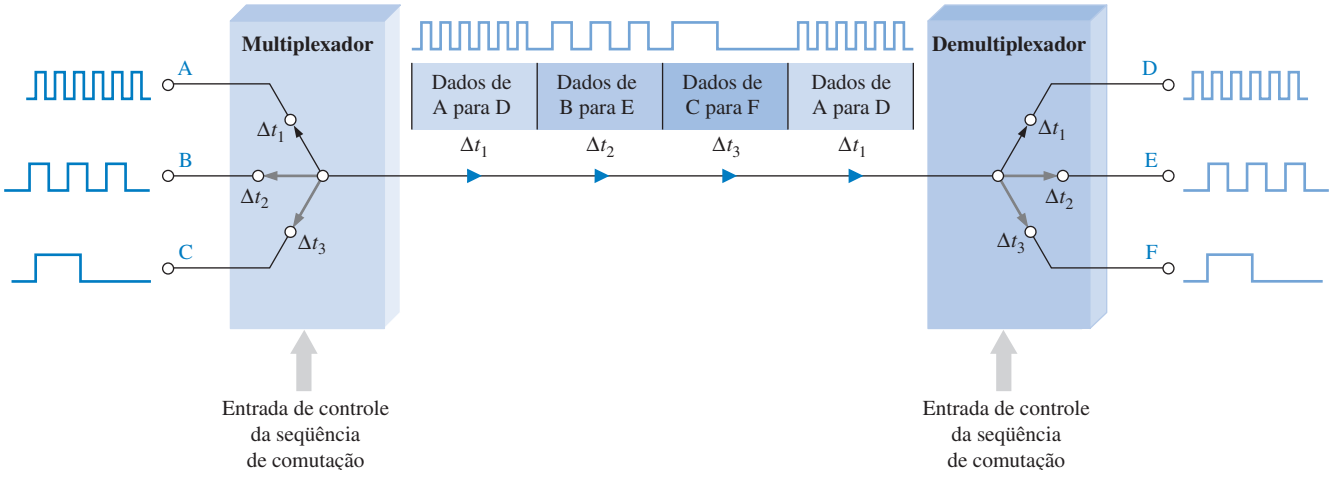
FIGURA 1–22
Um decodificador usado para converter um determinado código em um caractere decimal de 7 segmentos.

A Função de Seleção de Dados

Dois tipos de circuitos que selecionam dados são o multiplexador e o demultiplexador. Um **multiplexador**, ou apenas mux para abreviar, é um circuito lógico que comuta dados digitais a partir de diversas linhas de entrada em uma única linha de saída numa seqüência temporal especificada. Funcionalmente, um multiplexador pode ser representado pela operação de uma chave eletrônica que conecta seqüencialmente cada uma das linhas de entrada à linha de saída. Um **demultiplexador** (demux) é um circuito lógico que comuta dados digitais de uma linha de

entrada para diversas linhas de saída numa seqüência temporal especificada. Essencialmente, um demux faz o contrário do mux.

A multiplexação e a demultiplexação são usadas quando dados de fontes diversas são transmitidos ao longo de uma linha para um local distante e redistribuídos para diversos destinatários. A Figura 1–23 ilustra esse tipo de aplicação em que dados digitais a partir de três fontes são enviados ao longo de uma única linha para três destinatários num outro local.



▲ **FIGURA 1–23**

Ilustração de uma aplicação básica de multiplexação/demultiplexação.

Na Figura 1–23, os dados da entrada *A* são conectados à linha de saída durante o intervalo de tempo Δt_1 e transmitido para o demultiplexador que os conecta na saída *D*. Em seguida, durante o intervalo Δt_2, o multiplexador comuta para a entrada *B* e o demultiplexador comuta para a saída *E*. durante o intervalo de tempo Δt_3, o multiplexador comuta para a entrada *C* e o demultiplexador comuta para a saída *F*.

Para resumir, durante o primeiro intervalo de tempo, o dado na entrada *A* vai para a saída *D*. durante o segundo intervalo de tempo, o dado na entrada *B* vai para a saída *E*. durante o terceiro intervalo de tempo, o dado na entrada *C* vai para a saída *F*. Após isso, a seqüência se repete. Devido o tempo ser dividido entre as diversas fontes e destinatários, esse processo é denominado de *multiplexação por divisão do tempo* (TDM – *time division multiplexing*).

NOTA: COMPUTAÇÃO

A memória interna de um computador, RAM e ROM, bem como as memórias caches de capacidades menores são memórias semicondutoras. Os registradores em um microprocessador são construídos de flip-flops semicondutores. Os dispositivos de memória em disco são o disco rígido interno, o disquete e o CD-ROM.

A Função de Armazenamento

Armazenamento é uma função necessária na maioria dos sistemas digitais, sendo a sua finalidade guardar informação binária por um período de tempo. Alguns dispositivos de armazenamento são usados para armazenamento temporário e outros são usados para armazenamento por longo tempo. Um dispositivo de armazenamento pode "memorizar" um bit ou um grupo de bits e manter a informação pelo tempo necessário. Tipos comuns de dispositivos de armazenamento são: flip-flops, registradores, memórias semicondutoras, discos magnéticos, fitas magnéticas e discos ópticos (CDs).

Flip-flops Um **flip-flop** é um circuito lógico biestável (dois estados estáveis) que pode armazenar apenas um bit de cada vez, podendo ser 1 ou 0. A saída de um flip-flop indica qual bit está armazenado. Um nível ALTO na saída indica que um 1 está armazenado e um nível BAIXO na saída indica que um 0 está armazenado. Flip-flops são implementados com portas lógicas e serão abordados no Capítulo 7.

Registradores Um **registrador** é formado pela combinação de vários flip-flops de forma que um grupo de bits possa ser armazenado. Por exemplo, um registrador de 8 bits é construído a partir de oito flip-flops. Além de armazenar bits, registradores podem ser usados para deslocar os bits a partir de uma posição para outra dentro do registrador ou para fora (para um outro circuito); portanto, esses dispositivos são conhecidos com *registradores de deslocamento*. Registradores de deslocamento serão abordados no Capítulo 9.

Os dois tipos básicos de registradores de deslocamento são: serial e paralelo. Os bits são armazenados um de cada vez em um registrador de deslocamento serial, conforme ilustrado na Figura 1–24. Uma boa analogia para o registrador de deslocamento serial é a entrada de passageiros em fila única através da porta de um ônibus. Eles também saem do ônibus em fila única.

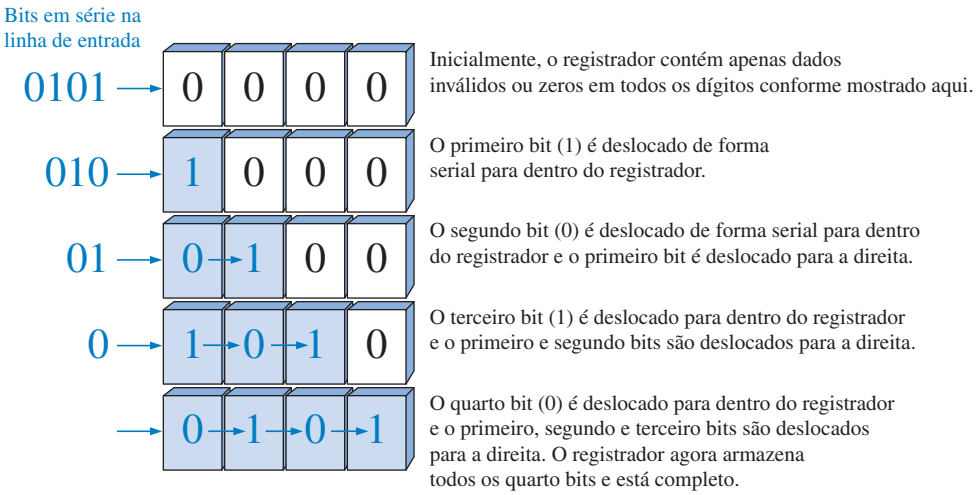


FIGURA 1–24

Exemplo da operação de um registrador de deslocamento serial de 4 bits. Cada bloco representa uma "célula" ou flip-flop de armazenamento.

Os bits são armazenados em um registrador paralelo simultaneamente a partir de linhas em paralelo, conforme mostra a Figura 1–25. Nesse caso, uma boa analogia é a entrada de passageiros em uma montanha russa onde todos eles entram nos carros em paralelo.

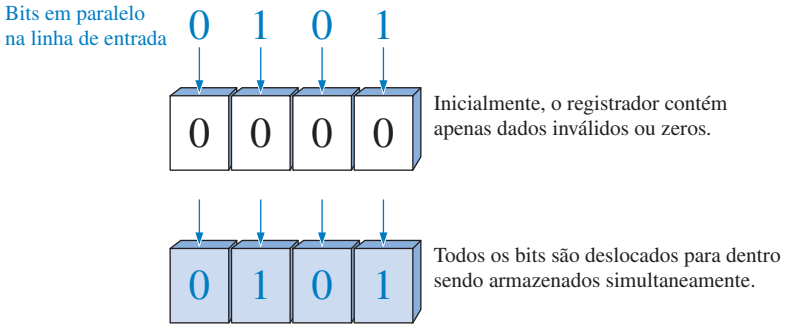


FIGURA 1–25

Exemplo da operação de um registrador paralelo de 4 bits.

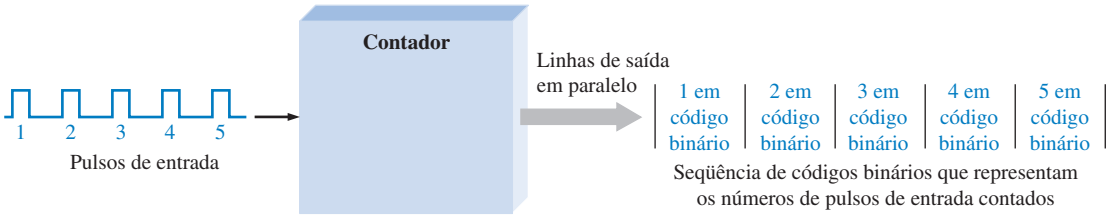
Memórias Semicondutoras As memórias semicondutoras são dispositivos usados tipicamente para armazenagem de uma grande quantidade de bits. Em um tipo de memória, denominado *memória apenas de leitura* ou ROM (*Read-Only Memory*), os dados em binário são armazenados de forma permanente ou semipermanente não podendo ser alterados prontamente. Na *memória de acesso aleatório* ou RAM (*Random-Access Memory*), os dados em binário são armazenados temporariamente e podem ser alterados facilmente. Esse assunto é abordado no Capítulo 10.

Memórias Magnéticas As memórias de discos magnéticos são usadas no armazenamento de massa de dados em binário. Como exemplos temos os disquetes usados em computadores bem como os discos rígidos (HDs – *hard disks*) internos. Fitas magnéticas ainda são usadas em aplicações de memorização e para back up de dados a partir de outros dispositivos de armazenamento.

A Função de Contagem

A função de contagem é importante em sistemas digitais. Existem muitos tipos de **contadores** digitais, mas a finalidade básica deles é contar eventos representados por transições de níveis ou pulsos. Para contar, o contador tem que "lembrar" do número atual para poder passar

para o próximo número da seqüência. Portanto, a capacidade de armazenamento é uma importante característica de todos os contadores, sendo que os flip-flops são geralmente usados para implementá-los. A Figura 1–26 ilustra a idéia básica da operação de um contador. Os contadores são abordados no Capítulo 8.



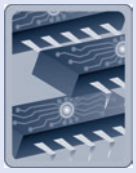
▲ **FIGURA 1–26**

Ilustração da operação básica de um contador.

SEÇÃO 1–4 REVISÃO

1. O que faz um comparador?
2. Quais são as quatro operações aritméticas básicas?
3. Descreva a codificação e cite um exemplo.
4. Descreva a decodificação e cite um exemplo.
5. Explique a finalidade básica da multiplexação e da demultiplexação.
6. Cite quatro tipos de dispositivos de armazenamento.
7. O que faz um contador?

1-5 CIRCUITOS INTEGRADOS DE FUNÇÕES FIXAS



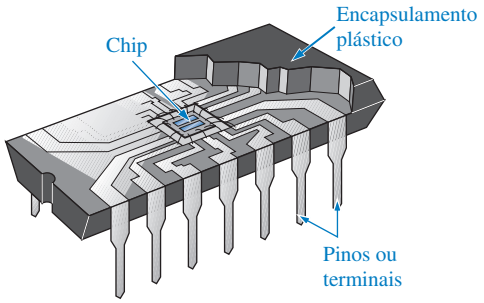
Todos os elementos e funções lógicas que foram discutidos estão geralmente disponíveis na forma de circuitos integrados (CIs). Sistemas digitais têm sido por muitos anos incorporados em CIs por causa do tamanho, alta confiabilidade, baixo custo e baixo consumo que os CIs apresentam. É importante sermos capazes de reconhecer os encapsulamentos de CIs e saber como os pinos são numerados, bem como nos familiarizarmos com a forma na qual a complexidade e as tecnologias dos circuitos determinam as diversas classificações dos CIs.

Ao final do estudo desta seção você deverá ser capaz de:

■ Reconhecer a diferença entre dispositivos PTH e SMD ■ Identificar encapsulamentos DIP ■ Identificar encapsulamentos SOIC ■ Identificar encapsulamentos PLCC ■ Identificar encapsulamentos LCCC ■ Determine a numeração de pinos nos diversos tipos de encapsulamentos de CIs ■ Explicar a classificação em termos da complexidade dos CIs de funções fixas.

Um **circuito integrado (CI)** monolítico é um circuito eletrônico construído totalmente em um único e pequeno chip de silício. Todos os componentes que formam o circuito (transistores, diodos, resistores e capacitores) são partes integrais de um único chip. Os CIs digitais podem ser divididos em duas grandes categorias: funções lógicas fixas e funções lógicas programáveis. No caso dos dispositivos de funções lógicas fixas, as funções são estabelecidas pelo fabricante e não podem ser alteradas.

A Figura 1–27 mostra uma vista em corte de um tipo de CI de função fixa com o chip do circuito que fica dentro do encapsulamento. Os pontos de conexão do chip são interligados aos pinos no encapsulamento para permitir as conexões de entrada e saída com o mundo externo.

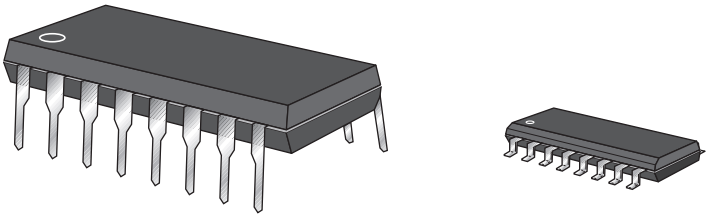


▶ **FIGURA 1–27**

Vista em corte de um tipo de encapsulamento de CI de função fixa mostrando o chip interno com as conexões aos pinos de entrada e saída.

Encapsulamento de CIs

Os encapsulamentos de circuitos integrados são classificados de acordo com a forma com que eles são montados nas placas de circuito impresso como dispositivos com pinos que passam através de furos (PTH – *pin through-hole*) na placa ou como dispositivos montados na superfície (SMD – *surface mounted-device*) da placa. O tipo PTH tem pinos (terminais) que são inseridos em furos na placa de circuito impresso e podem ser soldados a condutores (trilhas na placa) no lado oposto da placa. O tipo mais comum de PTH é o encapsulamento no qual os pinos estão dispostos em duas linhas paralelas (**DIP** – *dual in-line package*) mostrado na Figura 1–28(a).



(a) DIP (b) SOIC

▶ **FIGURA 1–28**

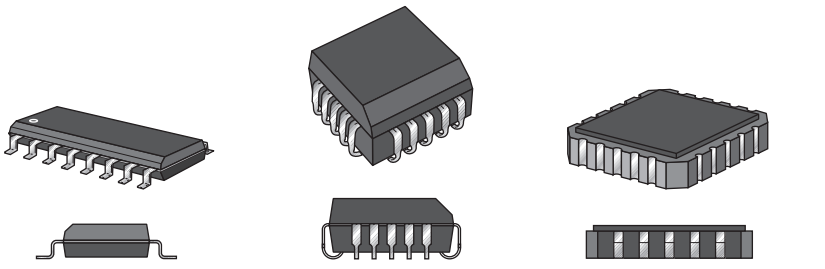
Exemplos de dispositivos PTH e SMD. O DIP é bem maior que o SOIC com o mesmo número de terminais. Esse DIP em particular tem aproximadamente 0,785 polegadas (~2 cm) de comprimento e este SOIC tem aproximadamente 0,385 polegadas (~0,98 cm) de comprimento.

Um outro tipo de encapsulamento de CI usa tecnologia de montagem em superfície (**SMT** – *surface mount tecnology*). A montagem em superfície é uma alternativa de economia de espaço em comparação aos dispositivos PTH. Os furos através da placa de circuito impresso não são necessários para SMT. Os pinos nos encapsulamentos de SMDs são soldados diretamente às trilhas na placa, deixando o outro lado livre para circuitos adicionais. Além disso, para um circuito com o mesmo número de pinos, o encapsulamento de um SMD é muito menor que um DIP porque os pinos estão mais próximos um do outro. Um exemplo de encapsulamento SMT para um circuito integrado de perfil baixo (SOIC – *small-outline integrated circuit*) é mostrado na Figura 1–28(b).

Três tipos comuns de encapsulamentos SMT são: **SOIC** (*small outline IC*), **PLCC** (*plastic lead chip carrier*) e **LCCC** (*leadless ceramic chip carrier*). Esses tipos de encapsulamentos estão disponíveis em vários tamanhos dependendo do número de terminais (mais terminais são necessários para circuitos mais complexos). A Figura 1–29 mostra exemplos de cada um desses tipos. Como podemos ver, os terminais do SOIC são construídos na forma de "asa de gaivota". Os terminais do PLCC são dobrados para baixo do encapsulamento dando a forma da letra J. em vez de terminais, o LCCC tem contatos metálicos moldados no seu corpo cerâmico. Outras variações de encapsulamentos SMT incluem **SSOP** (*shrink small-outline package*), **TSSOP** (*thin shrink small-outline package*) e **TVSOP** (*thin very small-outline package*).

Numeração dos Pinos

Todos os encapsulamentos de CIs têm um formato padrão para a numeração dos pinos (terminais). Os DIPs e os SOICs têm o estilo de numeração ilustrado na Figura 1–30(a) para um encapsulamento de 16 pinos. Observando a parte superior do encapsulamento, o pino 1 está indicado por um



(a) SOIC com terminais "asa de gaivota"

(b) PLCC com terminais tipo J

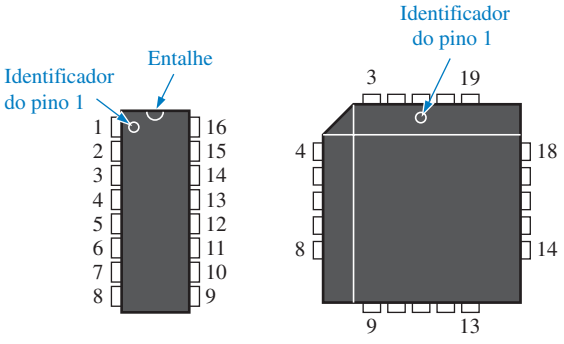
(c) LCCC sem terminais (os pontos de contato fazem parte do invólucro)

◀ **FIGURA 1–29**

Exemplos de configurações de encapsulamentos SMT.

identificador que pode ser um pequeno ponto, um entalhe ou uma borda chanfrada. O ponto está sempre próximo ao pino 1. Além disso, com o entalhe orientado para cima, o pino 1 é sempre o pino superior esquerdo, conforme indicado. Começando pelo pino 1, os números dos pinos aumentam à medida que se percorre os pinos para baixo, passando para o outro lado e subindo. O pino de maior número está sempre à direita do entalhe ou do lado oposto ao ponto.

Os encapsulamentos PLCC e LCCC têm terminais dispostos nos quatro lados. O pino 1 é indicado por um ponto ou uma marca de índice e está localizado no centro de um dos lados. Os números dos pinos aumentam no sentido anti-horário quando se visualiza o encapsulamento por cima. O pino de maior número está sempre à direita do pino 1. A Figura 1–30(b) ilustra esse formato para um encapsulamento PLCC de 20 pinos.



(a) DIP ou SOIC

(b) PLCC ou LCCC

◀ **FIGURA 1–30**

Numeração dos pinos para dois tipos padronizados de encapsulamentos de CI. Na figura são mostradas as vistas superiores.

Classificações de Complexidade para CIs de Funções Fixas

Os CIs digitais de funções fixas são classificados de acordo com suas complexidades. Eles são apresentados aqui a partir da função menos complexa até a mais complexa. As figuras de complexidade especificadas aqui por SSI, MSI, LSI, VLSI e ULSI são geralmente aceitas, porém podemos encontrar variações nas definições de uma fonte para outra.

- **Integração em pequena escala** (**SSI** – *small-scale integration*) descreve os CIs de função fixa que têm até dez circuitos de portas equivalentes em um único chip e eles incluem portas básicas e flip-flops.
- **Integração em escala média** (**MSI** – *medium-scale integration*) descreve os circuitos integrados que têm de 10 a 100 portas equivalentes num chip. Eles incluem funções lógicas tais como codificadores, decodificadores, contadores, registradores, multiplexadores, circuitos aritméticos, memórias de pequena capacidade entre outros circuitos.
- **Integração em escala ampla** (**LSI** – *large-scale integration*) é uma classificação de CIs com complexidades de 100 a 10.000 portas equivalentes por chip, que incluem memórias.
- **Integração em escala muito ampla** (**VLSI** – *very large-scale integration*) descreve circuitos integrados com complexidades de 10.000 a 100.000 portas equivalentes por chip.

- **Integração em escala ultra ampla** (**ULSI** – *ultra large-scale integration*) descreve memórias e microprocessadores de grandes capacidades e computadores de pastilha única. Complexidades maiores que 100.000 portas equivalentes por chip são classificados como ULSI.

Tecnologia de Circuitos Integrados

Os tipos de transistores com os quais os circuitos integrados são implementados são MOSFETs (*metal-oxide semiconductor field-effect transistors*) ou transistores de junção bipolar (BJT – *bipolar-junction transistor*). Uma tecnologia de circuitos que usa MOSFETs é a CMOS (*complementary metal-oxide semiconductor*). Um tipo de tecnologia de circuito digital de função fixa que usa BJTs é a TTL (*transistor-transistor logic*). A BiCMOS usa a combinação de CMOS e TTL.

Todas as portas e as outras funções podem ser implementadas com qualquer tipo e tecnologia de circuito. Os circuitos SSI e MSI estão geralmente disponíveis em CMOS e TTL. Os circuitos LSI, VLSI e ULSI são implementados geralmente com CMOS ou NMOS porque necessitam de menor área no chip e menor consumo de potência. Existe mais informações sobre tecnologias de circuitos integrados no Capítulo 3. Além disso, o Capítulo 14 fornece uma abordagem completa sobre algumas tecnologias de circuitos integrados digitais.

Precauções no Manuseio de Dispositivos CMOS Devido à sua estrutura particular, os dispositivos CMOS são muito sensíveis à carga estática e podem ser danificados por descarga eletrostática (ESD – *electrostatic discharge*) se não for manuseado adequadamente. As precauções a seguir devem ser tomadas quando se trabalha com dispositivos CMOS:

- Dispositivos CMOS devem ser transportados e armazenados em espuma condutiva.
- Todos os instrumentos e bancadas metálicas usados em testes devem ser aterrados.
- O pulso da pessoa que manuseia o dispositivo deve ser aterrada através um fio com um resistor de alto valor.
- Não remova um dispositivo CMOS (ou qualquer outro dispositivo no que diz respeito a esse assunto) de um circuito enquanto este estiver energizado.
- Não conecte sinais de tensão contínua ou alternada em um dispositivo CMOS enquanto a fonte estiver desligada.

SEÇÃO 1–5 REVISÃO

1. O que é um circuito integrado?
2. Defina os termos DIP, SMT, SOIC, SSI, MSI, LSI, VLSI e ULSI.
3. De uma forma geral, qual é a classificação de um CI de função fixa que apresenta os seguintes números de portas equivalentes?
   (a) 10 **(b)** 75 **(c)** 500 **(d)** 15.000 **(e)** 200.000

1-6 INTRODUÇÃO À LÓGICA PROGRAMÁVEL

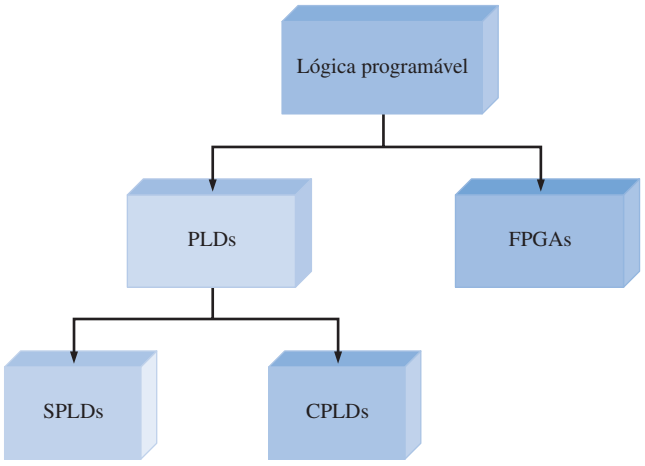
A lógica de programação necessita de hardware e software. Dispositivos de lógica programável podem ser programados para realizar funções lógicas especificadas pelo fabricante ou pelo usuário. Uma vantagem da lógica programável sobre a lógica de funções fixas é que os dispositivos programáveis ocupam bem menos espaço na placa para uma mesma quantidade de lógica. Uma outra vantagem é que, com a lógica programável, pode-se alterar os projetos com facilidade sem alterações físicas no hardware ou substituição de componentes. Além disso, um projeto lógico geralmente pode ser implementado mais rápido e com um menor custo com a lógica programável do que com CIs de função fixa.

Ao final do estudo desta seção você deverá ser capaz de:

■ Enunciar os principais tipos de lógica programável e discutir as diferenças entre eles ■ Discutir os métodos de programação ■ Fazer uma lista das principais linguagens de programação usadas em lógica programável ■ Discutir os processos de projetos com lógica programável

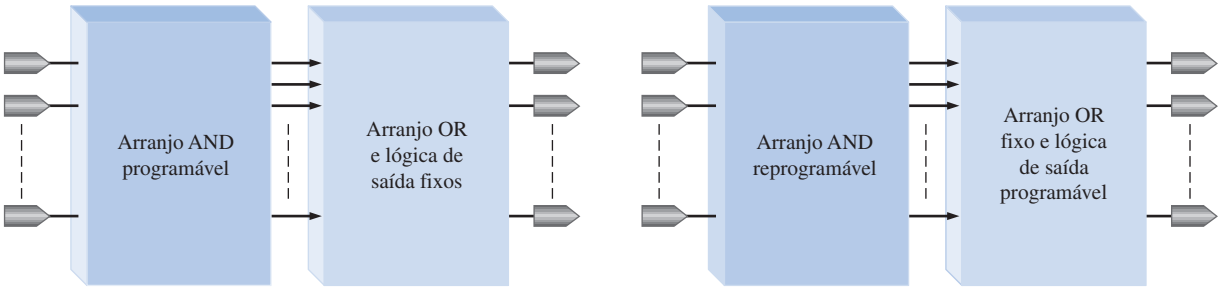
Tipos de Dispositivos Lógicos Programáveis

Temos disponíveis vários tipos de lógica programável, desde dispositivos de pequena capacidade, os quais podem substituir alguns dispositivos de funções lógicas fixas, até dispositivos complexos de alta densidade, que podem substituir centenas de dispositivos de funções fixas. As duas categorias principais de lógica programável pelo usuário são **PLD** (*programmable logic device*) e **FPGA** (*field programmable gate array*), conforme indicado na Figura 1–31. As PLDs se subdividem SPLDs (PLDs simples) e CPLDs (PLDs complexas).



◀ **FIGURA 1–31**
Lógica programável.

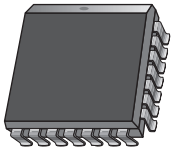
Dispositivo de Lógica Programável Simples (SPLD) O SPLD foi o PLD que surgiu inicialmente e ainda encontra-se disponível para aplicações de pequena escala. Geralmente, um **SPLD** pode substituir até dez CIs de função fixa e suas interconexões, dependendo dos tipos de funções e do SPLD específico. A maioria dos SPLDs estão em uma das duas categorias: PAL e GAL. Um dispositivo **PAL** (*programmable array of logic*) pode ser programado uma vez. Ele consiste de um arranjo programável de portas AND e um arranjo fixo de portas OR, como mostra a Figura 1–32(a). Um dispositivo **GAL** (*generic array logic*) é basicamente um dispositivo PAL que pode ser reprogramado várias vezes. Ele consiste de um arranjo programável de portas AND e um arranjo fixo de portas OR com saídas programáveis, como mostra



(a) PAL (b) GAL

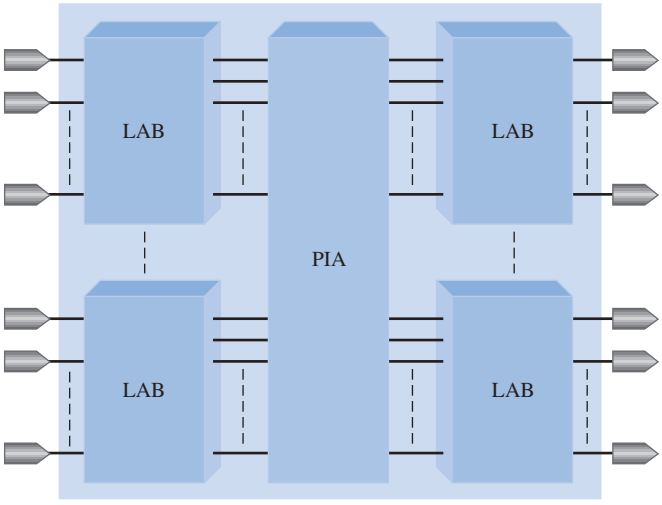
▲ **FIGURA 1–32**
Diagramas em bloco de dispositivos lógicos programáveis simples (SPLDs).

a Figura 1–32(b). Um típico encapsulamento para SPLD é mostrado na Figura 1–33 e geralmente tem de 24 a 28 pinos.



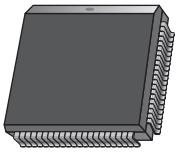
▶ **FIGURA 1–33**
Encapsulamento típico de um SPLD.

Dispositivo de Lógica Programável Complexo (CPLD) Conforme o progresso tecnológico e o aumento da quantidade de circuito que pode ser implementado em um chip (densidade de chip), os fabricantes conseguiram implementar mais de um SPLD num único chip, surgindo assim o CPLD. Essencialmente, o **CPLD** é um dispositivo que contém múltiplos SPLDs e pode substituir diversos CIs de funções fixas. A Figura 1–34 mostra o diagrama em bloco básico de um CPLD com quatro blocos de arranjo lógico (LABs – *logic array blocks*) e um arranjo de interconexões programáveis (PIA – *programmable interconnction array*). Dependendo do CPLD específico, podem existir de 2 a 64 LABs. Cada bloco de arranjo lógico é aproximadamente equivalente a um SPLD.

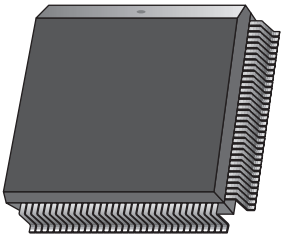


▶ **FIGURA 1–34**
Diagrama em bloco geral de um CPLD.

Geralmente, CPLDs podem ser usadas para implementar quaisquer tipos de funções lógicas discutidas anteriormente, como decodificadores, codificadores, multiplexadores, demultiplexadores e somadores. Eles estão disponíveis em uma variedade de configurações, tendo tipicamente um encapsulamento com uma extensão de 44 a 160 pinos. A Figura 1–35 mostra exemplos de encapsulamentos para CPLDs.



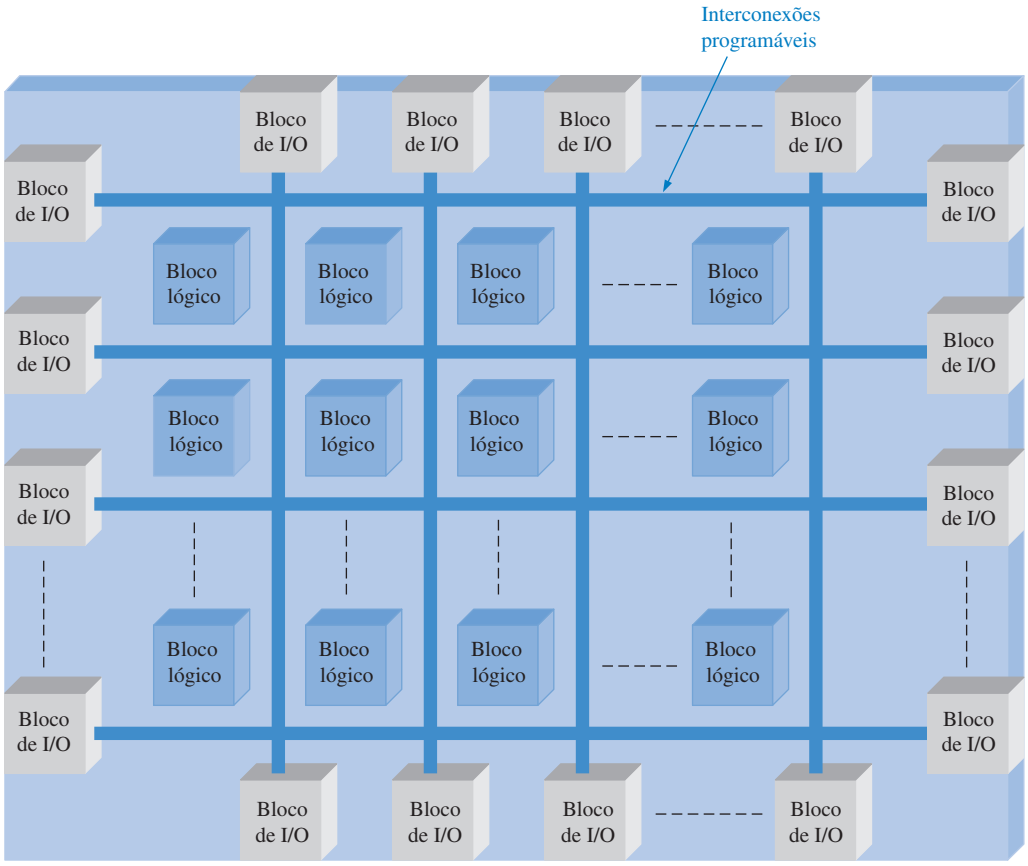
(a) Encapsulamento PLCC de 84 pinos



(b) Encapsulamento PQFP de 128 pinos

▶ **FIGURA 1–35**
Encapsulamentos típicos de CPLDs.

Arranjo de Portas Programáveis por Campo (FPGA) Um **FPGA** é geralmente mais complexo e tem uma densidade muito maior que CPLD, embora suas aplicações possam, em alguns casos, se sobrepor. Conforme mencionado, o SPLD e o CPLD são inter-relacionados porque o

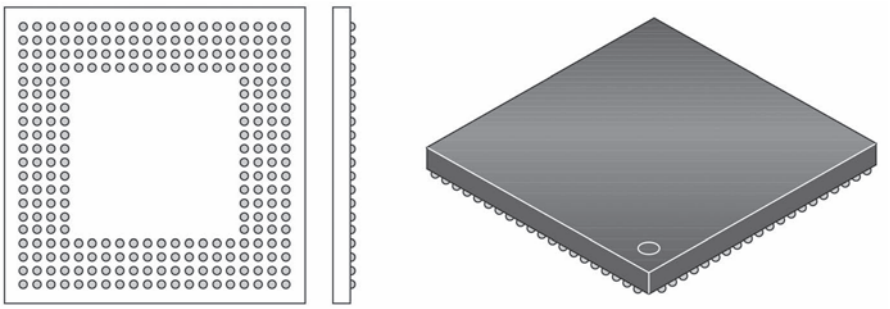


▲ **FIGURA 1–36**

Estrutura básica de um FPGA.

CPLD contém basicamente dispositivos SPLDs. Entretanto, o FPGA tem uma estrutura (arquitetura) interna diferente, conforme ilustra a Figura 1–36. Os três elementos básicos de um FPGA são o bloco lógico, as interconexões programáveis e os blocos de entrada/saída (I/O – *in/out*).

Os blocos lógicos em um FPGA não são complexos como os LABs em um CPLD, porém, geralmente eles existem em maior número. Quando os blocos lógicos são relativamente simples, a arquitetura do FPGA é denominada *grão fino* (*fine-graned*). Quando os blocos lógicos são maiores e mais complexos, a arquitetura é denominada *grão grosso* (*coarse-graned*). Os blocos de I/O estão nas bordas da estrutura e proporcionam acesso individualmente selecionável de entrada, saída ou bidirecional ao mundo externo. A matriz de interconexões programáveis distribuídas provê as interconexões de blocos lógicos e as conexões para as entradas e saídas. FPGAs de grande capacidade podem ter dezenas de centenas de blocos lógicos, além de memória e outros recursos. Um encapsulamento típico BGA (*ball-grid array*) de uma FPGA é mostrado na Figura 1–37. Esses tipos de encapsulamentos podem ter mais de 1.000 pinos de entradas e saídas.

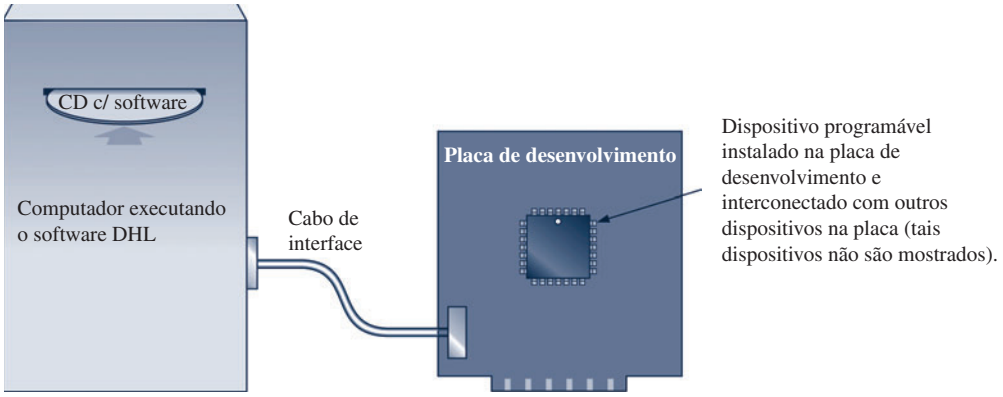


◀ **FIGURA 1–37**

Uma configuração típica de encapsulamento BGA (*boll-grid array*).

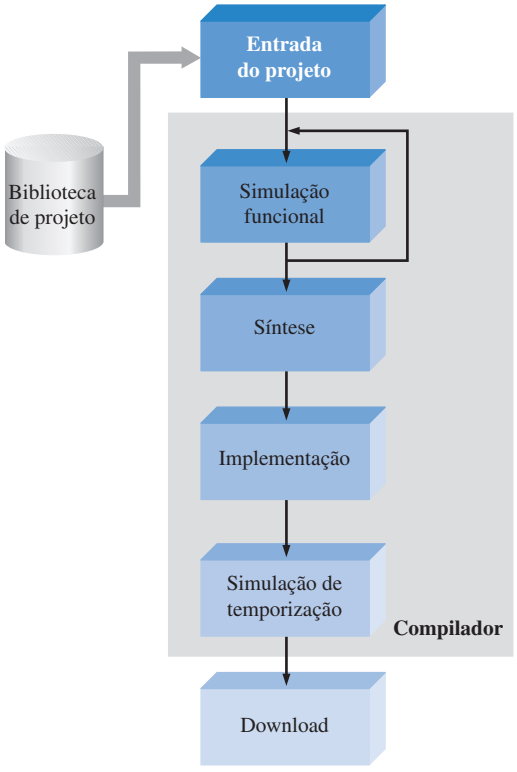
O Processo de Programação

Uma SPLD, CPLD ou FPGA pode ser vista como um "quadro branco" no qual podemos implementar um determinado circuito ou sistema projetado fazendo uso de um certo processo. Esse processo necessita de um software de desenvolvimento instalado em um computador para implementar um projeto de circuito em um chip programável. O computador tem que ser interfaceado por meio de uma placa de desenvolvimento ou um equipamento de programação contendo o dispositivo, conforme ilustrado na Figura 1–38.



▶ **FIGURA 1–38**
Configuração básica de um sistema para programação de PLD ou FPGA.

Vários passos, chamados de *fluxo de projeto*, estão envolvidos no processo de implementação de um projeto lógico em um dispositivo lógico programável. A Figura 1–39 mostra um diagrama em bloco de um processo de programação. Conforme indicado, o fluxo do projeto tem acesso a uma biblioteca de projeto.



▶ **FIGURA 1–39**
Diagrama em bloco básico da seqüência do projeto de uma lógica programável.

Entrada do Projeto Esse é o primeiro passo. O projeto do circuito ou sistema tem que ser inserido no software de desenvolvimento de projeto usando entrada baseada em texto, entrada gráfica

(desenho esquemático) ou descrição em diagrama de estado. A inserção do projeto é independente do dispositivo. A inserção baseada em texto é realizada com uma linguagem de descrição de hardware (HDL – *hardware description language*), tal como VHDL, Verilog, AHDL ou ABEL. A inserção gráfica (esquemático) permite que funções lógicas pré-armazenadas sejam selecionadas a partir de um biblioteca, apresentada na tela e então interconectadas para criar o projeto lógico. A inserção por diagrama de estados requer a especificação dos estados pelos quais um circuito lógico seqüencial passa e as condições que provocam a mudança de cada estado.

Uma vez inserido o projeto, ele é compilado. Um **compilador** é um programa que controla o processo do fluxo do projeto e traduz o código-fonte em código-objeto num formato que pode ser testado logicamente ou transferido para um dispositivo destino. O código-fonte é criado durante a inserção do projeto e o código-objeto é o código final que realmente faz com que o projeto seja implementado no dispositivo programável.

Simulação Funcional O projeto compilado inserido é simulado por software para confirmar se os circuitos lógicos funcionam conforme esperado. A simulação irá verificar se as saídas adequadas são geradas para um conjunto de entradas especificadas. Uma ferramenta de software independente do dispositivo que faz essa tarefa é denominada *editor de forma de onda*. Qualquer falha demonstrada pela simulação poderia ser corrigida voltando no projeto inserido e realizando as alterações apropriadas.

Síntese A fase **síntese** é onde o projeto é traduzido em uma lista (*netlist*), a qual tem uma forma padronizada e é independente do dispositivo.

Implementação A **implementação** é onde a estrutura lógica descrita pela netlist é mapeada na estrutura real do dispositivo a ser programado. O processo de implementação é denominado *fitting* ou *place and route* e resulta em uma saída denominada seqüência de bits, a qual depende do dispositivo usado.

Simulação de Temporização Esse passo ocorre após o projeto ser mapeado no dispositivo especificado. A simulação de temporização é basicamente usada para confirmar que não existem falhas no projeto ou problemas de temporização em função dos atrasos de propagação.

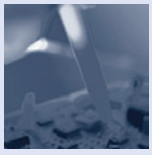
Download Uma vez gerada uma seqüência de bits para um dispositivo programável específico, ele tem que ser transferido (operação *download*) ao dispositivo para implementar o projeto de software no hardware. Alguns dispositivos programáveis são instalados em uma seção especial de um equipamento denominado *programador de dispositivos* ou em uma placa de desenvolvimento. Outros tipos de dispositivos podem ser programados quando ainda estão inseridos no sistema – denominados programáveis no sistema (ISP – *in-system programming*) – usando uma interface padrão JTAG (Joit Test Action Group). Alguns dispositivos são voláteis, o que significa que eles perdem o conteúdo armazenado quando sofrem uma operação de resete (inicialização) ou quando a alimentação é desligada. Nesse caso, os dados (seqüência de bits) têm que ser armazenados numa memória e recarregados no dispositivo após cada resete ou desligamento da alimentação. Além disso, o conteúdo de um dispositivo ISP pode ser manipulado ou atualizado enquanto estiver operando no sistema.

SEÇÃO 1–6 REVISÃO

1. Enuncie as principais categorias de dispositivos lógicos programáveis e especifique os seus acrônimos.
2. Em que um CPLD difere de um SPLD?
3. Cite os passos de um processo de programação de um dispositivo lógico programável.
4. Faça uma breve explicação de cada um dos passos citados na questão 3.

1-7 INSTRUMENTOS DE MEDIÇÃO E TESTE

A **análise de defeito** é o processo sistemático de isolamento, identificação e correção de defeitos em um circuito ou sistema. Uma variedade de instrumentos está disponível para ser usada em testes e análises de defeito. Alguns tipos comuns de instrumentos são apresentados e discutidos nesta seção.



Ao final do estudo desta seção você deverá ser capaz de:

■ Distinguir entre um osciloscópio analógico e um digital ■ Reconhecer os controles comuns de um osciloscópio ■ Determinar, usando um osciloscópio, a amplitude, o período, a freqüência e o ciclo de trabalho da forma de onda de um pulso ■ Discutir o analisador lógico e alguns formatos comuns ■ Descrever a finalidade de uma fonte de alimentação cc, de um gerador de funções e um multímetro digital (DMM)

O Osciloscópio

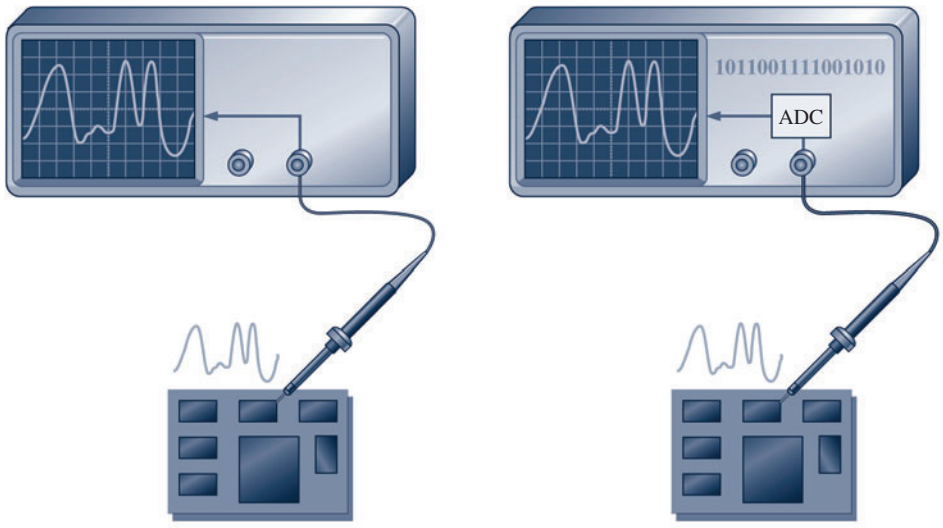
O osciloscópio é um dos instrumentos mais usados em teste e análise de defeito em geral. O osciloscópio é basicamente um dispositivo de apresentação gráfica que traça na tela o gráfico de um sinal elétrico. Na maioria das aplicações, o gráfico mostra como o sinal varia no tempo. O eixo vertical da tela do display representa a tensão e o eixo horizontal representa o tempo. A amplitude, o período e a freqüência de um sinal podem ser medidos usando um osciloscópio. Além disso, a largura do pulso, o ciclo de trabalho, o tempo de subida e o tempo de descida da forma de onda de um pulso podem ser determinados. A maioria dos osciloscópios pode mostrar pelo menos dois sinais na tela ao mesmo tempo, possibilitando que a relação de tempo entre os sinais possa ser observada. A Figura 1–40 mostra a imagem de um osciloscópio típico.

► **FIGURA 1–40**

Um osciloscópio de dois canais (imagem apresentada com a permissão da Tektronix, Inc).

Dois tipos básicos de osciloscópios, o analógico e o digital, podem ser usados para visualizar formas de onda digitais. Assim como mostra a Figura 1–41(b), o osciloscópio digital converte a forma de onda medida em informação digital por meio de um processo de amostragem em um conversor analógico-digital (ADC – *analog-digital converter*). A informação digital é então usada para reconstruir a forma de onda na tela.

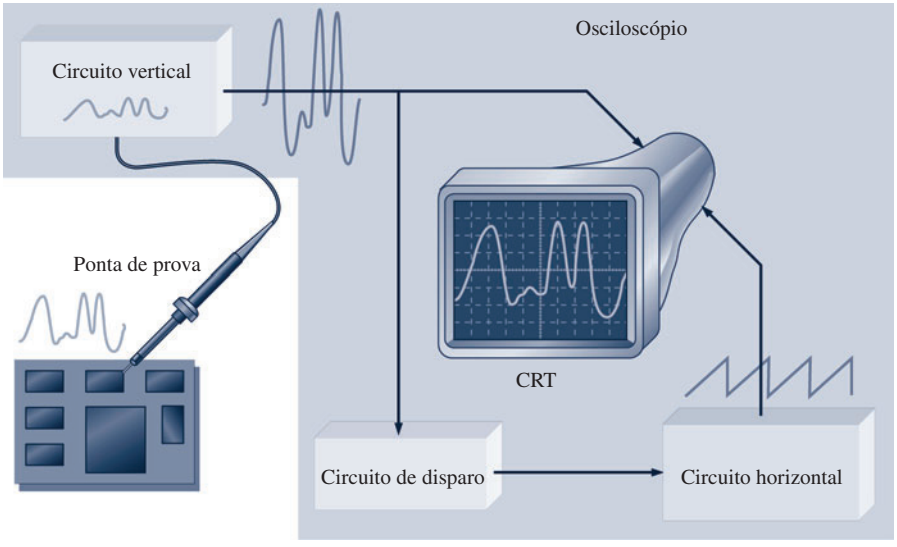
O osciloscópio digital é bem mais usado que o osciloscópio analógico. Entretanto, os dois tipos podem ser usados em diversas aplicações, sendo que cada um tem características que o torna mais adequado para determinadas situações. Um osciloscópio analógico mostra formas de onda "em tempo real" (á medida que elas acontecem). Osciloscópios digitais são úteis na medição de pulsos transientes que podem ocorrer aleatoriamente ou apenas uma vez. Além disso, pelo fato da informação relativa à forma de onda medida poder ser armazenada em um osciloscópio digital, ela pode ser visualizada algum tempo depois, pode ser impressa ou completamente analisada por um computador ou outros meios.



(a) Analógico (b) Digital

◀ **FIGURA 1–41**
Comparação entre os osciloscópios analógico e digital.

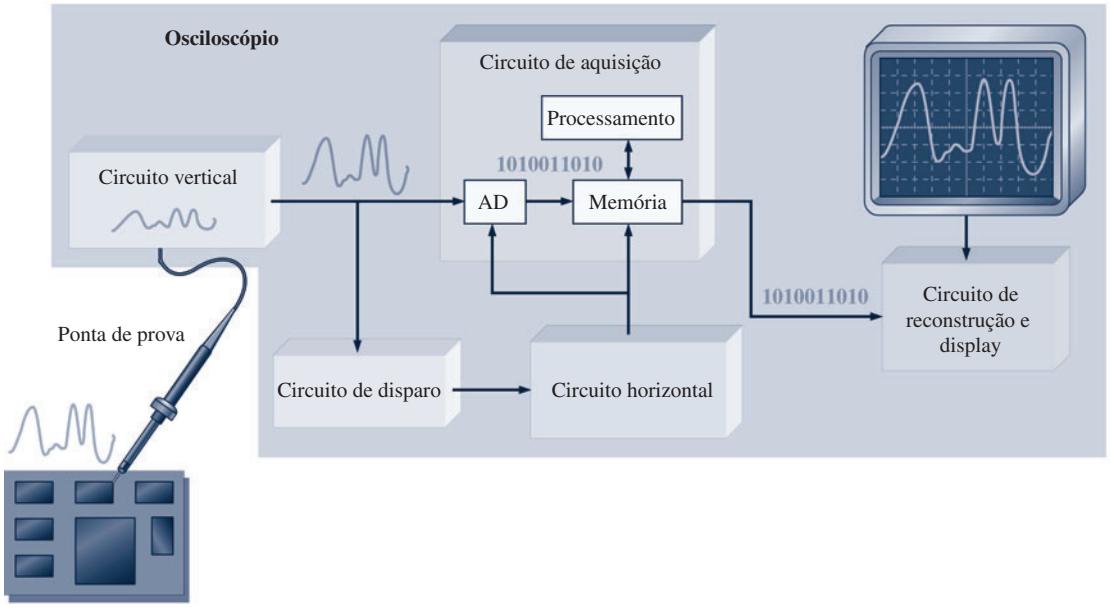
Operação Básica de um Osciloscópio Analógico Para medir uma tensão, uma **ponta de prova** tem que ser conectada entre o osciloscópio e o ponto do circuito no qual a tensão será medida. Geralmente, uma ponta de prova ×10 é usada para reduzir (atenuar) a amplitude do sinal por 10. O sinal, que passa pela ponta de prova, chega ao circuito vertical onde ele é atenuado ainda mais ou amplificado, dependendo da amplitude real e da escala ajustada no controle vertical do osciloscópio. O circuito vertical atua então nas placas de deflexão vertical do tubo de raios catódicos (CRT – *cathode Ray tube*). Além disso, o sinal passa pelo circuito de disparo que sincroniza o circuito horizontal para iniciar a varredura horizontal repetitiva do feixe de elétrons que percorre a tela usando uma forma de onda dente de serra. São realizadas diversas varreduras por segundo de forma que o feixe de elétrons tenha uma aparência de uma linha contínua na tela acompanhando o contorno da forma de onda. Essa operação básica é ilustrada na Figura 1–42.



◀ **FIGURA 1–42**
Diagrama em bloco de um osciloscópio analógico.

Operação Básica de um Osciloscópio Digital Algumas partes de um osciloscópio digital são similares às de um osciloscópio analógico. Entretanto, o osciloscópio digital é mais complexo, do que um analógico, tendo geralmente um display de cristal líquido (LCD – *liquid cristal display*) no lugar de um CRT. Em vez de mostrar uma forma de onda à medida que ela acontece, o osciloscópio digital

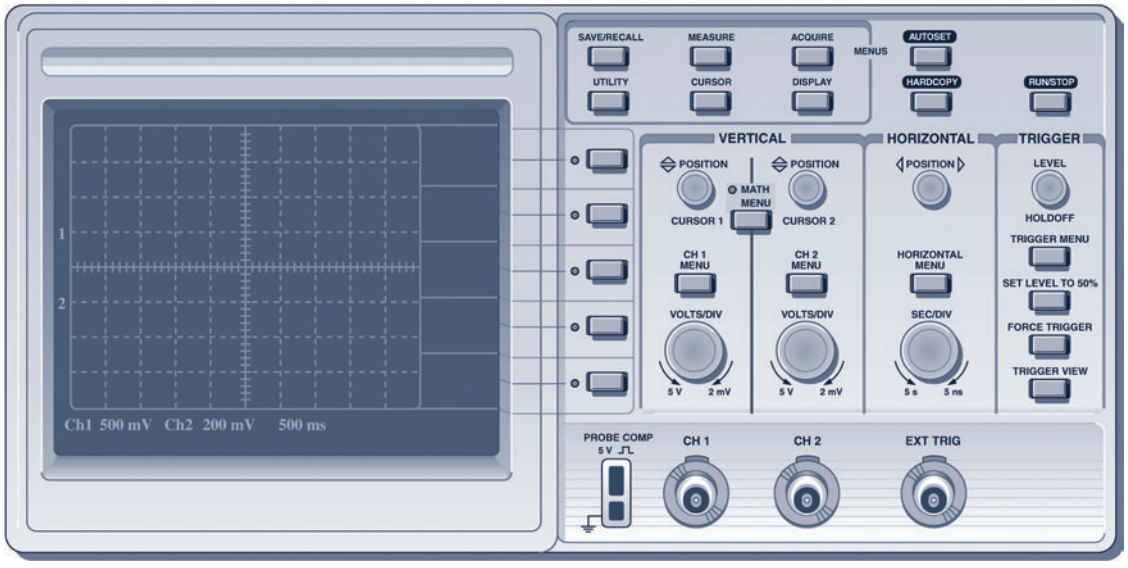
primeiro faz a aquisição da forma de onda analógica medida e a converte em um formato digital usando um conversor analógico-digital (ADC – *analog-to-digital converter*). Os dados digitais são armazenados e processados. Os dados então vão para o circuito de reconstrução e de display na sua forma original analógica. A Figura 1–43 mostra um diagrama em bloco básico para um osciloscópio digital.



▲ **FIGURA 1–43**

Diagrama em bloco de um osciloscópio digital.

Controles do Osciloscópio A vista frontal do painel de um osciloscópio é mostrada na Figura 1–44. Existem diferenças entre instrumentos dependendo do modelo e do fabricante, mas a maioria tem certas características comuns. Por exemplo, as duas seções verticais contêm um controle de posição, um botão de acesso ao menu do canal e um controle V/div. A seção horizontal contém um controle segundos/div (SEC/DIV).



▲ **FIGURA 1–44**

Um osciloscópio de dois canais. Os números embaixo na tela indicam os valores de cada divisão nas escalas vertical (tensão) e horizontal (tempo), os quais podem ser ajustados pelos controles do osciloscópio.

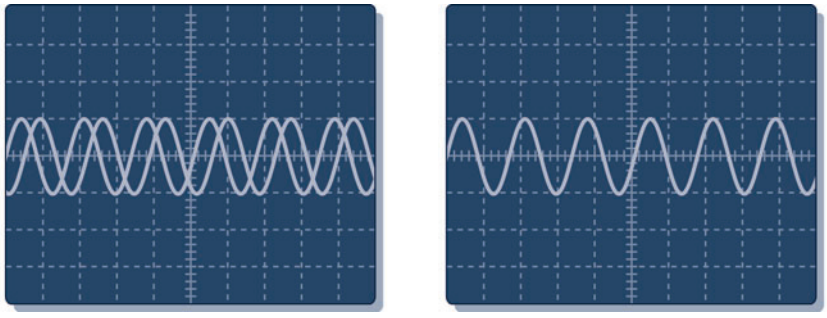
Alguns dos principais controles do osciloscópio são discutidos a seguir. Consulte o manual do usuário para mais saber detalhes do seu osciloscópio.

Controles Verticais Na seção vertical do osciloscópio mostrado na Figura 1–44, podemos ver que existem controles idênticos para cada um dos dois canais (CH1 e CH2). O controle de posição nos permite mover verticalmente para cima e para baixo a forma de onda mostrada na tela. O botão menu provê acesso para seleção entre diversos itens, os quais aparecem na tela, tais como os modos de acoplamento (ac, dc ou GND), ajuste fino ou grosso de V/div, atenuação da ponta de prova, entre outros parâmetros. O controle V/div ajusta o número de volts representado por cada divisão vertical na tela. O valor no qual V/div é ajustado para cada canal é mostrado na parte inferior da tela. O botão MATH MENU provê a opção de seleção de operações que podem ser realizadas sobre as formas de onda de entrada, tal como subtração, adição ou inversão.

Controles Horizontais Na seção horizontal, os controles se aplicam aos dois canais. O controle de posição (Position) nos permite mover horizontalmente para esquerda ou para a direita a forma de onda mostrada na tela. O botão Menu provê a opção de seleção entre diversos itens que aparecem na tela tais como base de tempo principal, visualizar a expansão de uma parte da forma de onda entre outros parâmetros. O controle sec/div ajusta o tempo representado por cada divisão horizontal ou base de tempo principal. O valor ajustado em sec/div é mostrado na parte inferior da tela.

Controles de Disparo (Trigger) Na seção controle de disparo, o controle de nível (Level) determina o ponto na forma de onda a ser sincronizada onde ocorrerá o início da varredura para mostrar a forma de onda de entrada. O botão Menu provê a opção de seleção entre diversos itens que aparecem na tela, incluindo trigger por borda ou por inclinação, fonte de trigger, modo de trigger entre outros parâmetros. Existe também uma entrada para sinal de trigger externo.

O trigger faz com que uma forma de onda estabilize na tela ou que um pulso que ocorre apenas uma vez ou aleatoriamente seja visualizado na tela. Além disso, ele possibilita que observemos atrasos de tempo entre duas formas de onda. A Figura 1–45 compara um sinal não-sincronizado com um sincronizado. O sinal não-sincronizado tende a ficar à deriva na tela, gerando uma visualização de múltiplas formas de onda.

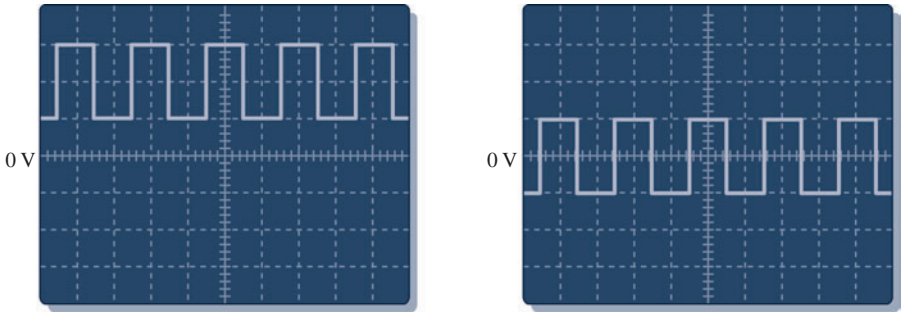


(a) Forma de onda não-sincronizada (b) Forma de onda sincronizada

◀ **FIGURA 1–45**
Comparação entre formas de onda sincronizada e não-sincronizada.

Acoplamento de um Sinal no Osciloscópio O acoplamento é um método usado para conectar ao osciloscópio uma tensão relativa a um sinal. Os acoplamentos DC (cc) e AC (ca) são geralmente selecionados a partir do MENU da seção vertical do osciloscópio. O acoplamento DC nos permite visualizar a componente cc. O acoplamento AC bloqueia a componente cc do sinal de forma que vemos a forma de onda centrada em 0 V. O modo GND nos permite conectar a entrada do canal ao GND para vermos onde está situada na tela a referência 0 V. A Figura 1–46 ilustra o resultado na tela dos acoplamentos DC e AC usando uma forma de onda digital que tem uma componente cc.

A ponta de prova de tensão, mostrada na Figura 1–47, é essencial para a conexão de um sinal ao osciloscópio. Como todos os instrumentos tendem a afetar o circuito a ser medido devido ao efeito de carga, a maioria das pontas de prova dos osciloscópios tem uma resistência em série de



(a) Forma de onda com acoplamento CC

(b) Forma de onda com acoplamento AC

▶ **FIGURA 1–46**

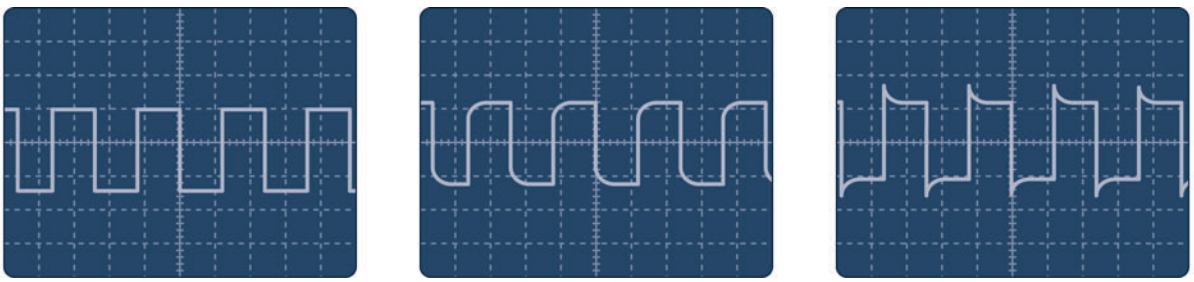
Apresentações da mesma forma de onda com um componente cc.

▶ **FIGURA 1–47**

Ponta de prova de tensão. Imagem apresentada com a permissão da Tektronix, Inc.

alto valor para minimizar este efeito. As pontas de prova que têm uma resistência em série dez vezes maior do que a resistência de entrada do osciloscópio são denominadas pontas de prova ×10. As pontas de prova sem resistência em série são chamadas de pontas de prova ×1. O osciloscópio ajusta sua calibração para a atenuação do tipo da ponta de prova que está sendo usada. Para a maioria das medições deve-se usar a ponta de prova ×10. Entretanto, se estivermos medindo sinais de pequena amplitude, uma ponta de prova ×1 é a melhor escolha.

A ponta de prova tem um ajuste que nos permite compensar a capacitância de entrada do osciloscópio. A maioria dos osciloscópios tem um sinal de saída usado para compensação de pontas de prova que fornece uma forma de onda quadrada calibrada. Antes de efetuar uma medição, devemos nos certificar de que a ponta de prova está compensada adequadamente para eliminar qualquer distorção introduzida. Normalmente, existe um parafuso ou outro mecanismo de ajuste da compensação da ponta de prova. A Figura 1–48 mostra formas de onda para três condições de ponta de prova: adequadamente compensada, subcompensada e sobrecompensada. Se a forma de onda se mostrar subcompensada ou sobrecompensada, ajuste a ponta de prova até obter uma forma de onda quadrada adequadamente compensada.



Compensada adequadamente

Subcompensada

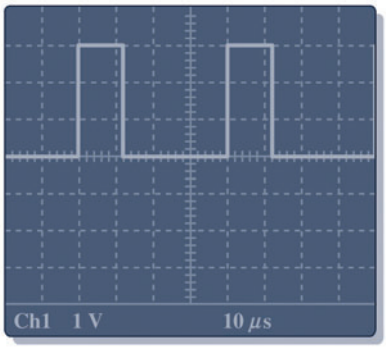
Sobrecompensada

▲ **FIGURA 1–48**

Condições de compensação da ponta de prova.

EXEMPLO 1–3

Baseado no que foi estudado sobre osciloscópios, determine a amplitude e o período da forma de onda digital na tela de um osciloscópio digital conforme mostra a Figura 1–49. Além disso, calcule a freqüência do sinal.



► **FIGURA 1–49**

Solução O controle V/div está ajustado em 1 V. Os pulsos têm uma altura de 3 divisões. Como cada divisão representa 1 V, a amplitude do pulso é

$$\text{Amplitude} = (3\text{ div})((1\text{ V/div}) = \mathbf{3\ V}$$

O ajuste de sec/div é 10 μs. Um ciclo completo da forma de onda (a partir do início de um pulso até o início do próximo) compreende quatro divisões; portanto, o período é

$$\text{Período} = (4\text{ div})(10\ \mu\text{s/div}) = \mathbf{40\ \mu s}$$

A freqüência é calculada como segue

$$f = \frac{1}{T} = \frac{1}{40\ \mu\text{s}} = \mathbf{25\ kHz}$$

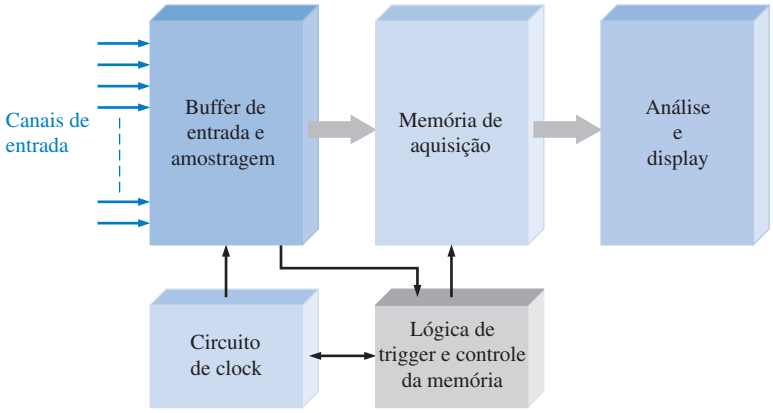
Problema relacionado Para um ajuste em V/div de 4 V e em sec/div de 2 ms, determine a amplitude e o período dos pulsos mostrados na tela da Figura 1–49.

O Analisador Lógico

Analisadores lógicos são usados para medidas de múltiplos sinais digitais e medidas de sinais que apresentam dificuldades de *trigger*. Basicamente, o analisador lógico surgiu como resultado de circuitos microprocessados necessitarem, para a análise de defeito e depuração de programas, muito mais entradas que um osciloscópio oferece. Muitos osciloscópios têm dois canais de entrada e alguns têm quatro canais disponíveis. Os analisadores lógicos são disponibilizados com 34 a 136 canais de entrada. Geralmente, um osciloscópio é usado quando se deseja medir amplitude, freqüência e outros parâmetros de poucos sinais de cada vez ou quando parâmetros como tempos de subida e descida, *overshoot* e atrasos de tempo precisam ser medidos. O analisador lógico é usado quando é necessário determinar os níveis lógicos de um grande número de sinais e para correlacionar os sinais simultâneos com base em suas relações de temporização. A Figura 1–50 mostra um analisador lógico típico e um diagrama em bloco simplificado na Figura 1–51.

Aquisição de Dados O grande número de sinais que podem ser recebidos ao mesmo tempo é o principal fator que distingue um analisador lógico de um osciloscópio. Geralmente, os dois tipos de aquisição de dados em um analisador lógico são a temporização e o estado lógico. A aquisição de temporização é usada principalmente quando a relação de temporização entre os diversos sinais

▶ **FIGURA 1–50**
Analisador lógico. Imagem apresentada com a permissão da Tektronix, Inc.



▶ **FIGURA 1–51**
Diagrama em bloco simplificado de um analisador lógico.

precisa ser determinada. A aquisição de estado lógico é usada quando precisamos visualizar a seqüência de estados lógicos conforme eles aparecem num sistema sob teste.

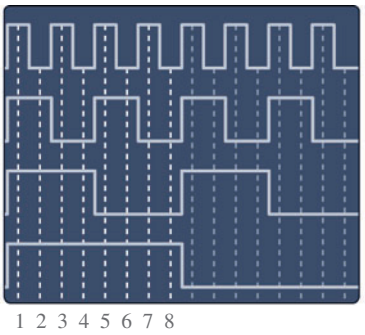
Freqüentemente, é útil correlacionar temporização com os estados dos dados, sendo que a maioria dos analisadores lógicos podem simultaneamente fazer a aquisição desses dados. Entretanto, a condição inválida pode ser causada por uma violação de temporização no sistema sob teste. Sem os dois tipos de informações disponíveis ao mesmo tempo, a isolação do problema pode ser muito difícil.

Número de Canais e Capacidade de Memória Os analisadores lógicos contêm uma memória de aquisição de tempo real na qual os dados amostrados dos canais são armazenados à medida que eles acontecem. As duas características de importância fundamental são o número de canais e a capacidade da memória. A memória de aquisição pode ser pensada como tendo uma largura igual ao número de canais e uma capacidade que é o número de bits que pode ser amostrado por cada canal durante um intervalo de tempo.

O número de canais determina o número de sinais que podem ser amostrados simultaneamente. Em certos tipos de sistemas, trabalha-se com um grande número de sinais, como os do barramento de dados num sistema microprocessado. A capacidade da memória de aquisição determina a quantidade de dados a partir de um determinado canal que podemos visualizar em qualquer instante de tempo.

Análise e Modos de Apresentação Uma vez amostrados os dados e armazenados na memória de aquisição, eles podem ser usados em diversos modos de apresentação e análise diferentes. A forma de onda apresentada é muito parecida com a apresentada por um osciloscó-

pio, onde podemos ver a relação temporal dos múltiplos sinais. A apresentação na forma de lista indica os estados do sistema sob teste mostrando os valores das formas de onda (1s e 0s) de entrada em vários pontos no tempo (pontos amostrados). Tipicamente, esses dados podem ser apresentados em hexadecimal ou outros formatos. A Figura 1–52 mostra versões simplificadas desses dois modos de apresentação. A apresentação na forma de lista de amostras corresponde aos pontos amostrados (identificados na cor cinza) no modo de apresentação de forma de onda. Estudaremos os números binário e hexadecimal (hexa) no próximo capítulo.



(a) Apresentação em forma de onda

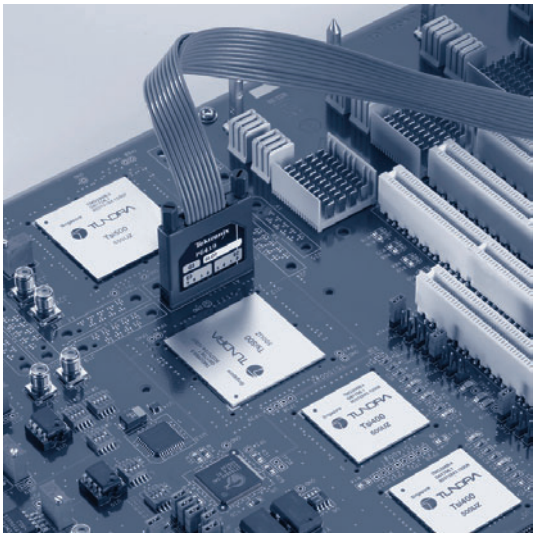
Sample	Binary	Hex	Time
1	1111	F	1 ns
2	1110	E	10 ns
3	1101	D	20 ns
4	1100	C	30 ns
5	1011	B	40 ns
6	1010	A	50 ns
7	1001	9	60 ns
8	1000	8	70 ns

(a) Apresentação em forma de lista

◀ **FIGURA 1–52**
Dois modos de tela de apresentação de analisador lógico.

No teste de computadores e sistemas baseados em microprocessador dois modos a mais são úteis: rastreamento de instruções e depuração de código fonte. O modo de rastreamento de instruções determina e mostra as instruções em curso, por exemplo, no barramento de dados de um sistema baseado em microprocessador. Nesse modo, os códigos de operação e os mnemônicos (nomes originários do inglês) de instruções são geralmente mostrados, bem como os correspondentes endereços de memória. Muitos analisadores lógicos também incluem um modo de depuração de código-fonte, o qual essencialmente nos permite ver o que realmente está acontecendo no sistema sob teste quando uma instrução é executada.

Pontas de Prova Três tipos básicos de pontas de prova são usadas com analisadores lógicos. Uma delas é a ponta de prova multicanal comprimida, que pode ser fixada nos pontos de medição na placa de circuito, conforme mostra a Figura 1–53. Um outro tipo de ponta de prova multicanal, similar a que é mostrada, é conectada em um soquete dedicado montado na placa de circuito. Um terceiro tipo é uma ponta de prova do tipo clip de único canal.



◀ **FIGURA 1–53**
Uma típica ponta de prova de um analisador lógico de múltiplos canais. Imagem apresentada com a permissão da Tektronix, Inc.

Geradores de Sinais

Fonte de Sinal Lógico Esses instrumentos também são conhecidos como geradores de pulsos e geradores padrão. Eles são projetados especificamente para gerar sinais digitais com amplitudes e bordas precisas e para gerar seqüências de 1s e 0s necessárias nos testes de barramentos de computadores, microprocessadores e outros sistemas digitais.

Geradores de Formas de Onda Arbitrárias e Geradores de Funções O gerador de formas de onda arbitrárias pode ser usado para gerar sinais padrão como ondas senoidais, triangulares e pulsos, bem como sinais com vários formatos e características. Formas de onda podem ser definidas por entradas matemáticas ou gráficas. Um gerador de formas de onda arbitrárias é mostrado na Figura 1–54(a).

O gerador de funções gera formas de onda digitais, bem como senoidais e triangulares. A maioria dos geradores de funções tem saídas compatíveis com circuitos lógicos para provê o nível de tensão adequado e acionar entradas de circuitos digitais. A Figura 1–54(b) mostra tipos de geradores de funções típicos.

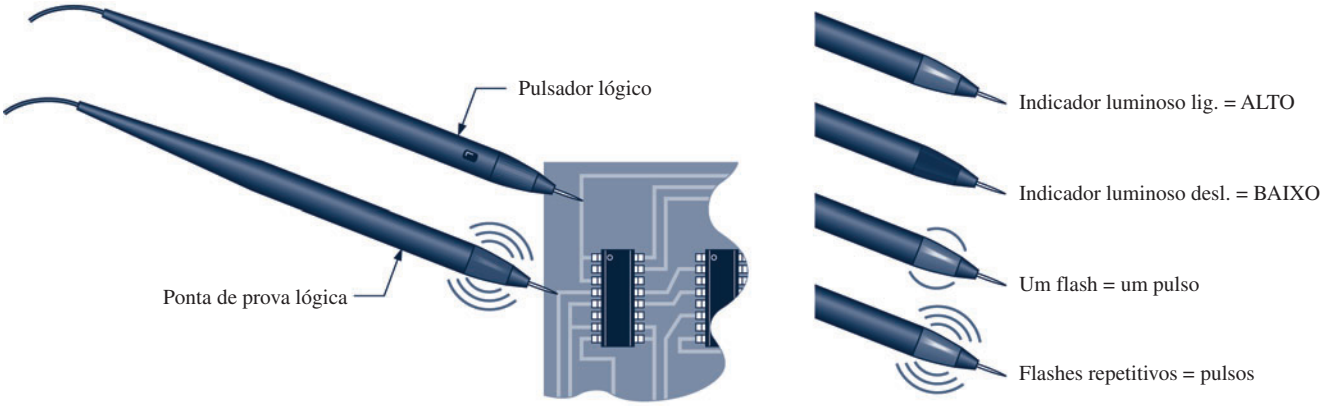
(a) Um gerador de forma de onda arbitrária.

(b) Exemplos de geradores de funções.

▲ **FIGURA 1–54**

Geradores de sinais típicos. Imagens apresentadas com a permissão da Tektronix, Inc.

A Ponta de Prova Lógica e o Pulsador Lógico A ponta de prova lógica é uma ferramenta portátil conveniente e barata que proporciona um recurso para análise de defeito de um circuito digital detectando várias condições em um ponto do circuito, conforme ilustrado na Figura 1–55. Essa ponta de prova pode detectar tensões de nível ALTO e BAIXO, pulsos que ocorrem uma vez,



▲ **FIGURA 1–55**

Ilustração de como um pulsador lógico e uma ponta de prova lógica podem ser usados para aplicar um pulso em um determinado ponto e verificar o resultado do pulso em outras partes do circuito.

pulsos repetitivos e circuito aberto em uma placa de circuito. O indicador luminoso sinaliza a condição que existe num certo ponto, conforme indicado na figura.

O pulsador lógico produz uma forma de onda digital (pulsos repetitivos) que pode ser aplicada em qualquer ponto do circuito. Podemos aplicar pulsos em um ponto do circuito com um pulsador lógico e verificar em outro ponto o resultado dos pulsos com uma ponta de prova lógica.

Outros Instrumentos

A Fonte de Alimentação CC Esse instrumento é indispensável em qualquer bancada de teste. A fonte de alimentação converte potência ca de uma tomada comum de parede em uma tensão cc regulada. Todos os circuitos digitais necessitam de tensão cc. Muitos circuitos lógicos necessitam de +5 V ou +3,3 V para operarem. A fonte de alimentação é usada para alimentar os circuitos nas etapas de projeto, desenvolvimento e análise de defeito quando o sistema não possui uma fonte de alimentação própria. A Figura 1–56 mostra fontes de alimentação típicas de bancadas de testes.

◀ **FIGURA 1–56**
Fontes de alimentação cc típicas. Cortesia da B + K Precision.®

O Multímetro Digital (DMM) O DMM (*digital multimeter*) é usado em medições de tensão cc e ca e resistência. A Figura 1–57 mostra DMMs típicos usados em bancadas de teste.

◀ **FIGURA 1–57**
DMMs típicos. Cortesia da B + K Precision.®

SEÇÃO 1–7 REVISÃO

1. Qual é a principal diferença entre um osciloscópio analógico e um digital?
2. Enuncie as duas diferenças principais entre um analisador lógico e um osciloscópio?
3. Para que serve o controle V/div num osciloscópio?
4. Para que serve o controle sec/div num osciloscópio?
5. Qual é a finalidade de um gerador de funções?

APLICAÇÕES EM SISTEMAS DIGITAIS

Nesse tópico de aplicações em sistemas digitais, apresentamos uma aplicação simplificada de um sistema que contém os elementos lógicos e funções discutidas na Seção 1–4. É importante que o leitor entenda como diversas funções lógicas podem operar juntas constituindo um sistema para desempenhar uma tarefa específica. É importante também iniciar o desenvolvimento do raciocínio em termos de operação sistêmica porque, na prática, uma grande parte do seu trabalho estará voltada para sistemas, em vez de particularidades como funções específicas de circuitos. É claro que, para entender os sistemas, temos que primeiro entender as funções e os elementos básicos que formam o sistema.

Esta seção de aplicações em sistemas digitais apresenta o conceito de sistema. O exemplo mostra como funções lógicas podem trabalhar juntas para realizar uma tarefa mais complexa e fazer com que o leitor comece a pensar de forma sistêmica. O sistema específico apresentado aqui para ilustrar o conceito de sistema serve como um modelo instrucional, não sendo necessário uma abordagem que seria usada na prática, embora pudesse ser feito. Em aplicações modernas industriais como a que é discutida aqui, equipamentos conhecidos como controladores programáveis são freqüentemente usados.

Sobre o Sistema

Vamos imaginar que uma fábrica usa o sistema de controle de processos mostrado no diagrama em blocos simplificado na Figura 1–58 para a contagem automática e engarrafamento de comprimidos. Os comprimidos são inseridos num largo alimentador em forma de funil. O pequeno ‘ “pescoço” do funil permite a passagem de apenas um comprimido de cada vez que cai dentro de uma garrafa na correia transportadora logo abaixo.

O sistema digital controla a quantidade de comprimidos colocados em cada garrafa e mostra num display o número total que é atualizado continuamente próximo à linha transportadora bem como num local remoto em outra parte da planta de produção. Esse sistema utiliza todas as funções lógicas básicas que foram apresentadas na Seção 1–4, sendo que o seu único propósito é mostrar como essas funções podem ser combinadas para alcançar um resultado desejado.

A operação geral é descrita a partir de agora. Um sensor óptico na parte inferior do pescoço do funil detecta cada comprimido que passa e produz um pulso elétrico. Esse pulso vai para um contador que avança uma contagem; portanto, a qualquer momento durante o enchimento de uma garrafa o contador contém a representação em binário do número de comprimidos na garrafa. A contagem em binário é transferida do contador para a entrada *B* do comparador (comp.) através de linhas paralelas. O número binário que define o número de comprimidos em cada garrafa é colocado na entrada *A* do comparador. Este número vem do teclado e circuitos associados, os quais incluem codificador, registrador *A* e conversor de código *A*. Quando o número desejado de comprimidos é digitado no teclado, ele é codificado e então armazenado no registrador paralelo *A* até que uma alteração na quantidade de comprimidos por garrafa seja necessária.

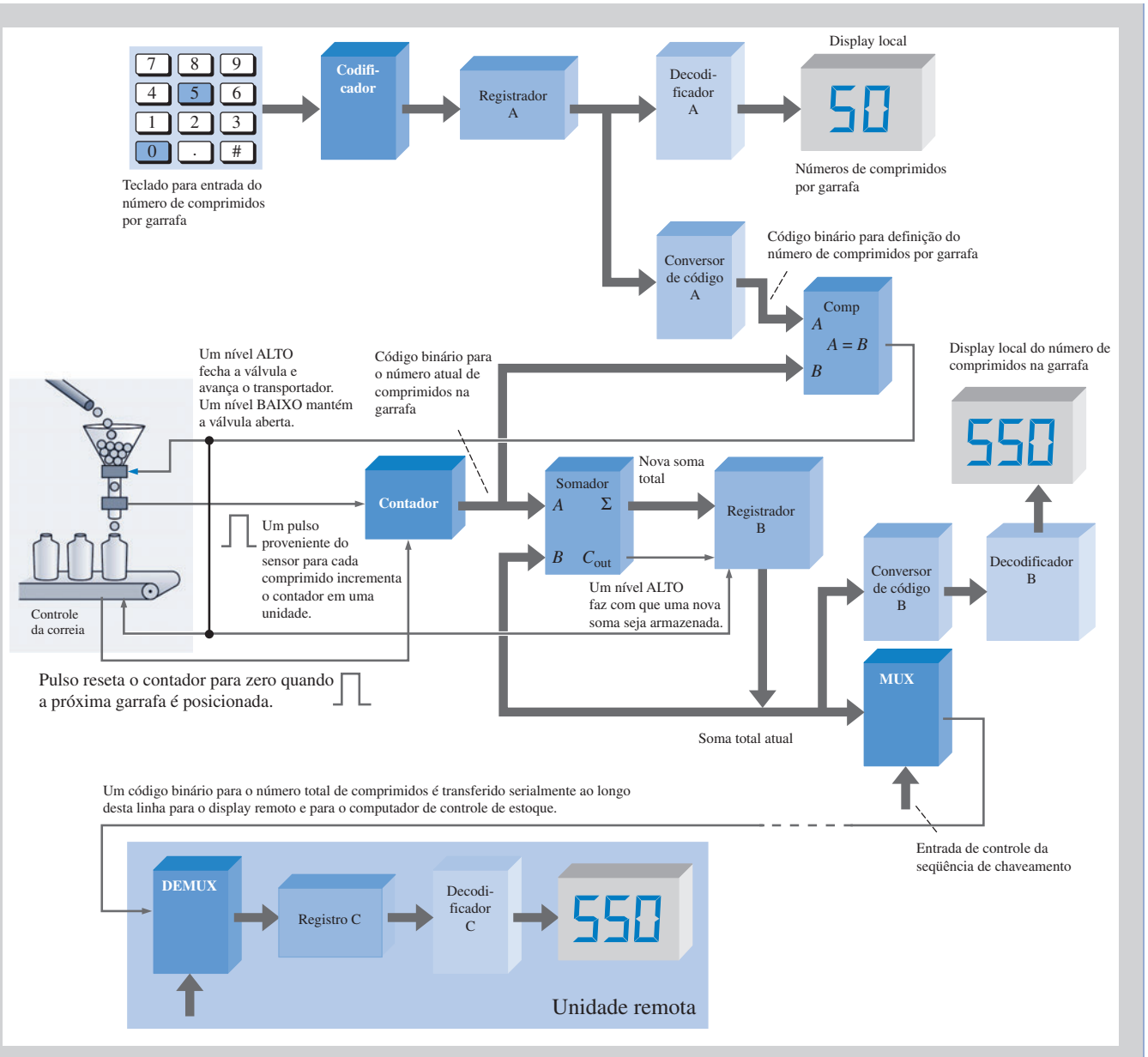
Suponha, por exemplo, que cada garrafa comporte 50 comprimidos. Quando o número no contador chegar a 50, a saída $A=B$ do comparador será nível ALTO, indicando que a garrafa está cheia.

A saída do comparador em nível ALTO faz fechar a válvula no pescoço do funil parando o fluxo de comprimidos e ao mesmo tempo ativa a correia transportadora que move a próxima garrafa para a posição logo abaixo do funil. Quando a próxima garrafa é posicionada adequadamente sob o pescoço do funil, o circuito de controle da correia transportadora produz um pulso que reseta o contador para zero. A saída $A=B$ do comparador cai para nível BAIXO, abrindo a válvula no funil e reinicia o fluxo de comprimidos.

Na parte do sistema que compreende o circuito do display, o número no contador é transferido em paralelo para a entrada *A* do somador. A entrada *B* do somador vem do registrador paralelo *B* que guarda o número total de comprimidos engarrafado até a última garrafa ser preenchida. Por exemplo, se dez garrafas foram preenchidas e cada garrafa tem 50 comprimidos, o registrador *B* contém a representação binária para 500. Então, quando a próxima garrafa for preenchida, o número binário equivalente a 50 aparece na entrada *A* do somador e o número binário para 500 aparece na entrada *B*. O somador produz uma nova soma de 550, que é armazenada no registrador *B*, substituindo a soma anterior que era 500.

O número binário no registrador B é transferido em paralelo para o conversor de código e decodificador, que converte do formato binário para o decimal para mostrar no display situado na linha transportadora. O número binário no registrador também é transferido para um multiplexador (mux) de forma que ele possa ser convertido da forma paralela para a serial e transmitido ao longo de uma única linha para um local remoto a uma determinada distância. É mais econômico instalar uma única linha do que várias linhas paralelas quando distâncias significativas estão envolvidas e a velocidade da transmissão de dados não é um fator importante nesta aplicação. No local remoto, os dados em formato serial são demultiplexados e enviados para o registrador *C*. Deste registro os dados são então decodificados para o display da unidade remota.

Tenha em mente que esse sistema é puramente um modelo instrucional e não representa necessariamente a forma mais atual e eficiente de implementar esse processo hipotético. Embora existam outras abordagens, essa abordagem foi usada para ilustrar uma aplicação de funções lógicas que foram introduzidas na Seção 1–4 e que serão abordadas em detalhes nos próximos capítulos. O leitor acaba de analisar uma aplicação de vários dispositivos funcionais em nível sistêmico e como eles podem ser interconectados para atender a um objetivo específico.



▲ **FIGURA 1–58**

Diagrama em bloco simplificado para um sistema de contagem de comprimidos e controle de engarrafamento.

RESUMO

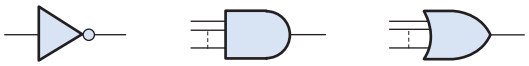
- Uma grandeza analógica tem valores contínuos.
- Uma grandeza digital tem um conjunto de valores discretos.
- Um dígito binário é denominado bit.
- Um pulso é caracterizado por tempo de subida, tempo de descida, largura de pulso (duração) e amplitude.
- A freqüência de uma onda periódica é o inverso do período. A fórmula que relaciona a freqüência e o período é

$$f = \frac{1}{T} \quad \text{e} \quad T = \frac{1}{f}$$

- O ciclo de trabalho de uma forma de onda digital é a razão entre a largura do pulso e o período, expresso pela seguinte fórmula em porcentagem:

$$\text{Ciclo de trabalho} = \left(\frac{t_W}{T}\right)100\%$$

- Um diagrama de temporização é um arranjo de duas ou mais formas de onda mostrando a relação temporal entre elas.
- As três operações lógicas básicas são NOT, AND e OR. O símbolo padrão de cada uma é dado na Figura 1–59.



▶ **FIGURA 1–59**

- As funções lógicas básicas são: comparação, aritmética, conversão de código, decodificação, codificação, seleção de dados, armazenamento e contagem.
- As duas grandes categorias físicas de encapsulamento de CIs são PTH (para dispositivos cujos pinos passam através de furos na placa de circuito impresso) e SMD (para dispositivos montados na superfície da placa de circuito impresso).
- As categorias de CIs em termos de complexidade do circuito são SSI (integração em pequena escala), MSI (integração em média escala), LSI, VLSI e ULSI (larga escala, escala muito ampla, escala ultra ampla)
- Os dois tipos de SPLDs (dispositivos de lógica programável simples) são PAL (lógica de arranjo programável) e GAL (lógica de arranjo genérico).
- O CPLD (dispositivo lógico programável complexo) contém múltiplos SPLDs com interconexões programáveis.
- O FPGA (arranjo de porta programável por campo) tem uma estrutura interna diferente comparada a de um CPLD e é geralmente usado para sistemas e circuitos mais complexos.
- Os instrumentos comumente usados em teste e análise de defeito de circuitos digitais são: osciloscópio, analisador lógico, gerador de formas de onda, gerador de funções, fonte de alimentação cc, multímetro digital, ponta de prova lógica e pulsador lógico.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Análise de defeito A técnica ou processo de identificação, isolação e correção sistemática de um defeito num circuito ou sistema.

Analógico Tem valores contínuos no tempo.

AND Uma função lógica básica na qual uma saída verdadeira (nível ALTO) ocorre apenas quando todas as condições de entrada são verdadeiras (nível ALTO).

Binário Que apresenta dois valores ou estados; descreve um sistema de numeração que tem base dois e utiliza os dígitos 0 e 1.

Bit Um dígito binário, que pode ser 1 ou 0.

Circuito integrado (CI) Um tipo de circuito no qual todos os componentes são integrados num único chip de material semicondutor de tamanho extremamente reduzido.

Clock O sinal de temporização básico num sistema digital; uma forma de onda periódica na qual cada intervalo entre pulsos é igual ao tempo de um bit.

Compilador Um programa que controla o processo do fluxo do projeto e traduz o código fonte em código objeto num formato que pode ser testado logicamente ou transferido para o dispositivo de destino.

CPLD Dispositivo lógico programável complexo que consiste basicamente de múltiplos arranjos SPLD com interconexões programáveis.

Dados Informação na forma numérica, alfabética ou outra forma.

Diagrama de temporização Um gráfico de formas de onda digitais que mostra a relação temporal de duas ou mais formas de onda.

Digital Relativo a dígitos ou quantidades discretas; tem um conjunto de valores discretos.

FPGA Arranjo de porta programável por campo.

Input O sinal enviado para um circuito ou a linha que chega ao circuito.

Inversor Um circuito NOT; um circuito que comuta um nível ALTO para um nível BAIXO ou vice-versa.

Lógica Em eletrônica digital, a capacidade dos circuitos das portas de tomar decisões, para os quais um nível ALTO representa uma sentença verdadeira e um nível BAIXO uma sentença falsa.

NOT Uma operação lógica básica que realiza inversões.

OR Uma operação lógica básica na qual uma saída verdadeira (nível ALTO) ocorre quando uma ou mais das condições de entrada são verdadeiras (nível ALTO).

Paralelo Em sistemas digitais, dados que ocorrem simultaneamente em diversas linhas; a transferência ou processamento de vários bits simultaneamente.

Porta Um circuito lógico que realiza uma operação lógica específica tal como uma AND ou uma OR.

Pulso Uma mudança súbita do sinal de um nível para outro e, após um tempo (denominado de largura de pulso), retorna subitamente para o nível original.

Saída O sinal ou a linha que sai de um circuito.

Serial Que tem um elemento seguido de outro, como em uma transferência serial de bits; ocorre em seqüência em vez de simultaneamente.

SPLD Dispositivo lógico programável simples.

AUTOTESTE

As respostas estão no final do capítulo.

1. Uma grandeza que apresenta valores contínuos é
   (a) uma grandeza digital **(b)** uma grandeza analógica
   (c) uma grandeza binária **(d)** uma grandeza natural
2. O termo *bit* significa
   (a) uma pequena quantidade de dados **(b)** um 1 ou um 0
   (c) dígito binário **(d)** as respostas (b) e (c) estão corretas
3. O intervalo de tempo relativo à borda positiva de um pulso entre 10% e 90% da amplitude é
   (a) tempo de subida **(b)** tempo de descida
   (c) largura de pulso **(d)** período
4. Um pulso em uma determinada forma de onda ocorre a cada 10 ms. A freqüência é
   (a) 1kHz **(b)** 1Hz **(c)** 100 Hz **(d)** 10Hz
5. Em uma determinada forma de onda digital, o período é duas vezes a largura do pulso. O ciclo de trabalho é
   (a) 100% **(b)** 200% **(c)** 50%
6. Um inversor
   (a) realiza a operação NOT
   (b) comuta de um nível ALTO para um nível BAIXO
   (c) comuta um nível BAIXO para um nível ALTO
   (d) todas as alternativas acima estão corretas
7. A saída de uma porta AND é nível ALTO quando
   (a) qualquer entrada for nível ALTO **(b)** todas as entradas são nível ALTO
   (c) nenhuma entrada for nível ALTO **(d)** as alternativas (a) e (b) estão corretas
8. A saída de uma porta OR é nível ALTO quando
   (a) qualquer entrada for nível ALTO **(b)** todas as entradas são nível ALTO
   (c) nenhuma entrada for nível ALTO **(d)** as alternativas (a) e (b) estão corretas
9. O dispositivo usado para converter um número binário para um display de 7 segmentos é o
   (a) multiplexador **(b)** codificador
   (c) decodificador **(d)** registrador
10. Um exemplo de um dispositivo de armazenamento de dados é
    (a) a porta lógica **(b)** o flip-flop
    (c) o comparador **(d)** o registrador
    (e) as alternativas (b) e (d) estão corretas
11. Um encapsulamento de CI de função fixa que contém quatro portas AND é um exemplo de
    (a) MSI **(b)** SMT **(c)** SOIC **(d)** SSI

12. Um dispositivo LSI tem uma complexidade de
(a) 10 a 100 portas equivalentes **(b)** mais de 100 a 10.000 portas equivalentes
(c) 2.000 a 5.000 portas equivalentes **(d)** mais de 10.000 a 100.000 portas equivalentes

13. VHDL é um(a)
(a) dispositivo lógico **(b)** linguagem de programação de PLD
(c) linguagem de computador **(d)** lógica de densidade muito alta

14. Um CPLD é um
(a) dispositivo de lógica programável controlado **(b)** acionador lógico programável complexo
(c) dispositivo lógico programável complexo **(d)** dispositivo lógico de processamento central

15. Um FPGA é um(a)
(a) arranjo de porta programável por campo **(b)** arranjo de porta programável rápido
(c) arranjo genérico programável por campo **(d)** aplicação de porta de processamento rápido

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 1–1 Grandezas Analógicas e Digitais

1. Cite duas vantagens dos dados digitais em comparação com dados analógicos.

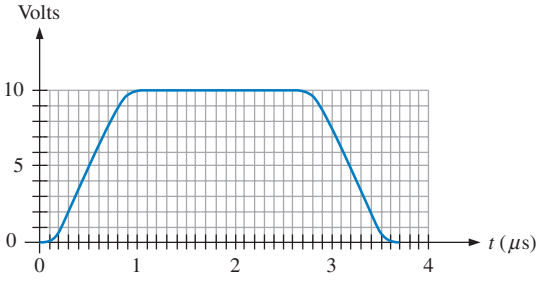
2. Cite uma grandeza analógica que não seja a temperatura nem a intensidade sonora.

SEÇÃO 1–2 Dígitos Binários, Níveis Lógicos e Formas de Onda Digitais

3. Defina a seqüência e bits (1s e 0s) representada por cada uma das seguintes seqüências de níveis:
(a) ALTO, ALTO, BAIXO, ALTO, BAIXO, BAIXO, BAIXO, ALTO
(b) BAIXO, BAIXO, BAIXO, ALTO, BAIXO, ALTO, BAIXO, ALTO, BAIXO

4. Faça uma lista da seqüência dos níveis (ALTO e BAIXO) que representa cada uma das seguintes seqüências de bits:
(a) 1 0 1 1 1 0 1 **(b)** 1 1 1 0 1 0 0 1

5. Para o pulso mostrado na Figura 1–60 determine graficamente:
(a) o tempo de subida **(b)** o tempo de descida **(c)** a largura de pulso **(d)** a amplitude



▶ FIGURA 1–60

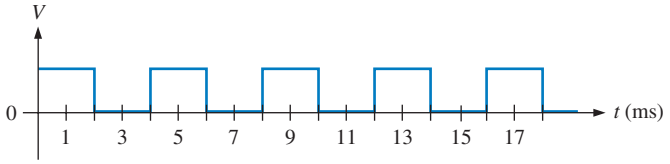
6. Determine o período da forma de onda digital mostrada na Figura 1–61?

7. Qual é a freqüência da forma de onda vista na Figura 1–61?

8. A forma de onda digital mostrada na Figura 1–61 é periódica ou não-periódica?

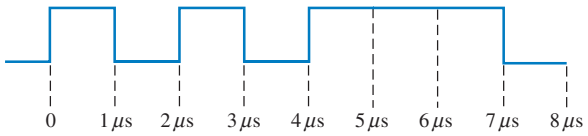
9. Determine o ciclo de trabalho da forma de onda mostrada na Figura 1–61.

10. Determine a seqüência de bits representada pela forma de onda vista na Figura 1–62. O tempo de bit é 1 μs nesse caso.



▲ FIGURA 1–61

11. Qual é o tempo total da transferência serial para os oito bits mostrados na Figura 1–62? Qual é o temo total para uma transferência em paralelo?



▲ FIGURA 1–62

SEÇÃO 1–3 Operações Lógicas Básicas

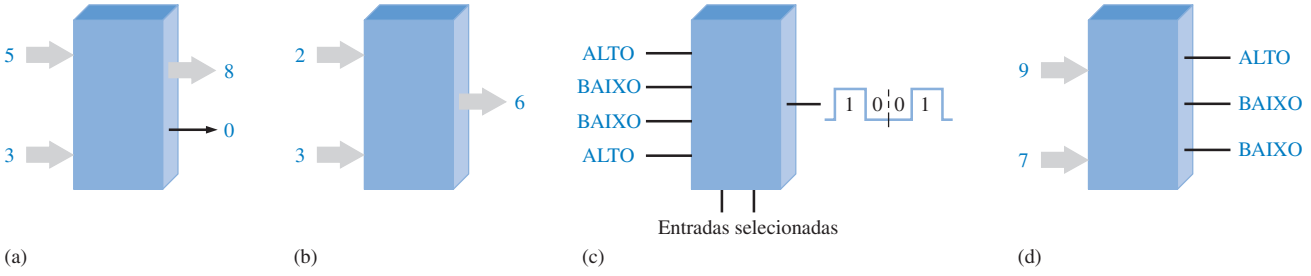
12. Um circuito lógico necessita de nível ALTO em todas as suas entradas para tornar a saída nível ALTO. Que tipo de circuito é esse?

13. Um circuito lógico básico de duas entradas tem um nível ALTO numa entrada e um nível BAIXO na outra, sendo a saída nível BAIXO. Identifique o circuito.

14. Um circuito lógico básico de duas entradas tem um nível ALTO numa entrada e um nível BAIXO na outra, sendo a saída nível ALTO. Qual é o tipo do circuito lógico?

SEÇÃO 1–4 Visão Geral das Funções Lógicas Básicas

15. Cite a função lógica de cada bloco na Figura 1–63 observando as entradas e saídas.



▲ FIGURA 1–63

16. Uma forma de onda digital com uma freqüência de 10 kHz é aplicado na entrada de um contador. Durante 100 ms, quantos pulsos são contados?

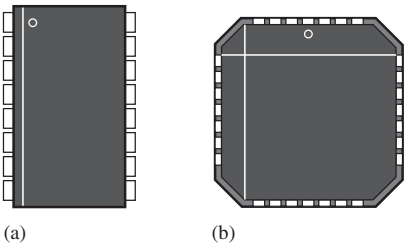
17. Considere um registrador que pode armazenar oito bits. Vamos admitir que ele tenha sido inicializado de forma que ele passa a ter zero em todas as posições. Se transferirmos quatro bits alternados (0101) serialmente para o registrador, começando pelo 1 e deslocando para a direita, qual é o conteúdo final do registrador logo que os quatro bits forem armazenados?

SEÇÃO 1–5 Circuitos Integrados de Funções Fixas

18. Um CI digital de função fixa tem uma complexidade de 200 portas equivalentes. Como ele é classificado?

19. Explique a principal diferença entre os encapsulamentos DIP e SMT.

20. Determine a numeração dos pinos para os encapsulamentos vistos na Figura 1–64. Na figura são mostradas as vistas superiores.



▶ FIGURA 1–64

SEÇÃO 1–6 Introdução à Lógica Programável

21. Qual dos seguintes acrônimos não descreve lógica programável?

PAL, GAL, SPLD, ABEL, CPLD, CUPL, FPGA

22. O que significa cada um dos seguintes acrônimos?

(a) SPLD **(b)** CPLD **(c)** HDL **(d)** FPGA **(e)** GAL

23. Defina cada um dos seguintes termos relativos à programação de PLDs.

(a) inserção do projeto **(b)** simulação **(c)** compilação **(d)** download

24. Descreva o processo place e rout.

SEÇÃO 1–7 Instrumentos de Medição e Teste

25. Um pulso é mostrado na tela de um osciloscópio sendo que a linha de base mede 1 V e a parte superior mede 8 V. Qual é a amplitude?

26. Uma ponta de prova lógica é colocada num dos terminais de um CI em operação num sistema. O indicador luminoso da ponta de prova pisca repetidamente. O que isso indica?

SEÇÃO 1–8 Aplicações em Sistemas Digitais

27. Defina o termo *sistema*.

28. No sistema ilustrado na Figura 1–58, qual é a necessidade do multiplexador e do demultiplexador?

29. O que pode ser feito para mudar a quantidade de comprimidos por garrafa no sistema da Figura 1-58?

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 1–1 Grandezas Analógicas e Digitais

1. *Analógico* significa contínuo.

2. *Digital* significa discreto.

3. Uma grandeza digital tem um conjunto de valores discretos e uma grandeza analógica tem um conjunto de valores contínuos.

4. Um sistema de som é analógico. Um aparelho de CD é analógico e digital. Um computador é totalmente digital.

SEÇÃO 1–2 Dígitos Binários, Níveis Lógicos e Formas de Onda Digitais

1. Binário significa que tem dois estados ou valores.

2. Um bit é um dígito binário.

3. Os bits são 1 e 0.

4. Tempo de subida: de 10% a 90% da amplitude. Tempo de descida: de 90% a 10% da amplitude.

5. A freqüência é o inverso do período.

6. Uma forma de onda de clock é uma forma de onda de temporização básica a partir da qual outras formas de onda são derivadas.

7. Um diagrama de tempo mostra a relação temporal de duas ou mais formas de onda.

8. A transferência paralela é mais rápida que a transferência serial.

SEÇÃO 1–3 Operações Lógicas Básicas

1. Quando a entrada for nível BAIXO.

2. Quando todas as entrada forem nível ALTO.

3. Quando qualquer uma das entradas for nível ALTO.

4. Um inversor é um circuito NOT.

5. Uma porta lógica é um circuito que realiza uma operação lógica (AND, OR).

SEÇÃO 1–4 Visão Geral das Funções Lógicas Básicas

1. Um comparador compara a magnitude de dois números colocados nas entradas.
2. Soma, subtração, multiplicação e divisão.
3. A codificação é uma conversão de uma forma familiar, como a forma decimal, em uma forma codificada, como a forma binária.
4. A decodificação é uma conversão de um código para uma forma familiar, como acontece na conversão de binário para decimal.
5. A multiplexação coloca dados de diversas fontes em uma linha. A demultiplexação recebe os dados de uma linha e distribui os dados para diversos destinos.
6. Flip-flops, registradores, memórias semicondutoras e discos magnéticos.
7. Um contador conta eventos com uma seqüência de estados binários.

SEÇÃO 1–5 Circuitos Integrados de Funções Fixas

1. Um CI é um circuito eletrônico em que todos os componentes são integrados num único chip de silício.
2. DIP – encapsulamento com pinos em duas linhas; SMT – tecnologia de montagem em superfície; SOIC – circuito integrado de perfil baixo; SSI – integração em pequena escala; MSI – integração em escala média; LSI – integração em escala ampla; VLSI – integração em escala muito ampla; ULSI – integração em escala ultra ampla.
3. **(a)** SSI **(b)** MSI **(c)** LSI **(d)** VSLI **(e)** ULSI

SEÇÃO 1–6 Introdução à Lógica Programável

1. Dispositivo lógico programável simples (SPLD), dispositivo lógico programável complexo (CPLD) e arranjo lógico programável por campo (FPGA).
2. Uma CPLD é constituída de múltiplas SPLDs.
3. Inserção do projeto, simulação funcional, síntese, implementação, simulação de temporização e Download.
4. *Inserção do projeto*: o projeto lógico é inserido usando um software de desenvolvimento. *Simulação funcional*: o projeto é simulado com o uso de um software para garantir o correto funcionamento lógico. *Síntese*: o projeto é traduzido em uma netlist. *Implementação*: a lógica desenvolvida pela netlist é mapeada num dispositivo programável. *Simulação de temporização*: o projeto é simulado por software para confirmar que não existem problemas de temporização. *Download*: o projeto é inserido num dispositivo programável.

SEÇÃO 1–7 Instrumentos de Medição e Teste

1. O osciloscópio analógico transfere a forma de onda medida diretamente para o circuito do display. O osciloscópio digital converte primeiro o sinal medido para o formato digital.
2. O analisador lógico tem mais canais que o osciloscópio e tem mais que um formato de apresentação de dados.
3. O controle V/div ajusta a tensão para cada divisão na tela.
4. O controle sec/div ajusta o tempo para cada divisão na tela.
5. O gerador de funções produz diversos tipos de formas de onda.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

1–1. f = 6,67 kHz; Ciclo de trabalho = 16,7%

1–2. Transferência em paralelo: 100 ns; Transferência serial: 1,6 μs

1–3. Amplitude = 12 V; T = 8 ms

AUTOTESTE

1. (b) **2.** (d) **3.** (a) **4.** (c) **5.** (c) **6.** (d) **7.** (b) **8.** (d)
9. (c) **10.** (e) **11.** (d) **12.** (d) **13.** (b) **14.** (c) **15.** (a)

2

Sistemas de Numeração, Operações e Códigos

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Revisar o sistema de numeração decimal
- Contar no sistema de numeração binário
- Converter de decimal para binário e vice-versa
- Aplicar operações aritméticas em números binários
- Determinar os complementos de 1 e de 2 de um número binário
- Expressar números binários sinalizados nos formatos sinal-magnitude, complemento de 1, complemento de 2 e ponto flutuante.
- Realizar operações aritméticas com carry de saída sobre números binários sinalizados
- Realizar conversões entre os sistemas de numeração binário e hexadecimal
- Somar números na forma hexadecimal
- Realizar conversões entre os sistemas de numeração binário e octal
- Expressar números decimais na forma de decimal codificado em binário (BCD)
- Somar números BCD
- Realizar conversões entre o sistema binário e o código gray
- Interpretar o código padrão americano para troca de informações (ASCII)
- Explicar como detectar e corrigir erros de código

TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- LSB
- MSB
- Byte
- Número de ponto flutuante
- Hexadecimal
- Octal
- BCD
- Alfanumérico
- ASCII
- Paridade
- Código Hamming

INTRODUÇÃO

O sistema de numeração binário e os códigos digitais são fundamentais para os computadores e para a eletrônica digital em geral. Nesse capítulo, estudaremos o sistema de numeração binário e as suas relações com outros sistemas de numeração como decimal, hexadecimal e octal. Abordaremos as operações aritméticas com números binários com o intuito de fornecer uma base de conhecimento para o entendimento de como os computadores e muitos outros sistemas digitais funcionam. Além disso, abordaremos também o código BCD (decimal codificado em binário), o código Gray e o código ASCII. O método da paridade para detecção de erros em códigos é introduzido e um método de correção de erro é descrito. Os tutoriais sobre o uso de calculadoras em certas operações se baseiam na calculadora gráfica TI-86 e na calculadora TI-36X. Os procedimentos mostrados podem diferir de outros tipos de calculadoras.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

2-1 NÚMEROS DECIMAIS

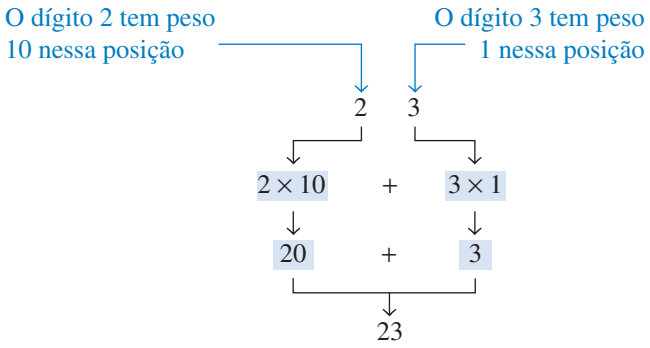
O leitor já tem familiaridade com o sistema de numeração decimal porque deve usar números decimais todos os dias. Embora os números decimais sejam comuns, a sua estrutura de pesos não é freqüentemente compreendida. Nesta seção, a estrutura dos números decimais é revisada. Essa revisão lhe ajudará a compreender mais facilmente a estrutura do sistema de numeração binário, o qual é importante no estudo de computadores e eletrônica digital.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar por que o sistema de numeração decimal é um sistema em que os dígitos apresentam pesos ■ Explicar como potências de dez são usadas no sistema decimal ■ Determinar o peso de cada dígito em um número decimal.

O sistema de numeração decimal tem dez dígitos.

No sistema de numeração **decimal**, cada um dos dígitos, de 0 a 9, representa uma certa quantidade. Como sabemos, os dez símbolos (**dígitos**) não nos limita a expressar apenas dez quantidades diferentes porque usamos vários dígitos posicionados adequadamente formando um número para indicar a magnitude (módulo) da quantidade. Podemos expressar quantidades até nove antes de usar mais dígitos; se queremos expressar uma quantidade maior que nove, usamos dois ou mais dígitos, sendo que a posição de cada dígito dentro do número nos diz a magnitude que ele representa. Se, por exemplo, queremos expressar a quantidade vinte e três, usamos (pela suas respectivas posições no número) o dígito 2 para representar a quantidade vinte e o dígito 3 para representar a quantidade três, conforme ilustrado a seguir.



O sistema de numeração decimal tem uma base 10.

A posição de cada dígito em um número decimal indica a magnitude da quantidade representada e pode ser associada a um **peso**. Os pesos para os números inteiros são potências de dez positivas que aumentam da direita para a esquerda, começando com $10^0 = 1$.

$$\ldots\ 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$$

Para números fracionários, os pesos são potências de dez negativas que diminuem da esquerda para a direita começando com 10^{-1}.

O valor de um dígito é determinado por sua posição no número.

$$10^2\ 10^1\ 10^0,10^{-1}\ 10^{-2}\ 10^{-3}\ldots$$

vírgula decimal

O valor de um número decimal é a soma dos dígitos após cada um ser multiplicado pelo seu peso, conforme ilustra os Exemplos 2–1 e 2–2.

EXEMPLO 2–1

Expresse o número decimal 47 como uma soma dos valores de cada dígito.

Solução O dígito 4 tem um peso de 10, que é 10^1, conforme indicado pela sua posição. O dígito 7 tem um peso de 1, que é 10^0, conforme indicado pela sua posição.

$$\begin{aligned} 47 &= (4 \times 10^1) + (7 \times 10^0) \\ &= (4 \times 10) + (7 \times 1) = \mathbf{40 + 7} \end{aligned}$$

Problema relacionado* Determine o valor de cada dígito em 939.

* As respostas estão no final do capítulo.

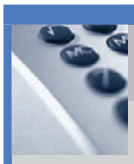
EXEMPLO 2–2

Expresse o número decimal 568,23 como uma soma dos valores de cada dígito.

Solução O dígito 5, na parte inteira do número, tem um peso de 100, que é 10^2, o dígito 6 tem um peso de 10, que é 10^1, o dígito 8 tem um peso de 1, que é 10^0, o dígito fracionário 2 tem um peso de 0,1, que é 10^{-1}, e o dígito fracionário 3 tem um peso de 0,01, que é 10^{-2}.

$$\begin{aligned} 568{,}23 &= (5 \times 10^2) + (6 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (3 \times 10^{-2}) \\ &= (5 \times 100) + (6 \times 10) + (8 \times 1) + (2 \times 0{,}1) + (3 \times 0{,}01) \\ &= \mathbf{500 + 60 + 8 + 0{,}2 + 0{,}03} \end{aligned}$$

Problema relacionado Determine o valor de cada dígito em 67,924.



TUTORIAL: CALCULADORA

Potência de Dez

Exemplo Determine o valor de 10^3.

		10^x	
TI-86	**Passo 1**	2 ^	
	Passo 2	3	10 ^ 30 1000
	Passo 3	ENTER	
TI-36X	**Passo 1**	1 0 y^x	
	Passo 2	3 =	1000

SEÇÃO 2–1 REVISÃO

As respostas estão no final do capítulo.

1. Qual é o peso que o dígito 7 tem em cada um dos seguintes números?
   (a) 1370 **(b)** 6725 **(c)** 7051 **(d)** 58,72
2. Expresse cada um dos seguintes números decimais como uma soma dos produtos obtidos pela multiplicação de cada dígito pelo peso apropriado:
   (a) 51 **(b)** 137 **(c)** 1492 **(d)** 106,58

2-2 NÚMEROS BINÁRIOS

O sistema de numeração binário é uma outra forma de representar quantidades. Ele é menos complicado que o sistema decimal porque usa apenas dois dígitos. O sistema decimal com os seus dez dígitos é um sistema de base dez; o sistema binário com seus dois dígitos é um sistema de base dois. Os dois dígitos binários (bits) são 1 e 0. A posição de um 1 ou um 0 em um número binário indica o seu peso, ou valor dentro do número, assim como a posição de um dígito decimal determina o valor daquele dígito. Os pesos em um número binário são baseados em potência de dois.

Ao final do estudo desta seção você deverá ser capaz de:

■ Contar em binário ■ Determinar o maior número decimal que pode ser representado por um determinado número de bits ■ Converter um número binário em um número decimal

Contagem em Binário

O sistema de numeração binário faz uso de dois dígitos (bits).

Para aprender a contar no sistema binário, primeiro analise como contamos no sistema decimal. Começamos pelo zero e contamos de forma crescente até nove antes de acabar os dígitos. Então começamos com o dígito de outra posição (à esquerda) e continuamos a contagem de 10 até 99. Neste momento, esgotamos todas as combinações de dois dígitos, sendo necessário um terceiro dígito para contar de 100 a 999.

Uma situação semelhante ocorre quando contamos em binário, exceto que temos apenas dois dígitos, denominados *bits*. Começando a contagem: 0, 1. Nesse momento, usamos os dois dígitos, assim incluímos uma nova posição de dígito e continuamos: 10, 11. Esgotamos todas as combinações de dois dígitos, de forma que é necessário uma terceira posição. Com posições para três dígitos podemos continuar a contagem: 100, 101, 110 e 111. Agora precisamos de uma quarta posição de dígito para continuar, e assim por diante. A Tabela 2–1 mostra uma contagem binária de zero a quinze. Observe o padrão de alternância de 1s e 0s em cada coluna.

O sistema de numeração binário tem base 2.

▶ **TABELA 2–1**

NÚMERO DECIMAL	NÚMERO BINÁRIO			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Como vemos na Tabela 2–1, são necessários 4 bits para contar de zero a 15. Em geral, com n bits podemos contar até um número igual a $2^n - 1$.

O valor de um bit é determinado pela sua posição no número.

Maior número decimal = $2^n - 1$

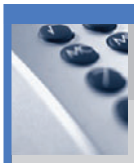
Por exemplo, com cinco bits ($n = 5$) podemos contar de zero a trinta e um.

$2^5 - 1 = 32 - 1 = 31$

Com seis bits ($n = 6$) podemos contar de zero a sessenta e três.

$2^6 - 1 = 64 - 1 = 63$

Uma tabela de potências de dois é dada no Apêndice A.



TUTORIAL: CALCULADORA

Potência de Dois

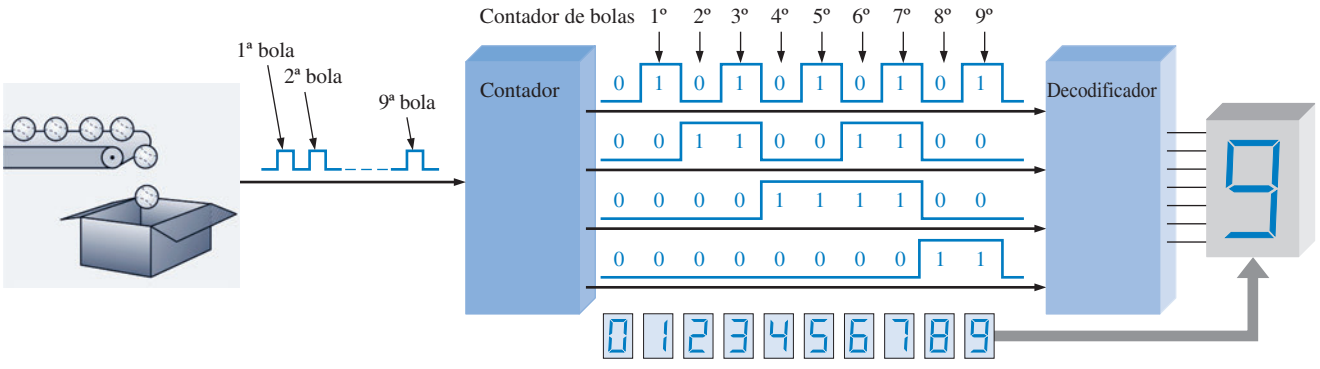
Exemplo Determine o valor de 2^5.

TI-86	**Passo 1**	2 ^	2 ^ 5
	Passo 2	5 ENTER	32
TI-36X	**Passo 1**	2 y^x	
	Passo 2	5 =	32

Uma Aplicação

Ao aprender a contar em binário, entenderemos basicamente como os circuitos digitais podem ser usados para contar eventos. Isso pode ser qualquer coisa, desde a contagem de itens em uma linha de montagem até a contagem de operações em um computador. Vamos considerar um exemplo simples da contagem de bola de tênis colocadas em uma caixa a partir de uma correia transportadora. Considere que são colocadas nove bolas em cada caixa.

O contador ilustrado na Figura 2–1 conta os pulsos de um sensor que detecta a passagem de uma bola e gera uma seqüência de níveis lógicos (formas de onda digitais) em cada uma das suas quatro saídas paralelas. Cada conjunto de níveis lógicos representa um número binário de 4 bits (nível ALTO = 1 e nível BAIXO = 0), conforme indicado. À medida que o decodificador recebe essas formas de onda, ele decodifica cada conjunto de quatro bits e converte no número decimal correspondente e mostra num display de 7 segmentos. Quando o contador chega no estado binário 1001, é porque ele contou 9 bolas de tênis, o display mostra o decimal 9 e uma nova caixa é posicionada sob o transportador. Então o contador retorna ao estado zero (0000) e o processo começa novamente. (O número 9 foi usado apenas com o intuito de se ter a simplicidade de um único dígito.)



▲ **FIGURA 2–1**

Ilustração de uma simples aplicação de contagem binária.

A Estrutura de Pesos dos Números Binários

O peso ou o valor de um bit aumenta da direita para a esquerda em um número binário.

Um número binário é um número em que os dígitos apresentam pesos. O bit mais à direita é o bit menos significativo (**LSB** – *least significant bit*) em um número inteiro binário e tem um peso de $2^0 = 1$. Os pesos aumentam da direita para a esquerda em potências de dois para cada bit. O bit mais à esquerda é o mais significativo (**MSB** – *most significant bit*); seu peso depende do tamanho do número binário.

Números fracionários também podem ser representados em binário colocando os bits à direita da vírgula binária, assim como os dígitos decimais fracionários são colocados à direita da vírgula decimal. O bit mais à esquerda é o MSB em um número binário fracionário e tem um peso de $2^{-1} = 0{,}5$. Os pesos da parte fracionária diminuem da esquerda para a direita por uma potência negativa de dois para cada bit.

A estrutura de pesos de um número binário é a seguinte:

$$2^{n-1} \ldots 2^3\ 2^2\ 2^1\ 2^0,\ 2^{-1}\ 2^{-2} \ldots 2^{-n}$$

vírgula binária

onde n é o número de bits a partir da vírgula binária. Portanto, todos os bits à esquerda da vírgula binária têm pesos que são potências positivas de dois, conforme discutido anteriormente para números inteiros. Todos os bits à direita da vírgula binária têm pesos que são potências negativas de dois, ou pesos fracionários.

As potências de dois e os seus pesos decimais equivalentes para um número inteiro binário de 8 bits e um número fracionário binário de 6 bits são mostradas na Tabela 2–2. Observe que o peso dobra para cada potência de dois positiva e que o peso é reduzido à metade a cada potência de dois negativa. Podemos estender facilmente a tabela dobrando o peso da potência de dois positiva mais significativa e reduzindo pela metade o peso da potência de dois negativa menos significativa, por exemplo, $2^9 = 512$ e $2^{-7} = 0{,}00787125$.

NOTA: COMPUTAÇÃO

Os computadores usam números binários para selecionar posições memória. Cada posição de memória está associada a um único número denominado *endereço*. Alguns microprocessadores Pentium, por exemplo, têm 32 linhas de endereço, as quais possibilitam selecionar (endereçar) 2^{32} (4.294.967.296) posições únicas de memória.

TABELA 2–1

Pesos binários

POTÊNCIAS DE DOIS POSITIVAS NÚMEROS INTEIROS									POTÊNCIAS DE DOIS NEGATIVAS NÚMEROS FRACIONÁRIOS					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0,5	0,25	0,125	0,0625	0,03125	0,015625

Conversão de Binário para Decimal

A soma dos pesos de todos os 1s de um número binário resulta no valor decimal.

O valor decimal de um número binário pode ser determinado somando-se os pesos de todos os bits que são 1 e descartando todos os pesos dos bits que são 0.

EXEMPLO 2–3

Converta o número binário inteiro 1101101 para decimal.

Solução Determine o peso de cada bit que for 1 e em seguida calcule o somatório desses pesos para obter o número decimal.

$$\begin{aligned} \text{Peso: } & 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 \\ \text{Número binário: } & 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1 \\ 1101101 &= 2^6 + 2^5 + 2^3 + 2^2 + 2^0 \\ &= 64 + 32 + 8 + 4 + 1 = \mathbf{109} \end{aligned}$$

Problema relacionado Converta o número binário 10010001 para decimal.

EXEMPLO 2–4

Converta o número binário fracionário 0,1011 para decimal.

Solução Determine o peso de cada bit que vale 1 e então some os pesos para obter o número decimal fracionário.

$$\begin{aligned}\text{Peso:} \quad & 2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \\ \text{Número binário: } 0, \; & 1 \quad 0 \quad 1 \quad 1 \\ 0{,}1011 &= 2^{-1} + 2^{-3} + 2^{-4} \\ &= 0{,}5 + 0{,}125 + 0{,}0625 = \mathbf{0{,}6875}\end{aligned}$$

Problema relacionado Converta o número binário 10,111 para decimal.

SEÇÃO 2–2 REVISÃO

1. Qual é o maior número decimal que pode ser representado em binário por 8 bits?
2. Determine o peso do bit 1 no número binário 10000.
3. Converta o número binário 10111101,011 para decimal.

2-3 CONVERSÃO DE DECIMAL PARA BINÁRIO

Na Seção 2–2, aprendemos como converter um número binário para o seu equivalente decimal. Agora vamos estudar duas formas de converter um número decimal em binário.

Ao final do estudo desta seção você deverá ser capaz de:

■ Converter um número decimal para binário usando o método da soma dos pesos ■ Converter um número inteiro decimal para binário usando o método da divisão sucessiva por dois ■ Converter um número fracionário decimal para binário usando o método da multiplicação sucessiva por dois

Método da Soma dos Pesos

Para obter um número binário a partir de um número decimal dado, determine os pesos que somados resultam no número decimal.

Uma forma de determinar o número binário que é equivalente a um dado número decimal é determinar o conjunto dos pesos binários cuja soma é igual ao número decimal. Um jeito fácil de lembrar dos pesos binários é saber que o menor dos pesos é 1, que corresponde a 2^0, e que dobrando esse peso obtemos o próximo peso de maior ordem; assim, uma lista de sete pesos em binário consta os pesos 64, 32, 16, 8, 4, 2, 1 conforme aprendido na seção anterior. O número decimal 9, por exemplo, pode ser expresso como a soma dos pesos binários mostrados a seguir:

$$9 = 8 + 1 \quad \text{ou} \quad 9 = 2^3 + 2^0$$

Colocando 1s nas posições apropriadas, 2^3 e 2^0, e 0s na posições 2^2 e 2^1, determina-se o número binário para o decimal 9.

2^3	2^2	2^1	2^0	
1	0	0	1	Número binário para o decimal 9

EXEMPLO 2–5

Converta os seguintes números decimais para binário:

(a) 12 **(b)** 25 **(c)** 58 **(d)** 82

Solução

(a) $12 = 8 + 4 = 2^3 + 2^2 \longrightarrow$ **1100**

(b) $25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 \longrightarrow$ **11001**

(c) $58 = 32 + 16 + 8 + 2 = 2^5 + 2^4 + 2^3 + 2^1 \longrightarrow$ **111010**

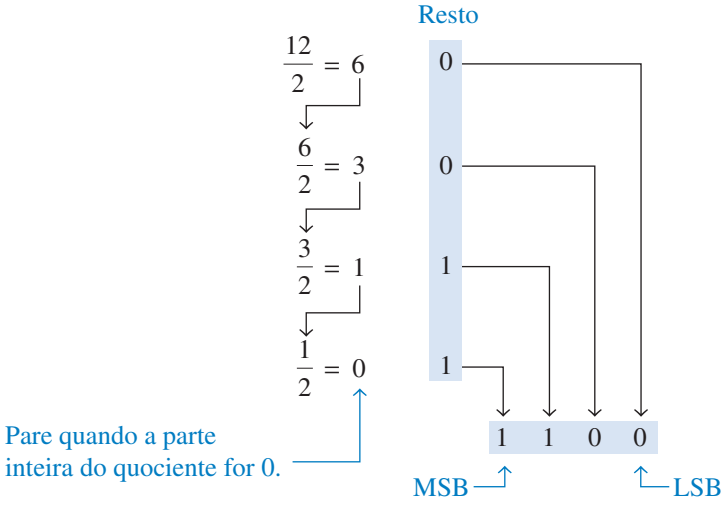
(d) $82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 \longrightarrow$ **1010010**

Problema relacionado Converta o número decimal 125 para binário.

Método da Divisão Sucessiva por 2

Para obter o número binário que corresponde a um dado número decimal, divida o número decimal por 2 até que o quociente seja 0. Os restos formam o número binário.

Um método sistemático de conversão de números decimais inteiros para o formato binário é o processo de *divisões sucessivas por 2*. Por exemplo, para converter o número decimal 12 para binário, comece dividindo 12 por 2. Em seguida divida cada quociente resultante por 2 até que a parte inteira do quociente seja 0. Os **restos** gerados em cada divisão formam o número binário. O primeiro resto gerado é o LSB (bit menos significativo) no número binário e o último resto gerado é o MSB (bit mais significativo). Esse procedimento é mostrado nos passos a seguir para converter o número decimal 12 em binário.

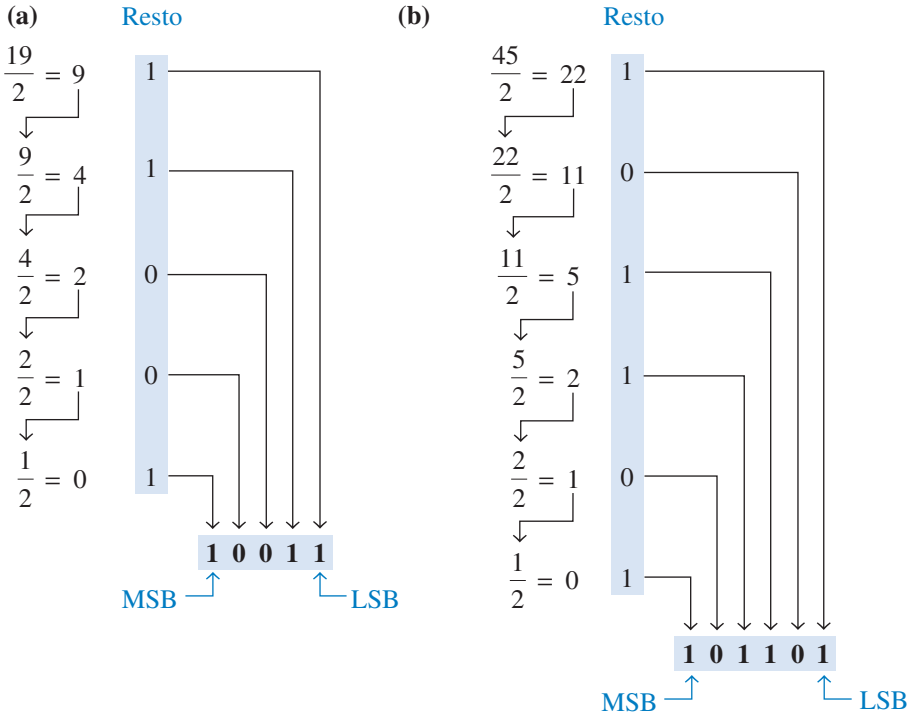


EXEMPLO 2–6

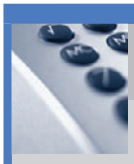
Converta os seguintes números decimais em binário:

(a) 19 **(b)** 45

Solução



Problema relacionado Converta o número decimal 39 em binário.

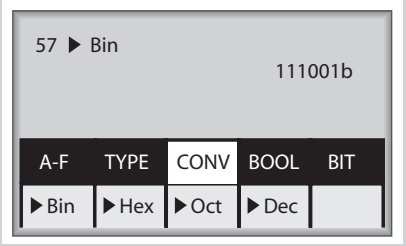


TUTORIAL: CALCULADORA

Conversão de um número decimal em número binário

Exemplo Converta o decimal 57 em binário.

		BASE
TI-86	**Passo 1**	2 1 F3
	Passo 2	5 7
	Passo 3	F1
	Passo 4	ENTER



		BASE
TI-36X	**Passo 1**	3rd EE
	Passo 2	5 7
		BIN
	Passo 3	3rd X

111001

Conversão de Decimal Fracionário em Binário

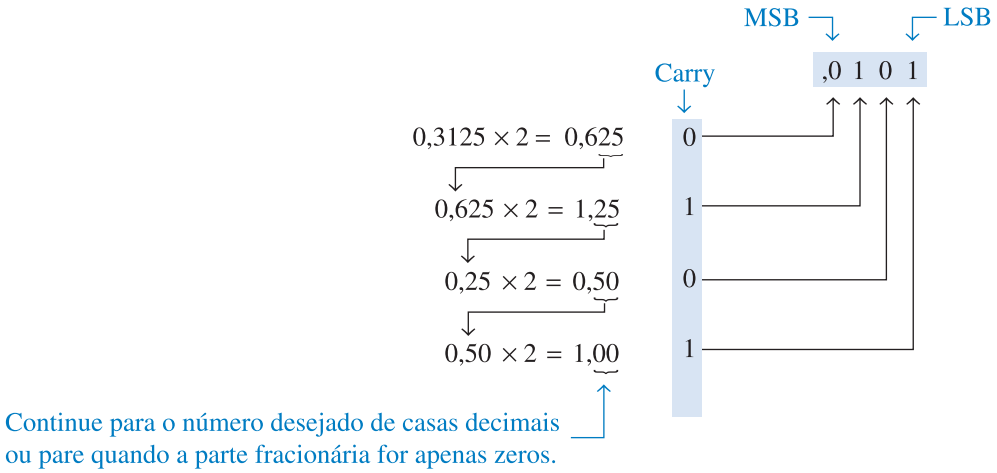
Os Exemplos 2–5 e 2–6 demonstraram conversões de números inteiros. Agora vamos analisar conversões entre números fracionários. Uma forma fácil de lembrar dos pesos da parte fracionária de um número binário é lembrar que o peso do bit mais significativo é 0,5, que equivale a 2^{-1}, e que dividindo qualquer peso por dois obtemos o próximo peso menos significativo; portanto, uma lista de quatro pesos binários fracionários seria 0,5; 0,25; 0,125; 0,0625.

Soma dos Pesos O método da soma dos pesos pode ser aplicado a números decimais fracionários, conforme mostra o exemplo a seguir:

$$0{,}625 = 0{,}5 + 0{,}125 = 2^{-1} + 2^{-3} = 0{,}101$$

Existe um 1 na posição 2^{-1}, um 0 na posição 2^{-2} e um 1 na posição 2^{-3}.

Multiplicações Sucessivas por 2 Como vimos, números decimais inteiros podem ser convertidos para binário por meio de divisões sucessivas por 2. Decimais fracionários podem ser convertidos para binário por meio de multiplicações sucessivas por 2. Por exemplo, para converter o fracionário decimal 0,3125 para binário, comece multiplicando 0,3125 por 2 e então multiplicar por 2 cada parte fracionária resultante do produto até que o produto seja 0 ou até que o número desejado de casas decimais seja alcançado. Os dígitos de carry, ou **carries**, gerados pela multiplicação formam o número binário. O primeiro carry gerado é o MSB e o último é o LSB. Esse procedimento é ilustrado a seguir:



SEÇÃO 2–3 REVISÃO

1. Converta cada número decimal a seguir em binário usando o método da soma dos pesos.
   (a) 23 **(b)** 57 **(c)** 45,5
2. Converta cada número decimal a seguir em binário usando o método das divisões sucessivas por 2 (multiplicações sucessivas por 2 no caso da parte fracionária):
   (a) 14 **(b)** 21 **(c)** 0,375

2-4 ARITMÉTICA BINÁRIA

A aritmética binária é essencial em todos os computadores digitais e em muitos outros tipos de sistemas digitais. Para entender os sistemas digitais, temos que saber os fundamentos das operações de soma, subtração, multiplicação e divisão em binário. Esta seção apresenta uma introdução ao assunto, que será estendido em seções posteriores.

Ao final do estudo desta seção você deverá ser capaz de:

■ Somar números em binário ■ Subtrair números em binário ■ Multiplicar números em binário ■ Dividir números em binário

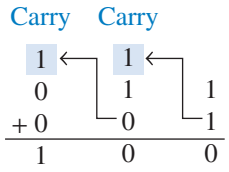
Adição Binária

As quatro regras básicas para a adição de dígitos binários (bits) são:

Lembre-se de que, em binário, 1 + 1 = 10, e não 2.

$0 + 0 = 0$ Resultado: 0; carry: 0
$0 + 1 = 1$ Resultado: 1; carry: 0
$1 + 0 = 1$ Resultado: 1; carry: 0
$1 + 1 = 10$ Resultado: 0; carry: 1

Observe que nas primeiras três regras a soma resulta em um único bit e na quarta regra a soma de dois 1s resulta no número binário dois (10). Quando números binários são somados, o último caso acima gera um resultado 0 em uma dada coluna e um carry de 1 para a próxima coluna à esquerda, conforme ilustrado na adição a seguir (11 + 1):



Na coluna da direita, 1 + 1 = 0 com um carry de 1 para a próxima coluna à esquerda. Na coluna do meio, 1 + 1 + 0 = 0 com um carry de 1 para a próxima coluna à esquerda. Na última coluna, 1 + 0 + 0 = 1.

Quando existe um carry de 1, temos uma situação na qual três bits estão sendo somados (um bit de cada um dos dois números e um bit de carry). Essa situação é ilustrada a seguir:

$1 + 0 + 0 = 01$ Resultado: 1; carry: 0
$1 + 1 + 0 = 10$ Resultado: 0; carry: 1
$1 + 0 + 1 = 10$ Resultado: 0; carry: 1
$1 + 1 + 1 = 11$ Resultado: 1; carry: 1

EXEMPLO 2–7

Efetue as seguintes adições de números binários:

(a) 11 + 11 **(b)** 100 + 10 **(c)** 111 + 11 **(d)** 110 + 100

Solução A soma decimal equivalente também é mostrada para referência.

(a)	11	3	**(b)**	100	4	**(c)**	111	7	**(d)**	110	6
	+11	+3		+10	+2		+ 11	+3		+100	+4
	110	6		**110**	6		**1010**	10		**1010**	10

Problema relacionado Some 1111 com 1100.

Subtração Binária

As quatro regras básicas para a subtração de bits são:

Lembre-se de que, em binário, 10 – 1 = 1, e não 9.

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$10 - 1 = 1$ sendo o empréstimo igual a 1

Quando subtraímos números, às vezes temos que fazer um empréstimo (borrow) da próxima coluna à esquerda. Em binário um borrow é necessário apenas quando tentamos subtrair 1 de 0. Nesse caso, quando um 1 é obtido como empréstimo da próxima coluna à esquerda, um 10 é criado na coluna que está ocorrendo a subtração e a última das quatro regras básicas apresentadas acima tem que ser aplicada. Os Exemplos 2–8 e 2–9 ilustram a subtração binária: as subtrações decimais equivalentes também são mostradas.

EXEMPLO 2–8

Efetue as seguintes subtrações binárias:

(a) 11 + 01 **(b)** 11 – 10

Solução

(a)	11	3	**(b)**	11	3
	− 01	− 1		− 10	− 2
	10	2		**01**	1

Nenhum empréstimo foi solicitado neste exemplo. O número binário 01 é o mesmo que 1.

Problema relacionado Efetue a subtração: 100 – 111.

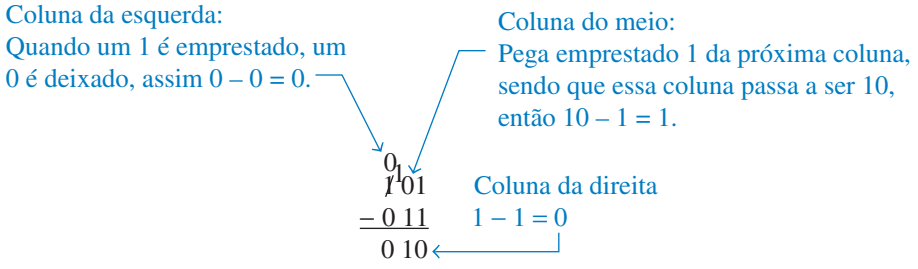
EXEMPLO 2–9

Efetue a subtração de 011 a partir de 101.

Solução

101	5
−011	−3
010	2

Vamos analisar exatamente o que foi feito para subtrair os dois números binários visto que um empréstimo foi solicitado. Comece pela coluna à direita.



Problema relacionado Efetue a subtração: 101 – 110.

Multiplicação Binária

A multiplicação binária de dois bits é realizada da mesma forma que na multiplicação dos dígitos decimais 0 e 1.

As quatro regras básicas para a multiplicação de bits são:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

A multiplicação é realizada com números binários da mesma maneira que com números decimais. Ela envolve a formação de produtos parciais, deslocamento de cada produto parcial sucessivo uma posição à esquerda, para então somar todos os produtos parciais. O Exemplo 2–10 ilustra o procedimento; as multiplicações decimais equivalentes são mostradas para referência.

EXEMPLO 2–10

Realize as seguintes multiplicações binárias:

(a) 11×11 **(b)** 101×111

Solução

(a)

$$\begin{array}{r} 11 \\ \times\ 11 \\ \hline 11 \\ +11 \\ \hline \mathbf{1001} \end{array} \quad \begin{array}{l} \\ \\ \text{Produtos parciais} \\ \\ \end{array} \qquad \begin{array}{r} 3 \\ \times\ 3 \\ \hline 9 \end{array}$$

(b)

$$\begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ 000 \\ +111 \\ \hline \mathbf{100011} \end{array} \quad \begin{array}{l} \\ \\ \text{Produtos parciais} \\ \\ \\ \end{array} \qquad \begin{array}{r} 7 \\ \times\ 5 \\ \hline 35 \end{array}$$

Problema relacionado Efetue a multiplicação: 1011×1010.

Divisão Binária

A divisão binária segue os mesmos procedimentos que a divisão decimal, como ilustra o Exemplo 2–11. As divisões decimais equivalentes também são mostradas.

Uma calculadora pode ser usada para realizar operações aritméticas com números binários enquanto a capacidade da calculadora não for excedida.

EXEMPLO 2–4

Realize as seguintes divisões binárias:

(a) $110 \div 11$ **(b)** $110 \div 10$

Solução

(a)

$$\begin{array}{r} \mathbf{10} \\ 11\overline{)110} \\ 11 \\ \hline 000 \end{array} \qquad \begin{array}{r} 2 \\ 3\overline{)6} \\ 6 \\ \hline 0 \end{array}$$

(b)

$$\begin{array}{r} \mathbf{11} \\ 10\overline{)110} \\ 10 \\ \hline 10 \\ 10 \\ \hline 00 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ 6 \\ \hline 0 \end{array}$$

Problema relacionado Efetue a divisão: $1100 \div 100$.

SEÇÃO 2-4 REVISÃO

1. Realize as seguintes adições binárias:
   (a) 1101 + 1010 **(b)** 10111 + 01101
2. Realize as seguintes subtrações binárias:
   (a) 1101 – 0100 **(b)** 1001 – 0111
3. Realize as operações binárias indicadas:
   (a) 110×111 **(b)** $1100 \div 011$

2-5 COMPLEMENTOS DE 1 E DE 2 DE NÚMEROS BINÁRIOS

O complemento de 1 e o complemento de 2 de um número binário são importantes porque eles permitem a representação de números negativos. O método da aritmética do complemento de 2 é geralmente usado em computadores na operação com números negativos.

Ao final do estudo desta seção você deverá ser capaz de:

- Converter um número binário no seu complemento de 1
- Converter um número binário no seu complemento de 2 usando os dois métodos

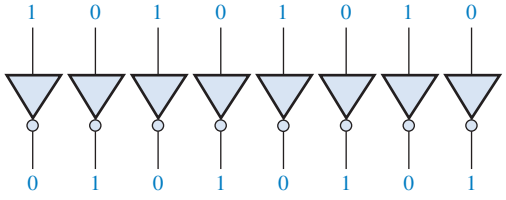
Determinação do Complemento de 1

Troque cada bit no número pelo seu complemento de 1.

O **complemento** de 1 de um número binário é determinado trocando-se todos os 1s por 0s e todos os 0s por 1s, conforme ilustrado a seguir:

1	0	1	1	0	0	1	0	Número binário
↓	↓	↓	↓	↓	↓	↓	↓	
0	1	0	0	1	1	0	1	Complemento de 1

A forma mais simples de obter o complemento de 1 de um número binário com um circuito digital é usar inversores em paralelo (circuitos NOT), conforme mostra a Figura 2–2 para um número binário de 8 bits.



▶ **FIGURA 2–2**
Exemplo do uso de inversores para obter o complemento de 1 de um número binário.

Determinação do Complemento de 2

Some 1 ao complemento de 1 para obter o complemento de 2.

O complemento de 2 de um número binário é determinado somando 1 ao LSB do complemento de 1.

complemento de 2 = (complemento de 1) + 1

EXEMPLO 2–12

Determine o complemento de 2 de 10110010.

Solução

10110010	Número binário
01001101	Complemento de 1
+ 1	Soma-se 1
01001110	Complemento de 2

Problema relacionado Determine o complemento de 2 de 11001011.

Um método alternativo para determinar o complemento de 2 de um número binário é:

1. Comece à direita com o LSB e escreva os bits como eles aparecem até o primeiro 1 (inclusive).
2. Tome o complemento de 1 dos bits restantes.

Troque todos os bits à esquerda do bit 1 menos significativo para obter o complemento de 2.

EXEMPLO 2–13

Determine o complemento de 2 de 10111000 usando o método alternativo.

Solução

10111000 Número binário

Complemento de 1 dos bits originais → **01001000** Complemento de 2

Estes bits permanecem os mesmos

Problema relacionado Determine o complemento de 2 de 11000000.

O complemento de 2 de um número binário negativo pode ser obtido usando inversores e um somador, conforme indicado na Figura 2–3. Essa figura ilustra como um número de 8 bits pode ser convertido no seu complemento de 2 invertendo primeiro cada bit (tomando o complemento de 1) e em seguida somando 1 ao complemento de 1 com um circuito somador.

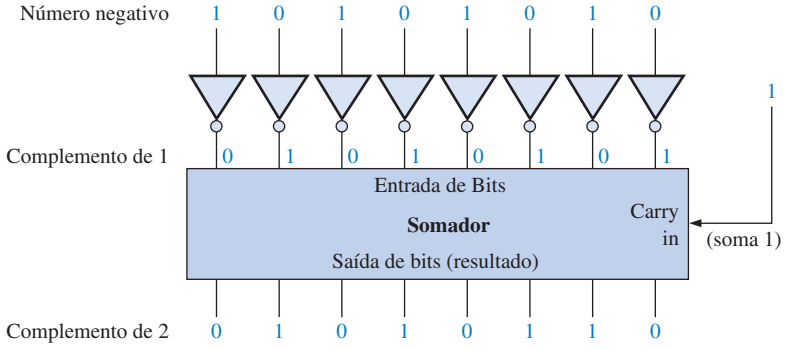


FIGURA 2–3

Exemplo da obtenção do complemento de 2 de número binário negativo.

Para converter a partir do complemento de 1 ou de 2 de volta para a forma binária verdadeira (não complementada), usamos os mesmos dois procedimentos descritos anteriormente. Para passar do complemento de 1 de volta para o binário verdadeiro, inverta todos os bits. Para passar do complemento de 2 de volta para a forma binária verdadeira, tome o complemento de 1 do número na forma do complemento de 2 e some 1 ao bit menos significativo.

SEÇÃO 2–5 REVISÃO

1. Determine o complemento de 1 e cada número binário a seguir:
   (a) 00011010 **(b)** 11110111 **(c)** 10001101
2. Determine o complemento de 2 de cada número binário a seguir:
   (a) 00010110 **(b)** 11111100 **(c)** 10010001

2-6 NÚMEROS SINALIZADOS

Os sistemas digitais, como o computador, têm que ser capazes de operar com números positivos e negativos. Um número binário sinalizado é constituído de duas informações: sinal e magnitude. O sinal indica se um número é positivo ou negativo e a magnitude é o valor do número. Existem três formas por meio das quais os números inteiros podem ser representados em binário: sinal-magnitude, complemento de 1 e complemento de 2. Dentre esses, a forma do complemento de 2 é a mais importante e a forma sinal-magnitude é a menos usada. Os números fracionários (não-inteiros) e muito grandes ou muito pequenos podem ser expressos na forma de ponto flutuante.

Ao final do estudo desta seção você deverá ser capaz de:

■ Expressar números positivos e negativos na forma sinal-magnitude ■ Expressar números positivos e negativos na forma do complemento de 1 ■ Expressar números positivos e negativos na forma do complemento de 2 ■ Determinar o valor decimal de números binários sinalizados ■ Expressar um número binário na forma de ponto flutuante

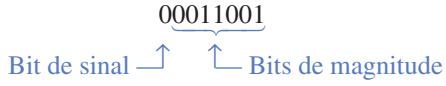
O Bit de Sinal

O bit mais à esquerda em um número binário sinalizado é o **bit de sinal**, o qual nos diz se o número é positivo ou negativo.

Um bit de sinal 0 indica um número positivo e um bit de sinal 1 indica um número negativo.

Forma Sinal-Magnitude

Quando um número binário sinalizado é representado na forma sinal-magnitude, o bit mais à esquerda é o bit de sinal e os bits restantes são os bits de magnitude. Os bits de magnitude estão na forma de binário verdadeiro (não-complementado) tanto para números positivos quanto para negativos. Por exemplo, o número decimal +25 é expresso como um número binário sinalizado de 8 bits usando a forma sinal-magnitude como a seguir:



O número decimal –25 é expresso como

10011001

Observe que a diferença entre +25 e –25 é apenas o bit de sinal porque os bits de magnitude estão na forma de binário verdadeiro tanto para números positivos quanto negativos.

Na forma sinal-magnitude, um número negativo tem os mesmos bits de magnitude como o número positivo correspondente mas o bit de sinal é 1 em vez de zero.

Forma do Complemento de 1

Números positivos na forma do complemento de 1 são representados da mesma forma que números positivos expressos como sinal-magnitude. Entretanto, os números negativos estão na forma do complemento de 1 do número positivo correspondente. Por exemplo, usando oito bits, o número decimal –25 é expresso como o complemento de 1 de +25 (00011001) como a seguir:

11100110

Na forma do complemento de 1, um número negativo é o complemento de 1 do número positivo correspondente.

NOTA: COMPUTAÇÃO

Os computadores usam a representação em complemento de 2 para os números inteiros negativos em todas as operações aritméticas. A razão para isso é que a subtração de um número é o mesmo que a adição com o complemento de 2 do número. Os computadores geram o complemento de 2 invertendo os bits e somando 1, usando instruções especiais que geram o mesmo resultado que o somador visto na Figura 2–3.

Forma do Complemento de 2

Os números positivos na forma do complemento de 2 são expressos da mesma forma que as representações sinal-magnitude e complemento de 1. Os números negativos são expressos em complemento de 2 dos números positivos correspondentes. Exemplificando novamente, usando 8 bits, vamos tomar o número decimal –25 e expressá-lo como complemento de 2 de +25 (00011001).

11100111

Na forma do complemento de 2, um número negativo é o complemento de 2 do correspondente número positivo.

EXEMPLO 2–14

Expresse o número decimal –39 como um número de 8 bits nas formas sinal-magnitude, complemento de 1 e complemento de 2.

Solução Primeiro escreva o número de 8 bits para +39.

00100111

Na *forma sinal-magnitude*, –39 é gerado alterando o bit de sinal para 1 e deixando os bits de magnitude como estavam. O número é

10100111

Na *forma do complemento de 1*, –39 é gerado tomando o complemento de 1 de +39 (00100111).

11011000

Na *forma do complemento de 2*, –39 é gerado tomando o complemento de 2 de +39 (00100111) como a seguir:

$$\begin{array}{rl} 11011000 & \text{Complemento de 1} \\ +\quad\ 1 & \\ \hline \mathbf{11011001} & \text{Complemento de 2} \end{array}$$

Problema relacionado Expresse +19 e –19 nas formas sinal-magnitude, complemento de 1 e complemento de 2.

O Valor Decimal de Números Sinalizados

Sinal-magnitude Os valores decimais de números positivos e negativos na forma sinal-magnitude são determinados somando os pesos de todos os bits de magnitude que são 1s e ignorando aqueles que são zeros. O sinal é determinado pela análise do bit de sinal.

EXEMPLO 2–15

Determine o valor decimal do número binário que vem a seguir expresso na forma sinal-magnitude: 10010101.

Solução Os sete bits de magnitude e os pesos em potências de dois são:

2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1

Somando os pesos dos bits que são 1s temos:

$$16 + 4 + 1 = 21$$

O bit de sinal é 1; portanto, o número decimal é –21.

Problema relacionado Determine o valor decimal do número 01110111 dado na forma sinal-magnitude.

Complemento de 1 Valores decimais de números positivos na forma do complemento de 1 são determinados somando os pesos de todos os bits 1s e ignorando os pesos relativos aos zeros. Os valores decimais de números negativos são determinados atribuindo um valor negativo ao peso do bit de sinal, somando os pesos relativos aos bits 1s e somando 1 ao resultado.

EXEMPLO 2–16

Determine os valores decimais dos números binários sinalizados expressos em complemento de 1:

(a) 00010111 **(b)** 11101000

Solução **(a)** Os bits e os respectivos pesos em potências de dois são:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	1	0	1	1	1

Somando os pesos correspondentes aos bits 1, temos:

$$16 + 4 + 2 + 1 = +\mathbf{23}$$

(b) Os bits e os respectivos pesos em potências de dois para o número negativo são mostrados a seguir. Observe que o bit de sinal negativo tem um peso de -2^7 ou –128.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	0	1	0	0	0

Somando os pesos em que os bits são 1s, temos:

$$-128 + 64 + 32 + 8 = -24$$

Somando 1 ao resultado, o número decimal final é

$$-24 + 1 = \mathbf{-23}$$

Problema relacionado Determine o valor decimal do número 11101011 expresso na forma do complemento de 1.

Complemento de 2 Valores decimais de números positivos e negativos na forma do complemento de 2 são determinados somando os pesos das posições de todos os bits 1s e ignorando as posições em que os bits são zeros. O peso do bit de sinal em números negativos é dado com um valor negativo.

EXEMPLO 2–17

Determine os valores decimais dos números binários sinalizados a seguir expressos na forma do complemento de 2:

(a) 01010110 **(b)** 10101010

Solução **(a)** Os bits e seus respectivos pesos em potências de dois para números positivos são:

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	1	0

Somando-se os pesos relativos aos bits 1s, temos:

$$64 + 16 + 4 + 2 = +\mathbf{86}$$

(b) Os bits e seus respectivos pesos em potências de dois para números positivos são os seguintes. Observe que o bit de sinal negativo tem um peso de -2^7 ou –128.

-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	0

Somando-se os pesos relativos aos bits 1s, temos:

$$-128 + 32 + 8 + 2 = \mathbf{-86}$$

Problema relacionado Determine o valor decimal do número 11010111 expresso na forma do complemento de 2.

A partir desses exemplos, podemos ver por que a forma do complemento de 2 é a preferida para representar números inteiros sinalizados: para converter para decimal, é necessário simplesmente somar os pesos independente se o número é positivo ou negativo. O sistema do complemento de 1 requer somar 1 ao resultado da soma dos pesos para números negativos, porém não para números positivos. Além disso, a forma do complemento de 1 não é muito usada porque existem duas representações possíveis para o zero (00000000 ou 11111111).

Faixa de Números Inteiros que Pode ser Representada

A faixa da magnitude de um número binário depende do número de bits(*n*).

Temos usado números de 8 bits para ilustração porque os grupos de 8 bits são comuns na maioria dos computadores, conhecidos como **byte**. Com um byte ou oito bits, podemos representar 256 números diferentes. Com dois bytes ou dezesseis bits, podemos representar 65.536 números diferentes. Com quatro bytes ou 32 bits, podemos representar $4{,}295 \times 10^9$ números diferentes. A fórmula para encontrar o número de combinações diferentes de *n* bits é

$$\text{Total de combinações} = 2^n$$

Para números sinalizados na forma do complemento de 2, a faixa de valores para números de *n* bits é

$$\text{Faixa} = -(2^{n-1}) \text{ a } +(2^{n-1} - 1)$$

onde existe em cada caso um bit de sinal e $n - 1$ bits de magnitude. Por exemplo, com quatro bits podemos representar números em complemento de 2 desde $-(2^3) = -8$ até $2^3 - 1 = +7$. De forma similar, com oito bits podemos representar desde –128 até +127, e com dezesseis bits podemos representar desde –32.768 até +32.767, e assim por diante.

Números em Ponto Flutuante

Para representar números **inteiros** muito grandes, são necessários muitos bits. Existe também um problema quando números que têm parte inteira e fracionária, como 23,5618, precisam ser representados. O sistema de numeração de ponto flutuante*, baseado em notação científica, é capaz de representar números muito grandes e muito pequenos sem o aumento do número de bits e também representa números que têm parte inteira e fracionária.

* N. de T.: A denominação ponto flutuante é mantida por ser de uso comum. O ponto em inglês equivale à vírgula em português. Então, o que vai se mover (flutuar) na representação numérica é a vírgula decimal.

Um **número em ponto flutuante** (também conhecido como *número real*) consiste em duas partes mais um sinal. A **mantissa** é a parte do número em ponto flutuante que representa a magnitude do número. O **expoente** é a parte do número em ponto flutuante que representa o número de casas decimais que a vírgula decimal (ou vírgula binária) é movida.

Um exemplo com número decimal será útil para a compreensão dos conceitos básicos de números em ponto flutuante. Vamos considerar um número decimal que, na forma de inteiro, é 241.506.800. A mantissa dele é 0,2415068 e o expoente é 9. Quando um inteiro é expresso como número em ponto flutuante, ele é normalizado movendo-se a vírgula decimal para a esquerda de todos os dígitos de forma que a mantissa seja um número fracionário e o expoente seja uma potência de dez. O número em ponto flutuante é escrito como a seguir:

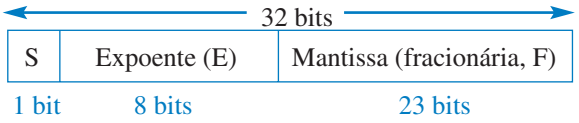
$$0{,}2415068 \times 10^{9}$$

Para números binários em ponto flutuante, o formato é definido pelo padrão 754-1985 da ANSI/IEEE de três formas: *precisão simples*, *precisão dupla* e *precisão estendida*. Todos têm o mesmo formato exceto pelo número de bits. Números em ponto flutuante de precisão simples têm 32 bits, números de precisão dupla têm 64 bits e números de precisão estendida têm 80 bits. Restringiremos nossa discussão ao formato de ponto flutuante de precisão simples.

NOTA: COMPUTAÇÃO

Além da unidade central de processamento (CPU – *central processing unit*), os computadores usam co-processadores para realizar operações matemáticas complexas usando números em ponto flutuante. A finalidade é aumentar a performance liberando a CPU para outras tarefas. O co-processador matemático também é conhecido como unidade de ponto flutuante (FPU – *floating-point unit*).

Números Binários de Ponto Flutuante de Precisão Simples No formato padrão para um número binário de precisão simples, o bit de sinal (S) é o bit mais à esquerda, o expoente (E) corresponde aos próximos 8 bits e a mantissa ou parte fracionária (F) inclui os 23 bits restantes, como mostrado a seguir.



Na mantissa ou parte fracionária, entende-se que a vírgula binária está à esquerda dos 23 bits. Efetivamente, existem 24 bits na mantissa porque em qualquer número binário o bit mais à esquerda (o mais significativo) é sempre um 1. Portanto, esse 1 está lá, embora não ocupe uma posição real de um bit.

Os oito bits do expoente representam um *expoente polarizado*, o qual é obtido acrescentando 127 ao expoente real. A finalidade da polarização é permitir números muito grandes ou muito pequenos sem a necessidade de um bit de sinal separado para os expoentes. O expoente polarizado permite uma faixa de valores de expoente de −126 até +128.

Para ilustrar como um número binário é expresso no formato de ponto flutuante, vamos usar o número 1011010010001 como exemplo. Primeiro, ele tem que ser expresso como 1 mais um número binário fracionário movendo a vírgula binária 12 posições para a esquerda e então multiplicar pela potência de dois apropriada.

$$1011010010001 = 1{,}011010010001 \times 2^{12}$$

Admitindo que esse número seja positivo, o bit de sinal (S) é 0. O expoente, 12, é expresso como um expoente polarizado somando-se 127 (12 + 127 = 139). O expoente polarizado (E) é expresso como o número binário 10001011. A mantissa é a parte fracionária (F) do número binário 0,011010010001. Como existe um 1 à esquerda da vírgula binária na expressão da potência de dois, não é incluído na mantissa. O número completo representado em ponto flutuante é:

S	E	F
0	10001011	01101001000100000000000

Em seguida, vamos ver como avaliar um número binário que já está no formato de ponto flutuante. A abordagem geral para determinar o valor de um número em ponto flutuante é expressa pela fórmula:

$$\text{Número} = (-1)^{S}(1 + F)(2^{E-127})$$

Para ilustrar, vamos considerar o seguinte número binário em ponto flutuante:

S	E	F
1	10010001	10001110001000000000000

O bit de sinal é 1. O expoente polarizado é 10010001 = 145. Aplicando a fórmula, obtemos:

$$\text{Número} = (-1)^1(1{,}10001110001)(2^{145-127})$$
$$= (-1)(1{,}10001110001)(2^{18}) = -1100011100010000000$$

Esse número binário em ponto flutuante é equivalente a –407.688 em decimal. Como o expoente pode ser qualquer número entre –126 e +128, pode-se expressar números extremamente grandes e pequenos. Um número de ponto flutuante de 32 bits pode substituir um número binário inteiro de 129 bits. Como o expoente determina a posição da vírgula binária, números que contém partes inteira e fracionária podem ser representados.

Existem duas exceções para números no formato de ponto flutuante: O número 0,0 é representado por 0s em todas as posições e infinito é representado 1s em todas as posições do expoente e por 0s em todas as posições da mantissa.

EXEMPLO 2–18

Converta o número decimal $3{,}248 \times 10^4$ para um número binário no formato de ponto flutuante de precisão simples.

Solução Converta o número decimal em binário.

$$3{,}248 \times 10^4 = 32480 = 111111011100000_2 = 1{,}11111011100000 \times 2^{14}$$

O MSB não ocupa a posição de um bit porque ele é sempre um 1. Portanto, a mantissa é o número binário fracionário de 23 bits 11111011100000000000000 e o expoente polarizado é

$$14 + 127 = 141 = 10001101_2$$

O número completo em ponto flutuante é

0	**10001101**	**11111011100000000000000**

Problema relacionado Determine o valor binário do seguinte número em ponto flutuante:

0 10011000 10000100010100110000000

SEÇÃO 2–6 REVISÃO

1. Expresse o número decimal +9 como um número binário de 8 bits no sistema sinal-magnitude.
2. Expresse o número decimal –33 como um número binário de 8 bits no sistema de complemento de 1.
3. Expresse o número decimal –46 como um número binário de 8 bits no sistema de complemento de 2.
4. Faça uma lista especificando as três partes de um número sinalizado no formato de ponto flutuante.

2-7 OPERAÇÕES ARITMÉTICAS COM NÚMEROS SINALIZADOS

Na seção anterior, aprendemos como os números sinalizados são representados de três formas diferentes. Nesta seção, estudaremos como os números sinalizados são somados, subtraídos, multiplicados e divididos. Devido à forma do complemento de 2 para representação de números sinalizados ser a mais usada em computadores e sistemas microprocessados, a abordagem nesta seção se limita a aritmética do complemento de 2. Os processos abordados podem ser estendidos a outros formatos, se necessário.

Ao final do estudo desta seção você deverá ser capaz de:

■ Somar números binários sinalizados ■ Explicar como os computadores somam seqüências numéricas ■ Definir *overflow* ■ Subtrair números binários sinalizados ■ Multiplicar números binários sinalizados usando o método da adição direta ■ Multiplicar números binários sinalizados usando o método dos produtos parciais ■ Dividir números binários sinalizados

Adição

Os dois números de uma adição são **1ª parcela** e **2ª parcela**. O resultado é a **soma**. Existem quatro casos que podem ocorrer quando dois números binários sinalizados são somados.

1. Os dois números são positivos
2. O número positivo com magnitude maior que o número negativo
3. O número negativo com magnitude maior que o número positivo
4. Os dois números são negativos

Vamos analisar um caso de cada vez usando números sinalizados de 8 bits como exemplos. Os números decimais equivalentes são mostrados para referência.

A adição de dois números positivos resulta em um número positivo.

Ambos os números são positivos

```
  00000111      7
+ 00000100    + 4
  00001011     11
```

A soma é positiva estando portanto em binário verdadeiro (não complementado).

A adição de um número positivo com um número negativo menor resulta em um número positivo.

Número positivo com magnitude maior que a do número negativo:

```
                00001111     15
              + 11111010   + −6
Carry descartado ⟶ 1 00001001    9
```

O bit de carry final é descartado. A soma é positiva e portanto é um binário verdadeiro (não complementado).

A adição de um número positivo com um número negativo maior, ou a adição de dois números negativos, resulta em um número negativo em complemento de 2.

Número negativo com magnitude maior que a do número positivo:

```
  00010000     16
+ 11101000   + −24
  11111000     −8
```

A soma é negativa e portanto na forma do complemento de 2.

Ambos os números são negativos:

```
                11111011     −5
              + 11110111   + −9
Carry descartado ⟶ 1 11110010   −14
```

O bit de carry final é descartado. A soma é negativa e portanto na forma do complemento de 2.

Em um computador, os números negativos são armazenados na forma do complemento de 2. Assim, como podemos ver, o processo de adição é muito simples: *somar os dois números e descartar o bit de carry final*.

Condição de Overflow Quando dois números são somados e o número de bits necessário para representar a soma excede o número de bits nos dois números, resulta em um **overflow** (transbordamento de capacidade) conforme indicado por um bit de sinal incorreto. Um overflow pode ocorrer apenas quando os dois números são positivos ou ambos negativos. O exemplo a seguir com números de 8 bits ilustra essa condição.

```
  01111101     125
+ 00111010   + 58
  10110111     183
```

Sinal incorreto
Magnitude incorreta

Nesse exemplo a soma de 183 requer oito bits de magnitude. Como existem sete bits de magnitude nos números (um bit é o sinal), existe um carry no lugar do bit de sinal que produz a indicação de overflow.

Os Números são Somado Dois de Cada Vez Agora vamos analisar a adição de uma seqüência de números, somados dois de cada vez. Essa operação pode ser realizada somando-se os dois primeiros números, somando em seguida o resultado ao terceiro número, somando outra vez o resultado ao quarto número e assim por diante. É assim que os computadores somam uma seqüência de números. A adição de números tomados dois de cada vez é ilustrada no Exemplo 2–19.

EXEMPLO 2–19

Some os seguintes números sinalizados: 01000100, 00011011, 00001110 e 00010010.

Solução As adições decimais equivalentes são dadas como referência.

68	01000100	
+ 27	+ 00011011	Soma dos dois primeiros números
95	01011111	1º subtotal
+ 14	+ 00001110	Soma do 3º número
109	01101101	2º subtotal
+ 18	+ 00010010	Soma do 4º número
127	**01111111**	Resultado final

Problema relacionado Some 00110011, 10111111 e 01100011. Esses números são sinalizados.

Subtração

A subtração é um caso especial da adição. Por exemplo, a subtração de +6 (o **subtraendo**) de +9 (o **minuendo**) é equivalente à soma de –6 com +9. Basicamente, *a operação de subtração troca o sinal do subtraendo e o soma ao minuendo*. O resultado da subtração é denominado de **diferença**.

A subtração é uma soma com o sinal do subtraendo trocado.

O sinal de um número binário positivo ou negativo é trocado tomando-se o complemento de 2 dele.

Por exemplo, quando se toma o complemento de 2 do número positivo 00000100 (+4), obtemos 11111100, que é –4, como mostra a análise da soma dos pesos a seguir:

$$-128 + 64 + 32 + 16 + 8 + 4 = -4$$

Em outro exemplo, quando tomamos o complemento de 2 do número negativo 11101101 (–19), obtemos 00010011, que é +19, conforme a análise da soma dos pesos a seguir:

$$16 + 2 + 1 = 19$$

Como a subtração é simples, uma adição com o subtraendo de sinal trocado, o processo é descrito da seguinte forma:

Para subtrair dois números sinalizados, tome o complemento de 2 do subtraendo e faça uma soma. Descarte qualquer bit de carry final.

O Exemplo 2-20 ilustra o processo de subtração.

EXEMPLO 2–20

Realize cada uma das seguintes subtrações de números sinalizados:

(a) 00001000 – 00000011 **(b)** 00001100 – 11110111

(c) 11100111 – 00010011 **(d)** 10001000 – 11100010

Solução Assim como em outros exemplos, as subtrações decimais equivalentes são dadas para referência.

(a) Neste caso, 8 – 3 = 8 + (–3) = 5.

```
                00001000    Minuendo (+8)
              + 11111101    Complemento de 2 do subtraendo (–3)
Carry descartado → 1 00000101   Diferença (+5)
```

(b) Neste caso, 12 – (–9) = 12 + 9 = 21

```
  00001100    Minuendo (+12)
+ 00001001    Complemento de 2 do subtraendo (+9)
  00010101    Diferença (+21)
```

(c) Neste caso, –25 – (+19) = –25 + (–19) = – 44

```
                11100111    Minuendo (–25)
              + 11101101    Complemento de 2 do subtraendo (–19)
Carry descartado → 1 11010100   Diferença (–44)
```

(d) Neste caso, –120 – (–30) = –120 + 30 = –90

```
  10001000    Minuendo (–120)
+ 00011110    Complemento de 2 do subtraendo (+30)
  10100110    Diferença (–90)
```

Problema relacionado Subtraia 01000111 de 01011000.

Multiplicação

Os termos de uma multiplicação são o **multiplicando**, o **multiplicador** e o **produto**. Eles são ilustrados na seguinte multiplicação decimal:

A multiplicação é equivalente à adição de um número com ele mesmo um número de vezes igual ao multiplicador.

```
  8    Multiplicando
× 3    Multiplicador
 24    Produto
```

A operação de multiplicação na maioria dos computadores é realizada usando adição. Como já vimos, a subtração é feita com um somador; agora, veremos como a multiplicação é feita.

Adição direta e *produtos parciais* são dois métodos básicos para realizarmos multiplicação usando adição. No método da adição direta, somamos o multiplicando um número de vezes igual ao multiplicador. No exemplo decimal anterior (3×8), três multiplicandos são somados: $8 + 8 + 8 = 24$. A desvantagem dessa abordagem é que ela se torna muito lenta se o multiplicador for um número grande. Por exemplo, para multiplicar 75×350, temos que somar 350 com ele mesmo 75 vezes. Aliás, esse é o motivo do termo vezes ser usado para significar multiplicação.

Quando dois números binários são multiplicados, os dois números têm que estar na forma verdadeira (não complementados). O método da adição direta é ilustrado no Exemplo 2–21 onde são somados dois números binários de cada vez.

EXEMPLO 2–21

Multiplique os seguintes números binários sinalizados: 01001101 (multiplicando) e 00000100 (multiplicador) usando o método da adição direta.

Solução Como os dois números são positivos, eles estão na forma verdadeira, sendo que o produto será positivo. O valor decimal do multiplicador é 4, de forma que o multiplicando deve ser somado com ele mesmo quatro vezes como mostrado a seguir:

01001101	1ª vez
+ 01001101	2ª vez
10011010	Resultado parcial
+ 01001101	3ª vez
11100111	Resultado parcial
+ 01001101	4ª vez
100110100	Produto

Como o bit de sinal do multiplicando é 0, ele não tem efeito no resultado. Todos os bits do produto são bits de magnitude.

Problema relacionado Multiplique 01100001 por 00000110 usando o método da adição direta.

O método do produto parcial talvez seja o mais comumente usado porque ele reflete a forma com que multiplicamos manualmente. O multiplicando é multiplicado por cada dígito do multiplicador começando pelo dígito menos significativo. O resultado da multiplicação do multiplicando por um dígito do multiplicador é denominado de *produto parcial*. Cada produto parcial sucessivo é movido (deslocado) uma posição para a esquerda e quando todos os produtos parciais são gerados, eles são somados para se obter o produto final. Eis um exemplo em decimal.

239	Multiplicando
× 123	Multiplicador
717	1º produto parcial (3×239)
478	2º produto parcial (2×239)
+ 239	3º produto parcial (1×239)
29.397	Produto final

O sinal do produto de uma multiplicação depende dos sinais do multiplicando e multiplicador de acordo com as seguintes regras:

- **Se os sinais são iguais, o produto é positivo.**
- **Se os sinais são diferentes, o produto é negativo.**

Os passos básicos no método dos produtos parciais da multiplicação binária são:

Passo 1 Determine se os sinais do multiplicando e multiplicador são iguais ou diferentes. Isso determina qual sinal o produto terá.

Passo 2 Troque qualquer número negativo para a forma verdadeira (não complementado). Como a maioria dos computadores armazena números negativos na forma do complemento de 2, uma operação de complemento de 2 é necessária para mudar um número negativo para a forma verdadeira.

Passo 3 Começando com o bit menos significativo do multiplicador, gere o produto parcial. Quando o bit do multiplicador for 1, o produto parcial é o mesmo que o multiplicando. Quando o bit do multiplicador for 0, o produto parcial é zero. Desloque cada produto parcial sucessivo um bit para a esquerda.

Passo 4 Some cada produto parcial sucessivo ao resultado da soma do produto parcial anterior até obter o produto final.

Passo 5 Se o bit de sinal que foi determinado no passo 1 for negativo, tome o complemento de 2 do produto. Caso seja positivo, deixe o produto na forma verdadeira. Acrescente o bit de sinal ao produto.

EXEMPLO 2–22

Multiplique os seguintes números binários sinalizados: 01010011 (multiplicando) e 11000101 (multiplicador).

Solução

Passo 1 O bit de sinal do multiplicando é 0 e o bit de sinal do multiplicador é 1. O bit de sinal do produto será 1 (negativo).

Passo 2 Obtenha o complemento de 2 para colocá-lo na forma verdadeira.

$$11000101 \rightarrow 00111011$$

Passo 3 e 4 Os procedimentos da multiplicação são registrados a seguir. Observe que apenas os bits de magnitude são usados nesses passos.

1010011	Multiplicando
× 0111011	Multiplicador
1010011	1º produto parcial
+ 1010011	2º produto parcial
11111001	Soma do 1º com o 2º
+ 0000000	3º produto parcial
011111001	Soma
+ 1010011	4º produto parcial
1110010001	Soma
+ 1010011	5º produto parcial
100011000001	Soma
+ 1010011	6º produto parcial
1001100100001	Soma
+ 0000000	7º produto parcial
1001100100001	Produto final

Passo 5 Como o sinal do produto é 1, conforme determinado no passo 1, obtenha o complemento de 2 do produto.

$$1001100100001 \longrightarrow 0110011011111$$

Acrescente o bit de sinal

$$\rightarrow \mathbf{1\ 0110011011111}$$

Problema relacionado Verifique se a multiplicação está correta convertendo para números decimais e realizando a multiplicação.

Divisão

Os termos de uma divisão são o **dividendo**, o **divisor** e o **quociente**. Eles são ilustrados no seguinte formato padrão de divisão.

$$\frac{\text{dividendo}}{\text{divisor}} = \text{quociente}$$

A operação de divisão nos computadores é realizada usando subtrações. Como as subtrações são feitas com um somador, a divisão também pode ser realizada com um somador.

O resultado de uma divisão é denominado *quociente*; o quociente é o número de vezes que o divisor 'cabe dentro' do dividendo. Isso significa que o divisor pode ser subtraído a partir do dividendo um número de vezes igual ao quociente, conforme ilustrado pela divisão de 21 por 7.

21	dividendo
− 7	1ª subtração do divisor
14	1º resto parcial
− 7	2ª subtração do divisor
7	2º resto parcial
− 7	3ª subtração do divisor
0	Resto zero

Nesse exemplo, o divisor foi subtraído do dividendo três vezes antes que o resto zero fosse obtido. Portanto, o quociente é 3.

O sinal do quociente depende dos sinais do dividendo e do divisor de acordo com as seguintes regras:

- **Se os sinais forem iguais, o quociente é positivo.**
- **Se os sinais forem diferentes, o quociente é negativo.**

Quando dois números binários são divididos, os dois números têm que estar na forma verdadeira (não complementados). Os passos básicos no processo de divisão são os seguintes:

Passo 1 Determine se os sinais do dividendo e do divisor são iguais ou diferentes. Isso determina o sinal do quociente. Inicialize o quociente com zero.

Passo 2 Subtraia o divisor a partir do dividendo usando a adição do complemento de 2 para obter o primeiro resto parcial e somar 1 ao quociente. Se esse resto parcial for positivo, vá para o passo 3. Caso o resto parcial seja zero ou negativo, a divisão está completa.

Passo 3 Subtraia o divisor a partir do dividendo e some 1 ao quociente. Se o resultado for positivo, repita a operação para o próximo resto parcial. Se o resultado for zero ou negativo, a divisão está completa.

Continue o divisor a partir do dividendo e os restos parciais até obter um resultado zero ou negativo. Conte o número de vezes que o divisor é subtraído e você terá o quociente. O Exemplo 2–23 ilustra esses passos usando números binários sinalizados de 8 bits.

EXEMPLO 2–23

Efetue a divisão de 01100100 por 00011001.

Solução **Passo 1** Os sinais dos dois números são positivos, de forma que o quociente será positivo. O quociente é inicializado em zero: 00000000.

Passo 2 Subtraia o divisor a partir do dividendo usando a adição do complemento de 2 (lembre-se que o carry final é descartado).

01100100	Dividendo
+ 11100111	Complemento de 2 do divisor
01001011	1º resto parcial positivo

Some 1 ao quociente: 00000000 + 00000001 = 00000001.

Passo 3 Subtraia o divisor do 1º resto parcial usando a adição do complemento de 2.

01001011	1º resto parcial
+ 11100111	Complemento de 2 do divisor
00110010	2º resto parcial positivo

Passo 4 Subtraia o divisor do 2º resto parcial usando a adição do complemento de 2.

00110010	2º resto parcial
+ 11100111	Complemento de 2 do divisor
00011001	3º resto parcial positivo

Some 1 ao quociente: 00000010 + 00000001 = 00000011.

Passo 5 Subtraia o divisor do 3º resto parcial usando a adição do complemento de 2.

00011001	3º resto parcial
+ 11100111	Complemento de 2 do divisor
00000000	Resto zero

Some 1 ao quociente: 00000011 + 00000001 = **00000100** (quociente final). O processo está completo.

Problema relacionado Verifique se o processo está correto convertendo para números decimais e realizando a divisão.

SEÇÃO 2–7 REVISÃO

1. Cite os quatro casos de sinalização quando os números são somados.
2. Some 00100001 e 10111100.
3. Subtraia 00110010 de 01110111.
4. Qual é o sinal do produto quando dois números negativos são multiplicados?
5. Multiplique 01111111 por 00000101.
6. Qual é o sinal do quociente quando um número positivo é dividido por um número negativo?
7. Divida 00110000 por 00001100.

2-8 NÚMEROS HEXADECIMAIS

O sistema de numeração hexadecimal tem dezesseis caracteres; ele é usado principalmente como uma forma compacta de apresentar ou escrever números binários, e é muito fácil realizar conversões entre binário e hexadecimal. Números binários longos são difíceis de serem lidos e escritos porque é fácil omitir ou trocar um bit. Como os computadores entendem apenas 1s e 0s, é necessário usar esses dígitos quando se programa em "linguagem de máquina". Imagine escrever uma instrução de dezesseis bits para um sistema microprocessado em 1s e 0s. É muito mais eficiente usar hexadecimal ou octal; os números octais são abordados na Seção 2–9. O sistema hexadecimal é bastante usado em aplicações de computador e microprocessador.

Ao final do estudo desta seção você deverá ser capaz de:

■ Fazer uma lista dos caracteres hexadecimais ■ Contar em hexadecimal ■ Converter de binário para hexadecimal ■ Converter de hexadecimal para binário ■ Converter de hexadecimal para decimal ■ Converter de decimal para hexadecimal ■ Somar números hexadecimais ■ Determinar o complemento de 2 de um número hexadecimal ■ Subtrair números hexadecimais

O sistema de numeração **hexadecimal** tem uma base de dezesseis; ou seja, ele é composto de 16 **caracteres numéricos** e alfabéticos. A maioria dos sistemas digitais processa dados binários em grupos que são múltiplos de quatro bits, tornando o número hexadecimal muito conveniente porque cada dígito hexadecimal representa um número binário de 4 bits (conforme vemos na Tabela 2–3).

O sistema de numeração hexadecimal consiste em dígitos de 0 a 9 e letras de A a F.

◀ TABELA 2–3

DECIMAL	BINÁRIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

O sistema de numeração hexadecimal é constituído de dez dígitos numéricos e seis caracteres alfabéticos. O uso das letras A, B, C, D, E e F para representar números pode inicialmente parecer estranho, mas tenha em mente que qualquer sistema de numeração é apenas um conjunto de símbolos seqüenciais. Se entendermos que quantidades esses símbolos representam, então a forma que os símbolos apresentam é menos importante uma vez que nos acostumamos a usá-los. Usaremos o subscrito 16 para designar números hexadecimais para evitar confusões com os números decimais. Algumas vezes veremos uma letra "h" seguida de um número hexadecimal.

NOTA: COMPUTAÇÃO

Com as memórias dos computadores na faixa de gigabytes (GB), especificar um endereço de memória em binário é bastante incômodo. Por exemplo, precisamos de 32 bits para especificar um endereço numa memória de 4 GB. É muito mais fácil expressar um código de 32 bits usando 8 dígitos hexadecimais.

Contagem em Hexadecimal

Como contar em hexadecimal uma vez atingida a contagem F? Simplesmente inicie uma nova coluna e continue como mostrado a seguir:

10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31...

Com dois dígitos hexadecimais, podemos contar até FF_{16}, que corresponde ao decimal 255. Para contar além desse valor, são necessários três dígitos hexadecimais. Por exemplo, 100_{16} equivale ao decimal 256, 101_{16} equivale ao decimal 257 e assim por diante. O maior número hexadecimal de três dígitos é FFF_{16}, que equivale ao decimal 4095. O maior número hexadecimal de quatro dígitos é $FFFF_{16}$, que equivale ao decimal 65.535.

Conversão de Binário para Hexadecimal

A conversão de um número binário para hexadecimal é um procedimento direto. Simplesmente separe o número binário em grupos de 4 bits começando do bit mais à direita e substituindo cada grupo de 4 bits pelo símbolo hexadecimal equivalente.

EXEMPLO 2–24

Converta os seguintes números binários para hexadecimal:

(a) 1100101001010111 **(b)** 1111110001011010 01

Solução **(a)** 1100 1010 0101 0111 → C A 5 7 = $\mathbf{CA57_{16}}$

(b) 0011 1111 0001 0110 1001 → 3 F 1 6 9 = $\mathbf{3F169_{16}}$

Dois zeros têm que ser acrescentados no item (b) para completar com 4 bits o grupo à esquerda.

Problema relacionado Converta o número binário 1001111011110011100 para hexadecimal.

Conversão de Hexadecimal para Binário

Hexadecimal é uma forma conveniente de representar números binários.

Para converter um número de hexadecimal para binário, o processo é inverso, sendo que substituímos cada símbolo hexadecimal pelos quatro bits correspondentes.

EXEMPLO 2–25

Determine os números binários correspondentes aos seguintes números hexadecimais:

(a) $10A4_{16}$ **(b)** $CF8E_{16}$ **(b)** 9742_{16}

Solução **(a)** 1 0 A 4 → **1000010100100**

(b) C F 8 E → **1100111110001110**

(c) 9 7 4 2 → **1001011101000010**

No item (a), considere o MSB precedido de três zeros, formando assim um grupo de 4 bits.

Problema relacionado Converta o número hexadecimal 6BD3 em binário.

Deve estar claro que é muito mais fácil lidar com um número hexadecimal do que com o equivalente em binário. Como a conversão é muito fácil, o sistema hexadecimal é amplamente usado na representação de números binários em programação, impressão e displays.

A conversão entre hexadecimal e binário é feita de forma direta e fácil.

Conversão de Hexadecimal para Decimal

Uma forma de determinar o equivalente decimal de um número hexadecimal é primeiro converter o número hexadecimal em binário e em seguida converter de binário para decimal.

EXEMPLO 2–26

Converta o seguinte número hexadecimal em decimal:

(a) $1C_{16}$ **(b)** $A85_{16}$

Solução Lembre-se, converta primeiro o número hexadecimal em binário e em seguida converta o número binário para decimal.

(a) 1 C

$$\overbrace{0001}^{1}\overbrace{1100}^{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$$

(b) A 8 5

$$\overbrace{1010}^{A}\overbrace{1000}^{8}\overbrace{0101}^{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$$

Problema relacionado Converta o número hexadecimal 6BD para decimal.

Outra forma de converter um número hexadecimal no seu equivalente decimal é multiplicar o valor decimal de cada dígito hexadecimal pelo seu peso e então realizar a soma desses produtos. Os pesos de um número hexadecimal são potências de 16 crescentes (da direita para a esquerda). Para um número hexadecimal de 4 dígitos, os pesos são:

16^3	16^2	16^1	16^0
4096	256	16	1

EXEMPLO 2–27

Converta os seguintes números hexadecimais em números decimais:

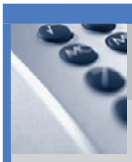
(a) $E5_{16}$ **(b)** $B2F8_{16}$

Solução Consultando a Tabela 2–3, vemos que as letras de A a F representam os números decimais de 10 a 15, respectivamente.

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229}_{10}$

(b) $$\begin{aligned} B2F8_{16} &= (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1) \\ &= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1) \\ &= 45.056 + 512 + 240 + 8 = \mathbf{45.816}_{10} \end{aligned}$$

Problema relacionado Converta $60A_{16}$ em decimal.



TUTORIAL: CALCULADORA

Potências de 16

Exemplo Determine o valor de 16^4.

TI-86 **Passo 1** 1 6 ^

Passo 2 4 ENTER

```
16 ^ 4
   65536
```

TI-36X **Passo 1** 1 6 y^x

Passo 2 4 =

```
65536
```

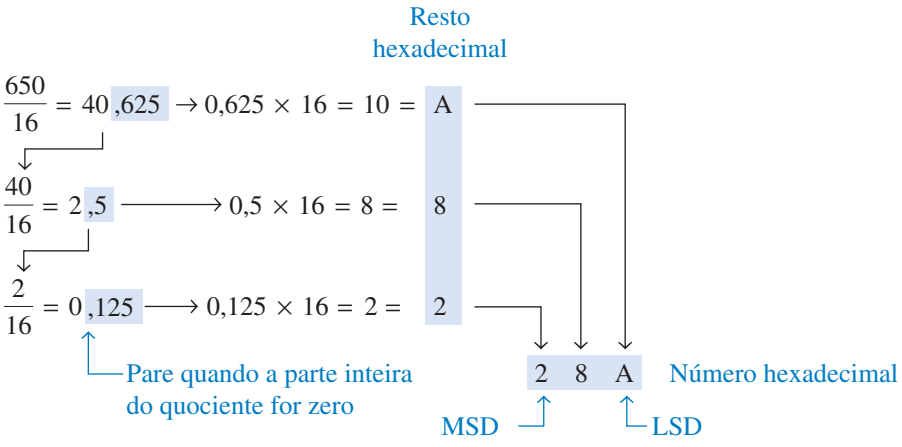
Conversão de Decimal para Hexadecimal

Divisões sucessivas de um número decimal por 16 produzem o número hexadecimal equivalente, formado pelos restos das divisões. O primeiro resto produzido é o digito menos significativo (LSD – *least significant digit*). Cada divisão sucessiva por 16 resulta num resto que se torna num dígito no número hexadecimal equivalente. Esse procedimento é similar à divisão sucessiva por 2 usada na conversão de decimal para binário que foi abordada na Seção 2–3. O Exemplo 2–28 ilustra esse procedimento. Observe que quando o quociente tem uma parte fracionária, essa parte é multiplicada pelo divisor para se obter o resto.

EXEMPLO 2–28

Converta o número decimal 650 em hexadecimal por meio de divisões sucessivas por 16.

Solução



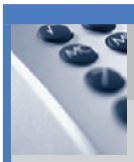
Problema relacionado Converta o decimal 2591 em hexadecimal.

Adição Hexadecimal

A adição pode ser feita diretamente com números hexadecimais lembrando que os dígitos hexadecimais de 0 a 9 são equivalentes aos dígitos decimais de 0 a 9 e que os dígitos hexadecimais de A a F são equivalentes aos números decimais de 10 a 15. Quando somar dois números hexadecimais, use as regras a seguir. (Os números decimais são indicados pelo subscrito 10).

1. Para qualquer coluna de um problema de adição, pense nos dois dígitos hexadecimais em termos dos seus valores decimais. Por exemplo, $5_{16} = 5_{10}$ e $C_{16} = 12_{10}$.
2. Se a soma dos dois dígitos for 15_{10} ou menos, registre o dígito hexadecimal correspondente.
3. Se a soma dos dois dígitos for maior que 15_{10}, registre o valor da soma que excede a 16_{10} e gere um carry de 1 para a próxima coluna.

Uma calculadora pode ser usada para realizar operações aritméticas com números hexadecimais.

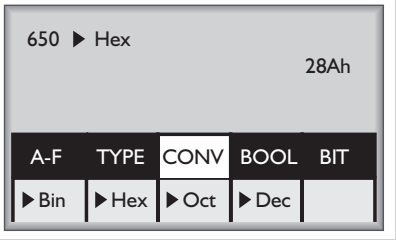


TUTORIAL: CALCULADORA

Conversão de um Número Decimal para Hexadecimal

Exemplo Converta o número decimal 650 para hexadecimal.

		BASE
TI-86	**Passo 1**	2 1 F3
	Passo 2	6 5 0
	Passo 3	F2
	Passo 4	ENTER



		DEC
TI-36X	**Passo 1**	3rd EE
	Passo 2	6 5 0
		HEX
	Passo 3	3rd (

28A

EXEMPLO 2–29

Efetue a soma dos seguintes números hexadecimais:

(a) $23_{16} + 16_{16}$ **(b)** $58_{16} + 22_{16}$ **(c)** $2B_{16} + 84_{16}$ **(d)** $DF_{16} + AC_{16}$

Solução

(a)
$$\begin{array}{r} 23_{16} \\ +\ 16_{16} \\ \hline \mathbf{39}_{16} \end{array}$$
coluna da direita: $3_{16} + 6_{16} = 3_{10} + 6_{10} = 9_{10} = 9_{16}$
coluna da esquerda: $2_{16} + 1_{16} = 2_{10} + 1_{10} = 3_{10} = 3_{16}$

(b)
$$\begin{array}{r} 58_{16} \\ +\ 22_{16} \\ \hline \mathbf{7A}_{16} \end{array}$$
coluna da direita: $8_{16} + 2_{16} = 8_{10} + 2_{10} = 10_{10} = A_{16}$
coluna da esquerda: $5_{16} + 2_{16} = 5_{10} + 2_{10} = 7_{10} = 7_{16}$

(c)
$$\begin{array}{r} 2B_{16} \\ +\ 84_{16} \\ \hline \mathbf{AF}_{16} \end{array}$$
coluna da direita: $B_{16} + 4_{16} = 11_{10} + 4_{10} = 15_{10} = F_{16}$
coluna da esquerda: $2_{16} + 8_{16} = 2_{10} + 8_{10} = 10_{10} = A_{16}$

(d)
$$\begin{array}{r} DF_{16} \\ +\ AC_{16} \\ \hline \mathbf{18B}_{16} \end{array}$$
coluna da direita: $F_{16} + C_{16} = 15_{10} + 12_{10} = 27_{10}$
$27_{10} - 16_{10} = 11_{10} = B_{16}$ com um carry de 1
coluna da esquerda: $D_{16} + A_{16} + 1_{16} = 13_{10} + 10_{10} + 1_{10} = 24_{10}$
$24_{10} - 16_{10} = 8_{10} = 8_{16}$ com um carry de 1

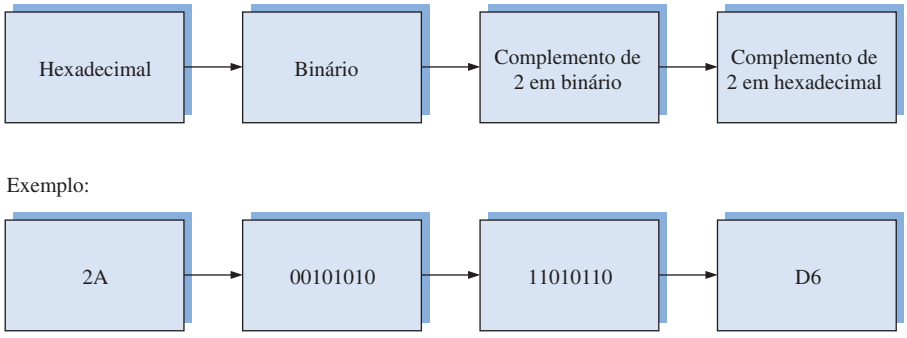
Problema relacionado Some $4C_{16}$ com $3A_{16}$.

Subtração Hexadecimal

Conforme estudamos, o complemento de 2 nos permite subtrair números binários por meio da adição. Como um número hexadecimal pode ser usado para representar um número binário, ele também pode ser usado para representar o complemento de 2 de um número binário.

Existem três formas de obter o complemento de 2 de um número hexadecimal. O método 1 é o mais comum e fácil de ser usado. Os métodos 2 e 3 são alternativos.

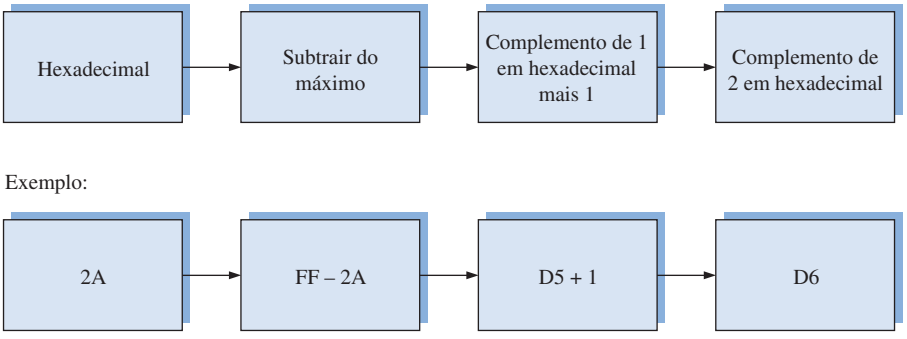
Método 1 Converta o número hexadecimal para binário. Obtenha o complemento de 2 do número binário. Converta o resultado para hexadecimal. Esses passos estão ilustrados na Figura 2–4.



▲ **FIGURA 2–4**

Obtenção do complemento de 2 de um número hexadecimal, Método 1.

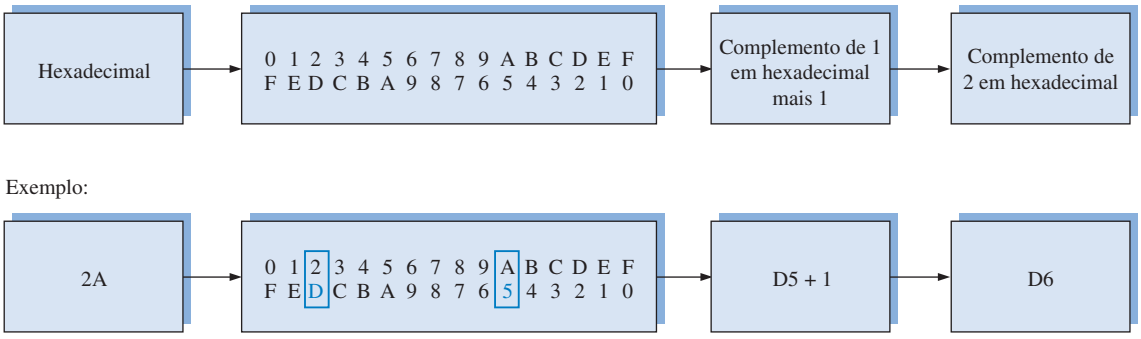
Método 2 Subtraia o número hexadecimal do maior número hexadecimal (com a mesma quantidade de dígitos) e some 1. Isso está ilustrado na Figura 2–5.



▲ **FIGURA 2–5**

Obtenção do complemento de 2 de um número hexadecimal, Método 2.

Método 3 Escreva a seqüência dos números hexadecimais de um dígito. Escreva a seqüência inversa abaixo da seqüência direta. O complemento de 1 de cada dígito hexa é o dígito diretamente abaixo dele. Some 1 ao número resultante para obter o complemento de 2. Esses procedimentos estão ilustrados na Figura 2–6.



▲ **FIGURA 2–6**

Obtenção do complemento de 2 de um número hexadecimal, Método 3.

EXEMPLO 2–30

Efetue a subtração dos seguintes números hexadecimais:

(a) $84_{16} + 2A_{16}$ **(b)** $C3_{16} + 0B_{16}$

Solução **(a)** $2A_{16} = 00101010$

O complemento de 2 de $2A_{16} = 11010110 = D6_{16}$ (usando o Método 1)

$$\begin{array}{rl} 84_{16} & \text{Soma} \\ +\ D6_{16} & \text{Desconsiderar o carry, como na adição} \\ \cancel{1}5A_{16} & \text{do complemento de 2} \end{array}$$

A diferença é $\mathbf{5A}_{16}$.

(b) $0B_{16} = 00001011$

O complemento de 2 de $0B_{16} = 11110101 = F5_{16}$ (usando o Método 1)

$$\begin{array}{rl} C3_{16} & \\ +\ F5_{16} & \text{Soma} \\ \cancel{1}B8_{16} & \text{Desconsiderar o carry} \end{array}$$

A diferença é $\mathbf{B8}_{16}$.

Problema relacionado Subtraia 173_{16} de BCD_{16}.

SEÇÃO 2–8 REVISÃO

1. Converta os seguintes números binários em hexadecimais.
   (a) 10110011 **(b)** 110011101000
2. Converta os seguintes números hexadecimais em binários.
   (a) 57_{16} **(b)** $3A5_{16}$ **(c)** $F80B_{16}$
3. Converta $9B30_{16}$ em decimal.
4. Converta o número decimal 573 em hexadecimal.
5. Some os seguintes números hexadecimais diretamente:
   (a) $18_{16} + 34_{16}$ **(b)** $3F_{16} + 2A_{16}$
5. Efetue as seguintes subtrações de números hexadecimais.
   (a) $75_{16} - 21_{16}$ **(b)** $94_{16} - 5C_{16}$

2-9 NÚMEROS OCTAIS

Assim como o sistema de numeração hexadecimal, o sistema de numeração octal proporciona uma forma conveniente de expressar números binários e códigos. Entretanto, ele é usado menos freqüentemente que o sistema hexadecimal em conjunção com computadores e microprocessadores para expressar quantidades binárias para fins de entrada e saída.

Ao final do estudo desta seção você deverá ser capaz de:

■ Escrever os dígitos do sistema de numeração octal ■ Converter de octal para decimal ■ Converter de decimal para octal ■ Converter de octal para binário ■ Converter de binário para octal

O sistema de numeração **octal** é composto de oito dígitos, os quais são:

0, 1, 2, 3, 4, 5, 6, 7

Para contar acima de 7, inicie uma nova coluna e continue:

10, 11, 12, 13, 14, 15, 16, 17, 20, 21,...

O sistema de numeração octal tem uma base 8.

A contagem em octal é similar à contagem em decimal, exceto que os dígitos 8 e 9 não são usados. A fim de distinguir os números octais dos números decimais ou hexadecimais, usamos o subscrito 8 para indicar que o número em questão é octal. Por exemplo, 15_8 em octal é equivalente a 13_{10} em decimal e D em hexadecimal. Algumas vezes podemos encontrar as letras "o" ou "Q" após o número, indicando que ele é octal.

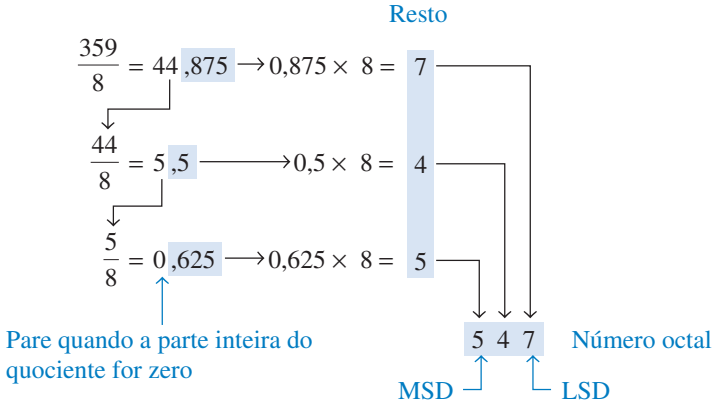
Conversão de Octal para Decimal

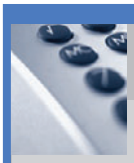
Como o sistema de numeração octal tem uma base de oito, cada posição sucessiva de um dígito é uma potência crescente de oito, começando pela coluna mais à direita com 8^0. O cálculo de um número octal em termos do seu equivalente decimal é realizado multiplicando-se cada dígito pelo seu peso e somando os produtos, conforme ilustrado a seguir para o número 2374_8.

$$\begin{aligned} &\text{Peso: } 8^3\ 8^2\ 8^1\ 8^0 \\ &\text{Número octal: } 2\ \ 3\ \ 7\ \ 4 \\ 2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\ &= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\ &= 1024 + 192 + 56 + 4 = 1276_{10} \end{aligned}$$

Conversão de Decimal para Octal

Um método de conversão de um número decimal para octal é o da divisão sucessiva por 8, similar ao método usado na conversão de números decimais para binário ou para hexadecimal. Para mostrar como se faz, vamos converter o número decimal 359 para octal. Cada divisão sucessiva por 8 resulta num resto que se torna um dígito do número octal equivalente. O primeiro resto gerado é o dígito menos significativo (LSD).



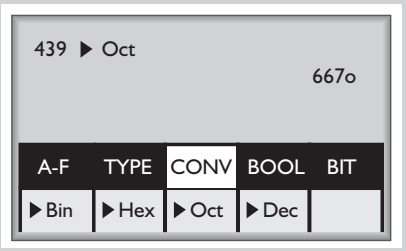


TUTORIAL: CALCULADORA

Conversão de um Número Decimal para Octal

Exemplo Converta o decimal 439 para octal.

		BASE
TI-86	**Passo 1**	2 1 F3
	Passo 2	4 3 9
	Passo 3	F3
	Passo 4	ENTER
		DEC
TI-36X	**Passo 1**	3rd EE
	Passo 2	4 3 9
		OCT
	Passo 3	3rd)



667

Conversão de Octal para Binário

Como o dígito octal pode ser representado por 3 bits, é muito fácil converter de octal para binário. Cada dígito octal é representado por três bits, conforme mostra a Tabela 2–4.

O sistema octal é uma forma conveniente de representar números binários, mas não é tão usado como o hexadecimal.

▼ TABELA 2–4

Conversão de octal para binário

DÍGITO OCTAL	0	1	2	3	4	5	6	7
BINÁRIO	000	001	010	011	100	101	110	111

Para converter um número octal para binário, simplesmente substitua cada dígito octal pelos três bits apropriados.

EXEMPLO 2–31

Converta cada um dos seguintes números octais para binário:

(a) 13_8 **(b)** 25_8 **(c)** 140_8 **(d)** 7526_8

Solução

(a) 1 3 → 001011 **(b)** 2 5 → 010101 **(c)** 1 4 0 → 001100000 **(d)** 7 5 2 6 → 111101010110

Problema relacionado Converta cada um dos números binários obtidos neste exemplo para decimal e verifique que cada valor está de acordo com o valor decimal correspondente ao número octal.

Conversão de Binário para Octal

A conversão de binário para octal é a operação inversa da conversão de octal para binário. O procedimento é o seguinte: comece pelo grupo de três bits mais à direita e, percorrendo os grupos de bits da direita para a esquerda, converta cada grupo no seu dígito octal correspondente. Caso o grupo mais à esquerda não tiver três bits, acrescente um ou dois zeros para completar o grupo. Esses zeros à esquerda não afetam o valor do número binário.

EXEMPLO 2–32

Converta cada número binário a seguir no seu equivalente em octal:

(a) 110101 **(b)** 101111001 **(c)** 100110011010 **(d)** 11010000100

Solução

(a) $\underbrace{110}_{6}\underbrace{101}_{5} = \mathbf{65}_8$

(b) $\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1} = \mathbf{571}_8$

(c) $\underbrace{1001}_{4}\underbrace{1001}_{6}\underbrace{1010}_{3}\underbrace{}_{2} = \mathbf{4632}_8$

(d) $\underbrace{0110}_{3}\underbrace{1000}_{2}\underbrace{0100}_{0}\underbrace{}_{4} = \mathbf{3204}_8$

Problema relacionado Converta o número binário 1010101000111110010 para octal.

SEÇÃO 2–9 REVISÃO

1. Converta os seguintes números octais em decimais:
   (a) 73_8 **(b)** 125_8
2. Converta os seguintes números decimais em octais:
   (a) 98_{10} **(b)** 163_{10}
3. Converta os seguintes números octais em binários:
   (a) 46_8 **(b)** 723_8 **(c)** 5624_8
4. Converta os seguintes números binários em octais:
   (a) 110101111 **(b)** 1001100010 **(c)** 10111111001

2-10 DECIMAL CODIFICADO EM BINÁRIO (BCD)

Decimal codificado em binário (BCD – *binary coded decimal*) é uma forma de expressar cada dígito decimal com um código binário. Existem apenas dez grupos de códigos no sistema BCD, de forma que é muito fácil converter decimal em BCD. Como preferimos ler e escrever em decimal, o código BCD provê uma excelente interface com o sistema binário. Exemplos de tais interfaces são as entradas do teclado e leituras digitais.

Ao final do estudo desta seção você deverá ser capaz de:

■ Converter cada dígito decimal em BCD ■ Expressar números decimais em BCD ■ Converter de BCD para decimal ■ Somar números em BCD

O Código 8421

Em BCD, 4 bits representa cada dígito decimal.

O código 8421 é um tipo de código **BCD** (decimal codificado em binário). Decimal codificado em binário significa que cada dígito decimal, de 0 a 9, é representado por um código binário de quatro bits. A designação 8421 indica os pesos binários dos quatro bits (2^3, 2^2, 2^1, 2^0). A facilidade de conversão entre números em código 8421 e números decimais é a principal vantagem desse código. Tudo o que precisamos fazer é lembrar as dez combinações binárias que representam os dez dígitos conforme mostra a Tabela 2–5. O código 8421 é o código BCD predominante, e quando nos referirmos a BCD, queremos dizer que o código é o 8421, a menos que seja relatado o contrário.

DÍGITO DECIMAL	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

TABELA 2–5
Conversão de decimal para BCD.

Códigos inválidos O leitor já deve ter percebido que, com quatro bits, podemos representar dezesseis números (de 0000 a 1111), porém, no código 8421, apenas dez deles são usados. As seis combinações do código que não são usadas (1010, 1011, 1100, 1101, 1110 e 1111) são inválidas no código BCD 8421.

Para expressar qualquer número decimal em BCD, substitua cada dígito decimal pelo código apropriado de 4 bits, conforme mostra o Exemplo 2–33.

EXEMPLO 2–33

Converta em BCD cada um dos seguintes números decimais:

(a) 35 **(b)** 98 **(c)** 170 **(d)** 2469

Solução

(a) 3 5 → 0011 0101 **(b)** 9 8 → 1001 1000

(c) 1 7 0 → 0001 0111 0000 **(d)** 2 4 6 9 → 0010 0100 0110 1001

Problema relacionado Converta o número decimal 9673 em BCD.

Determinar um número decimal a partir de um número BCD é igualmente fácil. Comece pelo bit mais à direita separando o código em grupos de 4 bits. Em seguida, escreva o dígito representado por cada grupo de quatro bits.

EXEMPLO 2–34

Converta cada um dos seguintes códigos BCD em decimal:

(a) 10000110 **(b)** 001101010001 **(c)** 1001010001110000

Solução

(a) 10000110 → 8 6 **(b)** 001101010001 → 3 5 1 **(c)** 1001010001110000 → 9 4 7 0

Problema relacionado Converta o código BCD 1000001000100111010 em decimal.

Adição em BCD

BCD é um código numérico e pode ser usado em operações aritméticas. A adição é a operação mais importante porque as outras três operações (subtração, multiplicação e divisão) podem ser realizadas através da adição. Eis como dois números BCD são somados:

Passo 1 Some os dois números BCD, usando as regras de adição binária dadas na Seção 2–4.

Passo 2 Se um resultado de 4 bits for igual ou menor que 9, ele é um número BCD válido.

Passo 3 Se um resultado de 4 bits for maior que 9, ou se um carry de saída de um grupo de 4 bits for gerado, ele será um resultado inválido. Some 6 (0110) ao resultado de 4 bits para "pular" os seis estados inválidos e retornar ao código 8421. Se ocorrer um carry quando 6 for somado, simplesmente acrescente o carry ao próximo grupo de 4 bits.

O Exemplo 2–35 ilustra adições BCD nas quais o resultado de cada coluna de 4 bits é igual ou menor que 9, sendo esses resultados números BCD válidos. O Exemplo 2–36 ilustra o procedimento no caso de resultados inválidos (maiores que 9 ou com carry).

EXEMPLO 2–35

Some os seguintes números BCD:

(a) 0011 + 0100 **(b)** 00100011 + 00010101

(c) 10000110 + 00010011 **(d)** 010001010000 + 010000010111

Solução As adições de números decimais são mostradas para comparação.

(a)
$$\begin{array}{r} 0011 \\ +\ 0100 \\ \hline \mathbf{0111} \end{array} \qquad \begin{array}{r} 3 \\ +\ 4 \\ \hline 7 \end{array}$$

(b)
$$\begin{array}{rr} 0010 & 0011 \\ +\ 0001 & 0101 \\ \hline \mathbf{0011} & \mathbf{1000} \end{array} \qquad \begin{array}{r} 23 \\ +\ 15 \\ \hline 38 \end{array}$$

(c)
$$\begin{array}{rr} 1000 & 0110 \\ +\ 0001 & 0011 \\ \hline \mathbf{1001} & \mathbf{1001} \end{array} \qquad \begin{array}{r} 86 \\ +\ 13 \\ \hline 99 \end{array}$$

(d)
$$\begin{array}{rrr} 0100 & 0101 & 0000 \\ +\ 0100 & 0001 & 0111 \\ \hline \mathbf{1000} & \mathbf{0110} & \mathbf{0111} \end{array} \qquad \begin{array}{r} 450 \\ +\ 417 \\ \hline 867 \end{array}$$

Observe que em cada caso o resultado é apenas uma coluna de 4 bits que não excede a 9, sendo números BCD válidos.

Problema relacionado Some os seguintes números BCD: 1001000001000011 + 0000100100100101.

EXEMPLO 2–36

Some os seguintes números BCD:

(a) 1001 + 0100 **(b)** 1001 + 1001

(c) 00010110 + 00010101 **(d)** 01100111 + 01010011

Solução As adições de números decimais são mostradas para comparação.

(a)
$$\begin{array}{rr} & 1001 \\ & +\ 0100 \\ & 1101 \\ & +\ 0110 \\ \hline \mathbf{0001} & \mathbf{0011} \\ \downarrow & \downarrow \\ 1 & 3 \end{array}$$

Número BCD inválido (>9)
Somar 6
Número BCD válido

$$\begin{array}{r} 9 \\ +4 \\ \hline 13 \end{array}$$

(b)
$$\begin{array}{rr} & 1001 \\ & +\ 1001 \\ 1 & 0010 \\ & +\ 0110 \\ \hline \mathbf{0001} & \mathbf{1000} \\ \downarrow & \downarrow \\ 1 & 8 \end{array}$$

Inválido por causa do carry
Somar 6
Número BCD válido

$$\begin{array}{r} 9 \\ +\ 9 \\ \hline 18 \end{array}$$

(c)

0001	0110		16
+ 0001	0101		+ 15
0010	1011	O grupo da direita é inválido (>9) e o grupo da esquerda é válido.	31
	+ 0110	Some 6 ao código inválido. Some o carry, 0001, ao próximo grupo.	
0011	**0001**	Número BCD válido	
↓	↓		
3	1		

(d)

	0110	0111		67
	+ 0101	+ 0011		+ 53
	1011	1010	Ambos os grupos são inválidos (>9)	120
	+ 0110	0110	Some 6 aos dois grupos	
0001	**0010**	**0000**	Número BCD válido	
↓	↓	↓		
1	2	0		

Problema relacionado Some os seguintes números BCD: 01001000 + 00110100.

SEÇÃO 2–10 REVISÃO

1. Qual é o peso binário de cada bit 1 nos números BCD a seguir?
   (a) 0010 **(b)** 1000 **(c)** 0001 **(d)** 0100
2. Converta os seguintes números decimais em números BCD:
   (a) 6 **(b)** 15 **(c)** 273 **(d)** 849
3. Quais números decimais são representados por cada código BCD?
   (a) 10001001 **(b)** 001001111000 **(c)** 000101010111
4. Na adição BCD, quando um resultado de 4 bits é inválido?

2-11 CÓDIGOS DIGITAIS

Muitos códigos específicos são usados em sistemas digitais. Acabamos de estudar o código BCD; agora vamos analisar outros códigos. Alguns códigos são estritamente numéricos, como o BCD, e outros são alfanuméricos; ou seja, são usados para representar números, letras, símbolos e instruções. Os códigos apresentados nesta seção são o Gray e o ASCII.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a vantagem do código Gray ■ Converter o código Gray em binário ■ Usar o código ASCII

O Código Gray

Os bits do **código Gray** não têm peso e ele não é um código aritmético; ou seja, não existem pesos associados às posições dos bits. A característica importante do código Gray é que *ele apresenta uma mudança de um único bit quando se passa de uma palavra do código para a seguinte na seqüência*. Essa propriedade é importante em muitas aplicações, como em codificadores de posição de eixo, onde a suscetibilidade a erros aumenta com o número de mudanças de bits entre números adjacentes em uma seqüência.

A alteração de um único bit, característica do código Gray, minimiza a chance de erro.

A Tabela 2–6 apresenta uma lista de um código Gray de 4 bits para os números decimais de 0 a 15. Os números binários são mostrados na tabela para referência. Assim como os números binários, *o código Gray pode ter qualquer número de bits*. Observe a mudança de apenas um bit entre as palavras do código Gray. Por exemplo, quando se passa do decimal 3 para o 4, o código Gray muda de 0010 para 0110, enquanto que o código binário muda de 0011 para 0100, uma mudança de três bits. Neste exemplo de código Gray, o único bit que muda é o terceiro bit da esquerda para a direita; os outros permanecem inalterados.

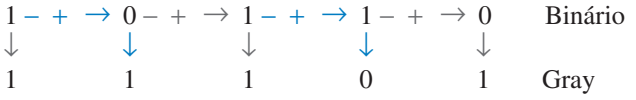
TABELA 2–6
Código Gray de 4 bits

DECIMAL	BINÁRIO	CÓDIGO GRAY	DECIMAL	BINÁRIO	CÓDIGO GRAY
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Conversão de Binário para Código Gray Às vezes é útil fazer conversões entre o código binário e o código Gray. As regras a seguir explicam como converter de um número binário para uma palavra em código Gray.

1. O bit mais significativo (mais à esquerda) no código Gray é o mesmo que o correspondente MSB no número binário.
2. Da esquerda para a direita, some cada par de bits adjacentes no código binário para obter o próximo bit do código Gray. Descarte os carries.

   Por exemplo, a conversão do número binário 10110 para o código Gray é a seguinte:

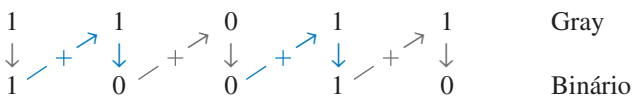


O código Gray é 11101.

Conversão de Código Gray para Binário Para converter de código Gray para binário, usamos um método similar que, entretanto, apresenta algumas diferenças. As seguintes regras são aplicadas:

1. O bit mais significativo (mais à esquerda) no código binário é o mesmo que o correspondente bit no código Gray.
2. Some cada bit do código binário gerado ao bit do código Gray na próxima posição adjacente. Descarte os carries.

   Por exemplo, a conversão do código Gray 11011 para binário é a seguinte:



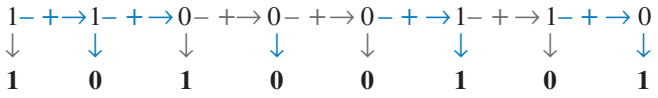
O número binário é 10010.

EXEMPLO 2–37

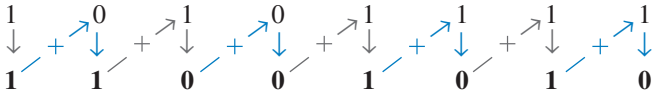
(a) Converta o número binário 11000110 para código Gray.
(b) Converta o código Gray 10101111 para binário.

Solução

(a) De binário para código Gray:



(b) De código Gray para binário:



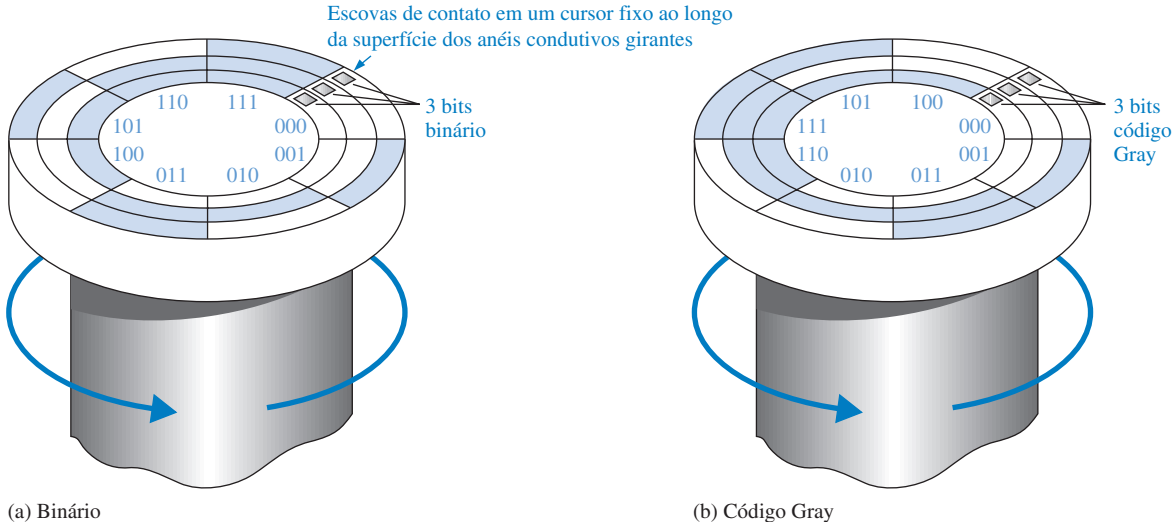
Problema relacionado

(a) Converta o binário 101101 para código Gray.

(b) Converta o código Gray 100111 para binário.

Uma Aplicação

Um diagrama simplificado de um mecanismo codificador de posição de eixo de três bits é mostrado na Figura 2–7. Basicamente, existem três anéis que são segmentados em oito setores. Quanto mais setores existirem, maior será a precisão do posicionamento que pode ser representada, porém estamos usando neste exemplo apenas oito para fins de ilustração. Cada setor de cada anel é fixado a uma tensão de nível alto ou a uma tensão de nível baixo para representar 1s e 0s. Um 1 é indicado por um setor colorido e um 0 por um setor branco. À medida que o eixo gira no sentido anti-horário ao longo dos 360º, os oito setores se movem sob três escovas produzindo uma saída binária de 3 bits que indicam a posição do eixo.



(a) Binário (b) Código Gray

▲ **FIGURA 2–7**

Uma ilustração simplificada de como o código Gray resolve o problema de erro em codificadores da posição de eixo.

Na Figura 2–7(a), os setores são organizados de forma a produzir um padrão binário direto, gerando na passagem das escovas a seqüência 000, 001, 010, 011, e assim por diante. Quando as escovas estão nos setores coloridos, a saída é 1 e quando elas estão nos setores brancos, a saída é 0. Se uma escova estiver um pouco a frente das outras durante a transição de um setor para o próximo, podem ocorrer erros na saída. Considere o que acontece quando as escovas estão no setor 111 e entram no setor 000. Se a escova relativa ao MSB estiver um pouco adiantada, a posição 011 seria indicada incorretamente em vez da transição direta de 111 para 000. Nesse tipo de aplicação, é praticamente impossível manter um alinhamento mecânico preciso para todas as escovas; portanto, alguns erros podem ocorrer em muitas das transições entre setores.

O código Gray é usado para eliminar o problema de erro que é inerente ao código binário. Conforme mostra a Figura 2–7(b), o código Gray garante que apenas um bit mude entre setores adjacentes. Isso significa que mesmo que as escovas não tenham um alinhamento preciso, nunca ocorrerá erros na transição. Por exemplo, vamos considerar novamente o que acontece quando as escovas estão no setor 111 e se movem para o próximo setor, 101. as duas únicas saídas possíveis durante a transição são 111 e 101, não importando como as escovas estão alinhadas. Uma situação similar ocorre na transição de cada um dos outros setores.

Códigos Alfanuméricos

Para nos comunicarmos, não usamos apenas números, mas também letras e outros símbolos. No sentido rigoroso, os códigos **alfanuméricos** representam números e caracteres alfabéticos (letras). Entretanto, a maioria desses códigos representa também outros caracteres, como símbolos e várias instruções necessárias para transmissão de informações.

Um código alfanumérico representa pelo menos 10 dígitos decimais e 26 letras do alfabeto, num total de 36 itens. Esse número requer seis bits em cada representação de código, pois cinco bits não são suficientes ($2^5 = 32$). Existe um total de 64 combinações de seis bits, sendo que 28 combinações do código não são usadas. Obviamente, em muitas aplicações, outros símbolos além de números e letras são necessários para uma comunicação completa. Precisamos de espaços, pontos, vírgulas, ponto-e-vírgulas, interrogação, etc. Precisamos também de instruções que digam ao sistema receptor o que fazer com a informação. Com códigos de seis bits de extensão, podemos operar com números decimais, o alfabeto e outros 28 símbolos. Isso deve ter despertado no leitor a idéia da necessidade de um código alfanumérico básico. O ASCII é o código alfanumérico mais comum e é abordado a seguir.

ASCII

ASCII é a abreviação de American Standard Code for Information Interchange (Código Padrão Americano para Troca de Informações). O código ASCII (pronunciado "askii"), é um código alfanumérico aceito universalmente e usado na maioria dos computadores e outros equipamentos eletrônicos. A maioria dos teclados de computadores é padronizada com o código ASCII. Quando digitamos uma letra, um número ou um comando de controle, o código ASCII correspondente é enviado para o computador.

O ASCII tem 128 caracteres e símbolos representados por um código de 7 bits. Na verdade, o código ASCII pode ser considerado um código de 8 bits com o MSB sempre 0. Esse código de 8 bits vai de 00 até 7F em hexadecimal. Os primeiros 32 caracteres ASCII são comandos não-gráficos que não são impressos ou mostrados e são usados apenas para fins de controle. São exemplos desses caracteres: "nulo", "próxima linha", "início de texto" e "escape". Os outros caracteres são símbolos gráficos que podem ser impresso ou mostrados e incluem as letras do alfabeto (minúsculas e maiúsculas), os dez dígitos decimais, sinais de pontuação e outros símbolos normalmente usados.

A Tabela 2–7 é uma lista do código ASCII mostrando a representação de cada caractere e símbolo em decimal, hexadecimal e binário. A seção à esquerda da tabela é uma lista dos nomes dos 32 caracteres de controle (de 00 a 1F em hexadecimal). Os símbolos gráficos são apresentados no restante da tabela (de 20 a 7F em hexadecimal).

NOTA: COMPUTAÇÃO

O teclado de um computador tem um microprocessador dedicado que constantemente escanea ("lê") o circuito do teclado para detectar quando uma tecla foi pressionada e liberada. Uma única varredura é gerada pelo software do computador representando aquela tecla em particular. O código de varredura é então convertido em código alfanumérico (ASCII) para ser usado pelo computador.

▼ TABELA 2–7

Código Padrão Americano para Troca de Informações (ASCII)

CARACTERES DE CONTROLE				SÍMBOLO GRÁFICO											
NOME	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA	SÍMBOLO	DEC	BINÁRIO	HEXA
NUL	0	0000000	00	espaço	32	0100000	20	@	64	1000000	40	`	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	"	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	’	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	’	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	−	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

EXEMPLO 2–38

Determine o código binário ASCII que é inserido pelo teclado do computador quando a seguinte linha de comando em BASIC é digitada. Expresse também cada código em hexadecimal.

20 PRINTI “A=”;X

Solução O código ASCII para cada símbolo é encontrado na Tabela 2–7.

Símbolo	Binário	Hexadecimal
2	0110010	32_{16}
0	0110000	30_{16}
Espaço	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Espaço	0100000	20_{16}
"	0100010	22_{16}
A	1000001	41_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
X	1011000	58_{16}

Problema relacionado Determine a seqüência de códigos ASCII necessária para expressar a seguinte linha de comando de um programa em hexadecimal:

80 INPUT Y

Os Caracteres de Controle do Código ASCII Os primeiros trinta e dois códigos da tabela ASCII (Tabela 2–7) representam os caracteres de controle. Esses caracteres são usados para permitir que dispositivos como um computador e uma impressora se comuniquem um com o outro quando passam informações e dados. A Tabela 2–8 é uma lista dos caracteres de controle e da função da tecla de controle que permite que os caracteres sejam inseridos diretamente a partir de um teclado ASCII pressionando a tecla de controle (CTRL) e o símbolo correspondente. Também é dada uma breve descrição de cada caractere de controle.

Caracteres Estendidos ASCII

Além dos 128 caracteres padrão ASCII, existem 128 caracteres adicionais que foram adotados pela IBM para uso em seus PCs (computadores pessoais). Devido à popularidade do PC, esses caracteres estendidos ASCII também são usados em outras aplicações além de PCs, e tornaram-se um padrão não-oficial.

Os caracteres estendidos ASCII são representados por um código 8 bits a partir do hexadecimal 80 até FF.

TABELA 2–8
Caracteres de controle ASCII

NOME	DECIMAL	HEXA	TECLAS	DESCRIÇÃO
NUL	0	00	CTRL @	caractere nulo
SOH	1	01	CTRL A	início do cab. de transmissão
STX	2	02	CTRL B	início de texto
ETX	3	03	CTRL C	fim de texto
EOT	4	04	CTRL D	fim de transmissão
ENQ	5	05	CTRL E	interroga
ACK	6	06	CTRL F	confirmação
BEL	7	07	CTRL G	sinal sonoro
BS	8	08	CTRL H	volta um caractere
HT	9	09	CTRL I	tabulação horizontal
LF	10	0A	CTRL J	próxima linha
VT	11	0B	CTRL K	tabulação vertical
FF	12	0C	CTRL L	próxima página
CR	13	0D	CTRL M	início da linha
SO	14	0E	CTRL N	liberação de shift
SI	15	0F	CTRL O	ativação de shift
DLE	16	10	CTRL P	escape da conexão de dados
DC1	17	11	CTRL Q	dipositivo de controle 1
DC2	18	12	CTRL R	dipositivo de controle 2
DC3	19	13	CTRL S	dipositivo de controle 3
DC4	20	14	CTRL T	dipositivo de controle 4
NAK	21	15	CTRL U	negativa de confirmação
SYN	22	16	CTRL V	sincronismo
ETB	23	17	CTRL W	fim de transmissão de bloco
CAN	24	18	CTRL X	cancela
EM	25	19	CTRL Y	fim de meio de transmissão
SUB	26	1A	CTRL Z	substitui
ESC	27	1B	CTRL [	escape
FS	28	1C	CTRL /	separador de arquivo
GS	29	1D	CTRL]	separador de grupo
RS	30	1E	CTRL ^	separador de registro
US	31	1F	CTRL _	separador de unidade

O código ASCII estendido contém caracteres nas seguintes categorias gerais:

1. Caracteres alfabéticos estrangeiros (idioma diferente do inglês)
2. Símbolos de moeda estrangeira
3. Letras gregas
4. Símbolos matemáticos
5. Caracteres gráficos
6. Caracteres de gráfico de barras
7. Caracteres de sombreamento

A Tabela 2–9 é uma lista do conjunto de caracteres ASCII estendidos com as representações decimal e hexadecimal.

TABELA 2–9

Caracteres ASCII estendidos

SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA	SÍMBOLO	DEC	HEXA
Ç	128	80	á	160	A0	└	192	C0	α	224	E0
ü	129	81	í	161	A1	┴	193	C1	β	225	E1
é	130	82	ó	162	A2	┬	194	C2	Γ	226	E2
â	131	83	ú	163	A3	├	195	C3	π	227	E3
ä	132	84	ñ	164	A4	─	196	C4	Σ	228	E4
à	133	85	Ñ	165	A5	┼	197	C5	σ	229	E5
å	134	86	ª	166	A6	╞	198	C6	µ	230	E6
ç	135	87	º	167	A7	╟	199	C7	τ	231	E7
ê	136	88	¿	168	A8	╚	200	C8	Φ	232	E8
ë	137	89	⌐	169	A9	╔	201	C9	Θ	233	E9
è	138	8A	¬	170	AA	╩	202	CA	Ω	234	EA
ï	139	8B	½	171	AB	╦	203	CB	δ	235	EB
î	140	8C	¼	172	AC	╠	204	CC	∞	236	EC
ì	141	8D	¡	173	AD	═	205	CD	φ	237	ED
Ä	142	8E	«	174	AE	╬	206	CE	ϵ	238	EE
Å	143	8F	»	175	AF	╧	207	CF	∩	239	EF
É	144	90	▒	176	B0	╨	208	D0	≡	240	F0
æ	145	91	▒	177	B1	╤	209	D1	±	241	F1
Æ	146	92	▒	178	B2	╥	210	D2	≥	242	F2
ô	147	93	│	179	B3	╙	211	D3	≤	243	F3
ö	148	94	┤	180	B4	╘	212	D4	⌠	244	F4
ò	149	95	╡	181	B5	╒	213	D5	⌡	245	F5
û	150	96	╢	182	B6	╓	214	D6	÷	246	F6
ù	151	97	╖	183	B7	╫	215	D7	≈	247	F7
ÿ	152	98	╕	184	B8	╪	216	D8	°	248	F8
Ö	153	99	╣	185	B9	┘	217	D9	•	249	F9
Ü	154	9A	║	186	BA	┌	218	DA	·	250	FA
¢	155	9B	╗	187	BB	█	219	DB	√	251	FB
£	156	9C	╝	188	BC	■	220	DC	η	252	FC
¥	157	9D	╜	189	BD	▌	221	DD	^{2}	253	FD
Pt	158	9E	╛	190	BE	▐	222	DE	■	254	FE
ƒ	159	9F	┐	191	BF	■	223	DF	□	255	FF

SEÇÃO 2–11 REVISÃO

1. Converta os seguintes números binários para o código Gray:
   (a) 1100 **(b)** 1010 **(c)** 11010
2. Converta as seguintes representações em código Gray para binário:
   (a) 1000 **(b)** 1010 **(c)** 11101
3. Qual é a representação ASCII para cada um dos seguintes caracteres? Expresse cada um como um padrão de bits e em notação hexadecimal.
   (a) K **(b)** r **(c)** $ **(d)** +

2-12 CÓDIGOS DE DETECÇÃO E CORREÇÃO DE ERRO

Nesta seção, discutiremos dois métodos que acrescentam bits aos códigos com a finalidade de detectar erro num único bit ou detectar e corrigir erro num único bit. O método da paridade de detecção de erro é introduzido e o método Hamming de detecção e correção de erro num único bit é abordado com mais detalhes. Quando é identificado que um bit numa dada palavra de código está errado, ele pode ser corrigido fazendo simplesmente a inversão do bit.

Ao final do estudo desta seção você deverá ser capaz de:

■ Determinar se existe um erro num código baseado no bit de paridade ■ Associar a um código o bit de paridade apropriado ■ Usar o código de Hamming para detecção e correção de erro num único bit ■ Associar os bits de paridade adequados para correção de erro num único bit

Método da Paridade para Detecção de Erro

Um bit de paridade diz se o número de 1s é ímpar ou par.

Muitos sistemas usam um bit de paridade como um meio de **detecção de erro** de bit. Qualquer grupo de bits possui um número de 1s par ou ímpar. Um bit de paridade é acrescentado a um grupo de bits para tornar o número de 1s no grupo sempre par ou sempre ímpar. Um bit de paridade par torna o número de 1s par e um bit de paridade ímpar torna ímpar o total de bits.

Um dado sistema pode operar com **paridade** par ou ímpar, porém não ambas. Por exemplo, se um sistema opera com paridade par, é feita uma verificação em cada grupo de bits recebido para certificar-se de que o número total de 1s no grupo seja par. Caso exista um número ímpar de 1s, ocorreu um erro.

Como uma ilustração da forma com que os bits de paridade são acrescentados a um código, a Tabela 2–10 apresenta uma lista dos bits de paridade para cada número BCD tanto para a paridade par quanto para a ímpar. O bit de paridade para cada número BCD está na coluna *P*.

TABELA 2–10

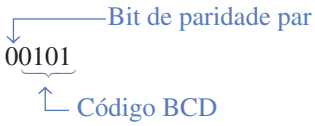
O código BCD com bits de paridade

PARIDADE PAR		PARIDADE ÍMPAR	
P	BCD	*P*	BCD
0	0000	1	0000
1	0001	0	0001
1	0010	0	0010
0	0011	1	0011
1	0100	0	0100
0	0101	1	0101
0	0110	1	0110
1	0111	0	0111
1	1000	0	1000
0	1001	1	1001

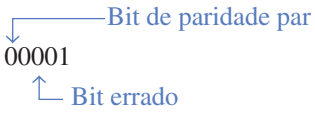
O bit de paridade pode ser acrescentado ao início ou ao final do código, dependendo do projeto do sistema. Observe que o número total de 1s, incluindo o bit de paridade, é sempre par para a paridade par e sempre ímpar para a paridade ímpar.

Detecção de um Erro Um bit de paridade provê a detecção de erro num único bit (ou qualquer número ímpar de erros, que é bem pouco provável) mas não pode verificar dois erros num grupo. Por exemplo, vamos admitir que desejamos transmitir o código BCD 0101. (A paridade pode ser

usada com qualquer número de bits; estamos usando quatro bits como ilustração.) O código total transmitido, incluindo o bit de paridade par, é:



Agora vamos admitir que ocorra um erro no terceiro bit a partir da esquerda (o 1 vira 0).



Quando esse código é recebido, o circuito de verificação de paridade determina que existe apenas um único 1 (paridade ímpar), quando deveria haver um número par de 1s. Devido ao número par de 1s não aparecer no código recebido, é indicado um erro.

Um bit de paridade ímpar também provê uma forma de detecção de erro num único bit num dado grupo de bits.

EXEMPLO 2–39

Associe o bit de paridade par apropriado para os seguintes grupos de códigos:

(a) 1010 **(b)** 111000 **(c)** 101101

(d) 1000111001001 **(e)** 101101011111

Solução Faça o bit de paridade 0 ou 1 conforme necessário para tornar o número total de 1s par. O bit de paridade será o bit mais à esquerda (colorido).

(a) **0**1010 **(b)** **1**111000 **(c)** **0**101101

(d) **0**1000111001001 **(e)** **1**101101011111

Problema relacionado Acrescente um bit de paridade par ao código ASCII de 7 bits para a letra K.

EXEMPLO 2–40

Um sistema de paridade ímpar recebe os seguintes grupos de código: 10110, 11010, 110011, 110101110100 e 1100010101010. Determine quais grupos, se houver algum, estão com erro.

Solução Como é informado que a paridade é ímpar, qualquer grupo com um número par de 1s está incorreto. Os seguintes grupos estão com erro: **110011** e **1100010101010**.

Problema relacionado O seguinte caractere ASCII é recebido por um sistema de paridade ímpar: 00110111. Ele está correto?

O Código de Correção de Erro Hamming

Conforme estudado, um único bit de paridade permite a detecção de erro num único bit numa palavra de código. Um único bit de paridade pode indicar que existe um erro num certo grupo de bits. Para corrigir um erro detectado, mais informação é necessária porque a posição do bit errado tem que ser identificada antes que ele possa ser corrigido. Mais do que um bit de paridade tem que ser incluído no grupo de bits para tornar possível a correção do erro detectado. Em um código de 7 bits, existem sete possibilidades de erro num único bit. Nesse caso, três bits de paridade podem não apenas detectar um erro mas podem especificar a posição do bit errado. O **código Hamming** provê a correção de um único erro. A abordagem a seguir ilustra a construção de um código Hamming de 7 bits para a correção de um único erro.

Número de Bits de Paridade Se o número de bits de dados projetado for d, então o número de bits de paridade, p, é determinado pela seguinte relação:

$$2^p \geq d + p + 1 \qquad \textbf{Equação 2–1}$$

Por exemplo, se temos quatro bits de dados, então p é determinado por tentativa e erro por meio da Equação 2–1. Façamos $p = 2$. Então:

$$2^p = 2^2 = 4$$

e

$$d + p + 1 = 4 + 2 + 1 = 7$$

Como 2^p tem que ser igual ou maior a $d + p + 1$, a relação na Equação 2–1 não é satisfeita. Temos que tentar novamente. Façamos $p = 3$. Então:

$$2^p = 2^3 = 8$$

e

$$d + p + 1 = 4 + 3 + 1 = 8$$

Esse valor de p satisfaz a Equação 2–1, assim são necessários três bits de paridade para proporcionar a correção de um único erro para quatro bits de dados. Deve-se notar que a detecção e correção são proporcionadas por todos os bits, de paridade e de dados, no grupo de código; ou seja, os bits de paridade também são verificados.

Inserção de Bits de Paridade no Código Agora que sabemos determinar o número de bits de paridade necessários no nosso exemplo particular, temos que arranjar os bits adequadamente no código. Devemos saber que nesse exemplo o código é composto de quatro bits de dados e três bits de paridade. O bit mais à esquerda é designado como bit 1, o próximo bit é o 2 e assim por diante, conforme a seguir:

bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7

Os bits de paridade estão localizados nas posições que são numeradas em correspondência às potências de dois ascendentes (1, 2, 4, 8,...), conforme indicado:

P_1, P_2, D_1, P_3, D_2, D_3, D_4

O símbolo P_n designa um bit de paridade em particular e D_n designa um bit de dado em particular.

Determinação dos Valores dos Bits de Paridade Finalmente, temos que designar adequadamente o valor 0 ou 1 a cada bit de paridade. Como cada bit de paridade provê uma verificação em outros determinados bits no código total, temos que saber o valor desses outros bits para determinar o valor do bit de paridade. Para determinar o valor do bit, primeiro numere cada posição de bit em binário, ou seja, escreva o número binário para cada número decimal da posição, conforme mostra a segunda e terceira linhas da Tabela 2–11. Em seguida, indique a localização dos bits de dados e de paridade, conforme mostra a primeira linha da Tabela 2–11. Observe que o número da posição em binário do bit de paridade P_1 tem um 1 no dígito mais à direita. *Esse bit de paridade verifica as posições de todos os bits, incluindo ele mesmo, que têm 1s na mesma posição nos números de posição em binário*. Portanto, o bit de paridade P_1 verifica as posições de bit 1, 3, 5 e 7.

▼ TABELA 2–11

Tabela de posicionamento dos bits para um código de correção de erro de 7 bits

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados (D_n)							
Bits de paridade (P_n)							

O número da posição em binário do bit de paridade P_2 tem um 1 no bit do meio. Ele verifica todas as posições de bit, incluindo ele mesmo, que têm 1 na mesma posição. Portanto, o bit de paridade P_2 verifica os bits das posições 2, 3, 6 e 7.

O número da posição em binário para o bit de paridade P_3 tem um 1 no bit mais à esquerda. Ele verifica todas as posições de bit, incluindo ele mesmo, que têm 1s na mesma posição. Portanto, o bit de paridade P_3 verifica as posições de bit 4, 5, 6 e 7.

Em cada caso, ao bit de paridade é designado um valor que torna a quantidade de 1s, no conjunto de bits que ele verifica, par ou ímpar, dependendo do que for especificado. Os exemplos a seguir devem tornar esse procedimento mais claro.

EXEMPLO 2–41

Determine o código de Hamming para o número BCD 1001 (bits de dados), usando paridade par.

Solução

Passo 1 Determine o número de bits de paridade necessários. Façamos $p = 3$. então:

$$2^p = 2^3 = 8$$
$$d + p + 1 = 4 + 3 + 1 = 8$$

Três bits de paridade são suficientes.

$$\text{Total de bits do código} = 4 + 3 = 7$$

Passo 2 Construa uma tabela de posições de bits, conforme mostra a Tabela 2–12 e insira os bits de dados. Os bits de paridade são determinados nos passos a seguir.

TABELA 2–12

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POS. EM BINÁRIO	001	010	011	100	101	110	111
Bits de dados			1		0	0	1
Bits de paridade	0	0		1			

Passo 3 Determine os bits de paridade como a seguir:

O bit P_1 verifica os bits das posições 1, 3, 5 e 7 e tem que ser 0 para que o número de 1s (2) seja par nesse grupo.

O bit P_2 verifica os bits das posições 2, 3, 6 e 7 e tem que ser 0 para que o número de 1s (2) seja par nesse grupo.

O bit P_3 verifica os bits das posições 4, 5, 6 e 7 e tem que ser 1 para que o número de 1s (2) seja par nesse grupo.

Passo 4 Esses bits de paridade são inseridos na Tabela 2–12 e o código combinado resultante é 0011001.

Problema relacionado Determine o código Hamming para o número BCD 1000 usando a paridade par.

EXEMPLO 2–42

Determine o código Hamming para os bits de dados 10110 usando a paridade ímpar.

Solução

Passo 1 Determine o número de bits de paridade necessários. Nesse caso o número de bits de dados, d, é cinco. A partir do exemplo anterior sabemos que fazendo $p = 3$ não adianta. Experimente fazer $p = 4$:

$$2^p = 2^4 = 16$$
$$d + p + 1 = 5 + 4 + 1 = 10$$

Quatro bits de paridade são suficientes.

$$\text{Total de bits do código} = 5 + 4 = 9$$

Passo 2 Construa uma tabela de posições de bits, Tabela 2–13, e insira os bits de dados. Os bits de paridade são determinados de acordo com os passos a seguir. Observe que P_4 está na posição do bit 8.

Passo 3 Determine os bits de paridade como a seguir:

▼ **TABELA 2–13**

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Bits de dados			1		0	1	1		0
Bits de paridade	1	0		1				1	

O bit P_1 verifica os bits das posições 1, 3, 5, 7 e 9 e tem que ser 1 para que o número de 1s (3) seja ímpar nesse grupo.

O bit P_2 verifica os bits das posições 2, 3, 6 e 7 e tem que ser 0 para que o número de 1s (3) seja ímpar nesse grupo.

O bit P_3 verifica os bits das posições 4, 5, 6 e 7 e tem que ser 1 para que o número de 1s (3) seja ímpar nesse grupo.

O bit P_4 verifica os bits das posições 8 e 9 e tem que ser 1 para que o número de 1s (1) seja ímpar nesse grupo.

Passo 4: Esses bits de paridade são inseridos na Tabela 2–13 e o código combinado resultante é 101101110.

Problema relacionado Determine o código Hamming para 11001 usando paridade ímpar.

Detecção e Correção de Erro com o Código de Hamming

Agora que o método Hamming para construção de um código de erro foi abordado, como o usamos para localizar e corrigir um erro? Cada bit de paridade, ao longo dos seu grupos de bits correspondentes, tem que ser verificado para a paridade adequada. Caso existam três bits de paridade na palavra de código, são geradas três verificações. Caso existam quatro bits de paridade, são geradas quatro verificações, e assim por diante. Cada verificação de paridade apresenta um resul-

tado bom ou ruim. O resultado total de todas as verificações de paridade indica o bit, se houver algum, que está errado, como a seguir:

Passo 1 Comece com o grupo verificado por P_1.

Passo 2 Verifique o grupo quanto a paridade correta. Um 0 representa uma verificação de paridade correta e um 1 representa uma verificação incorreta.

Passo 3 Repita o passo 2 para cada grupo de paridade.

Passo 4 O número binário formado pelo resultado de todas as verificações de paridade determina a posição do bit do código que está errado. Esse é o *código de posição de erro*. A primeira verificação de paridade gera o bit menos significativo (LSB). Se todas as verificações forem corretas, não há erro.

EXEMPLO 2–43

Considere que a palavra de código dada no Exemplo 2–41 (0011001) seja transmitida e que 0010001 seja recebida. O receptor não "sabe" o que foi transmitido e tem que testar as paridades para determinar se o código está correto. Determine qualquer erro que tenha ocorrido na transmissão se a paridade usada foi a par.

Solução Primeiro, faça uma tabela de posição de bit, conforme indicado na Tabela 2–14.

▼ **TABELA 2–14**

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4
POSIÇÃO DOS BITS	1	2	3	4	5	6	7
NÚMERO DA POSIÇÃO EM BINÁRIO	001	010	011	100	101	110	111
Código recebido	0	0	1	0	0	0	1

Primeira verificação de paridade:
O bit P_1 verifica as posições 1, 3, 5 e 7.
Existem dois 1s nesse grupo.
A verificação de paridade é correta. ⟶ 0 (LSB)

Segunda verificação de paridade:
O bit P_2 verifica as posições 2, 3, 6 e 7.
Existem dois 1s nesse grupo.
A verificação de paridade é correta. ⟶ 0

Terceira verificação de paridade:
O bit P_3 verifica as posições 4, 5, 6 e 7.
Existe um 1 nesse grupo.
A verificação de paridade é incorreta. ⟶ 1 (MSB)

Resultado:
O código de posição de erro é 100 (binário quatro). Isso diz que o bit na posição 4 está errado. Ele é 0 e deveria ser 1. O código corrigido é 0011001, que está de acordo com o código transmitido.

Problema relacionado Repita o processo ilustrado nesse exemplo se o código recebido for 0111001.

EXEMPLO 2–44

O código 101101010 é recebido. Corrija qualquer erro. Existem quatro bits de paridade, sendo que a paridade usada é a ímpar.

Solução Primeiro, faça uma tabela de posição de bit como a Tabela 2–15.

▼ **TABELA 2–15**

DESIGNAÇÃO DOS BITS	P_1	P_2	D_1	P_3	D_2	D_3	D_4	P_4	D_5
POSIÇÃO DOS BITS	1	2	3	4	5	6	7	8	9
NÚMERO DA POS. EM BINÁRIO	0001	0010	0011	0100	0101	0110	0111	1000	1001
Código recebido	1	0	1	1	0	1	0	1	0

Primeira verificação de paridade:

O bit P_1 verifica as posições 1, 3, 5, 7 e 9.
Existem dois 1s nesse grupo.
A verificação de paridade é incorreta. ⟶ 1 (LSB)

Segunda verificação de paridade:

O bit P_2 verifica as posições 2, 3, 6 e 7.
Existem dois 1s nesse grupo.
A verificação de paridade é incorreta. ⟶ 1

Terceira verificação de paridade:

O bit P_3 verifica as posições 4, 5, 6 e 7.
Existem dois 1s nesse grupo.
A verificação de paridade é incorreta. ⟶ 1

Quarta verificação de paridade:

O bit P_4 verifica as posições 8 e 9.
Existe um 1 nesse grupo.
A verificação de paridade é correta. ⟶ 0 (MSB)

Resultado:

O código de posição de erro é 0111 (binário sete). Isso diz que o bit na posição 7 está errado. O código correto é portanto 101101110.

Problema relacionado O código 101111001 é recebido. Corrija qualquer erro se a paridade ímpar foi usada.

SEÇÃO 2–12 REVISÃO

1. Qual código de paridade ímpar está errado?
   (a) 1011 **(b)** 1110 **(c)** 0101 **(d)** 1000
2. Qual código de paridade par está errado?
   (a) 11000110 **(b)** 00101000 **(c)** 10101010 **(d)** 11111011
3. Acrescente um bit de paridade par no final de cada um dos seguintes códigos:
   (a) 1010100 **(b)** 0100000 **(c)** 1110111 **(d)** 10001100
4. Quantos bits de paridade são necessários para os bits de dados 11010 usando o código de Hamming?
5. Crie o código de Hamming para os bits de dados 0011 usando a paridade par.

RESUMO

- O número binário é um número posicional em que o peso de cada dígito de um número inteiro é uma potência positiva de dois e o peso de cada dígito da parte fracionária é uma potência de dois negativa. Os pesos num número inteiro aumentam da direita para a esquerda (do dígito menos significativo para o mais significativo).
- Um número binário pode ser convertido para um número decimal somando os valores decimais dos pesos de todos os 1s no número binário.
- Um número inteiro decimal pode ser convertido em binário usando a soma dos pesos ou o método da divisão sucessiva por 2.
- Um número decimal fracionário pode ser convertido para binário usando a soma dos pesos ou o método da multiplicação sucessiva por 2.
- As regras básicas para a adição binária são:

  $0 + 0 = 0$
  $0 + 1 = 1$
  $1 + 0 = 1$
  $1 + 1 = 10$

- As regras básicas para a subtração binária são:

  $0 - 0 = 0$
  $1 - 1 = 1$
  $1 - 0 = 1$
  $10 - 1 = 1$

- O complemento de 1 de um número binário é obtido trocando 1s por 0s e 0s por 1s.
- O complemento de 2 de um número binário é obtido somando 1 ao complemento de 1.
- A subtração binária pode ser realizada por meio de adição usando o método do complemento de 1 ou de 2.
- Um número binário positivo é representado por um bit de sinal 0.
- Um número binário negativo é representado por um bit de sinal 1.
- Para operações aritméticas, os números binários negativos são representados na forma do complemento de 2 ou complemento de 1.
- Em operações de adição, um overflow é possível quando os dois números são positivos ou quando os dois números são negativos. Um bit de sinal incorreto numa soma indica a ocorrência de um overflow.
- O sistema de numeração hexadecimal consiste de 16 dígitos e caracteres, de 0 a 9 seguidos de A até F.
- Um dígito hexadecimal representa um número de 4 bits sendo a sua principal finalidade a simplificação de padrões de bits tornando-os de fácil leitura.
- Um número decimal pode ser convertido para hexadecimal usando o método da divisão sucessiva por 16.
- O sistema de numeração octal consiste de oito dígitos, de 0 a 7.
- Um número decimal pode ser convertido para octal usando o método da divisão sucessiva por 8.
- A conversão de octal para binário é realizada simplesmente substituindo cada dígito octal pelo seu equivalente binário de 3 bits. O processo é invertido na conversão de binário para octal.
- Um número decimal é convertido para BCD substituindo cada dígito decimal pelo código binário de 4 bits apropriado.
- ASCII é um código alfanumérico de 7 bits que é amplamente usado em sistemas de computador para entrada e saída de informação.
- Um bit de paridade é usado para detectar um erro num código.
- O código Hamming provê a detecção e correção de um único erro numa palavra de código.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Alfanumérico Consiste de numerais, letras e outros caracteres.

ASCII Código Padrão Americano para Troca de Informações; o código alfanumérico mais amplamente usado.

BCD Decimal codificado em binário; um código digital no qual cada um dos dígitos decimais, de 0 a 9, é representado por um grupo de quatro bits.

Byte Um grupo de oito bits.

Código Hamming Um tipo de código de correção de erro.

Hexadecimal Descreve um sistema de numeração com base 16.

LSB Bit menos significativo; o bit mais à direita num código ou número inteiro binário.

MSB Bit mais significativo; o bit mais à esquerda num código ou número inteiro binário.

Número em ponto flutuante Uma representação numérica baseada em notação científica na qual o número consiste de um expoente e uma mantissa.

Octal Descreve um sistema de numeração com base oito.

Paridade Em relação aos códigos binários, a condição de paridade par ou ímpar do número de 1s num grupo de código.

AUTOTESTE

As respostas estão no final do capítulo.

1. $2 \times 10^1 + 8 \times 10^0$ é igual a

(a) 10 **(b)** 280 **(c)** 2,8 **(d)** 28

2. O número binário 1101 é igual ao no número decimal

(a) 13 **(b)** 49 **(c)** 11 **(d)** 3

3. O número binário 11011101 é igual ao número decimal

(a) 121 **(b)** 221 **(c)** 441 **(d)** 256

4. O número decimal 17 é igual ao número binário

(a) 10010 **(b)** 11000 **(c)** 10001 **(d)** 01001

5. O número decimal 175 é igual ao número binário

(a) 11001111 **(b)** 10101110 **(c)** 10101111 **(d)** 11101111

6. O resultado da soma de 11010 + 01111 é igual a

(a) 101001 **(b)** 101010 **(c)** 110101 **(d)** 101000

7. A diferença de 110 – 010 é igual a

(a) 001 **(b)** 010 **(c)** 101 **(d)** 100

8. O complemento de 1 de 10111001 é

(a) 01000111 **(b)** 01000110 **(c)** 11000110 **(d)** 10101010

9. O complemento de 2 de 11001000 é

(a) 00110111 **(b)** 00110001 **(c)** 01001000 **(d)** 00111000

10. O número decimal +122 é expresso na forma do complemento de 2 como

(a) 01111010 **(b)** 11111010 **(c)** 01000101 **(d)** 10000101

11. O número decimal –34 é expresso na forma do complemento de 2 como

(a) 01011110 **(b)** 10100010 **(c)** 11011110 **(d)** 01011101

12. Um número binário de ponto flutuante de precisão simples tem um total de

(a) 8 bits **(b)** 16 bits **(c)** 24 bits **(d)** 32 bits

13. Na forma do complemento de 2, o número binário 10010011 é igual ao número decimal

(a) −19 **(b)** +109 **(c)** +91 **(d)** −109

14. O número binário 10110011100101010100001 pode ser escrito em octal como

(a) 5471230_8 **(b)** 5471241_8 **(c)** 2634521_8 **(d)** 23162501_8

15. O número binário 1000110101000110111 pode ser escrito em hexadecimal como
(a) $AD467_{16}$ **(b)** $8C46F_{16}$ **(c)** $8D46F_{16}$ **(d)** $AE46F_{16}$

16. O número binário equivalente a $F7A9_{16}$ é
(a) 1111011110101001 **(b)** 1110111110101001
(c) 1111111010110001 **(d)** 1111011010101001

17. O número BCD para o decimal 473 é
(a) 111011010 **(b)** 110001110011 **(c)** 010001110011 **(d)** 010011110011

18. Consulte a Tabela 2–7. O comando STOP em ASCII é
(a) 10100111010100100111111010000 **(b)** 10100101001100100111010100000
(c) 10010101101101100111010100001 **(d)** 10100111010100100111011100100

19. O código que tem erro de paridade par é
(a) 1010011 **(b)** 1101000 **(c)** 1001000 **(d)** 1110111

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 2–1 Números Decimais

1. Qual é o peso do dígito 6 em cada um dos seguintes números decimais?
(a) 1386 **(b)** 54.692 **(c)** 671.920

2. Expresse cada um dos seguintes números decimais como uma potência de dez:
(a) 10 **(b)** 100 **(c)** 10.000 **(d)** 1.000.000

3. Determine o valor de cada dígito nos números decimais a seguir:
(a) 471 **(b)** 9356 **(c)** 125.000

4. Até que valor é possível contar com números decimais de 4 dígitos?

SEÇÃO 2–2 Números Binários

5. Converta para decimal os números binários a seguir:
(a) 11 **(b)** 100 **(c)** 111 **(d)** 1000
(e) 1001 **(f)** 1100 **(g)** 1011 **(h)** 1111

6. Converta os seguintes números binários para decimal:
(a) 1110 **(b)** 1010 **(c)** 11100 **(d)** 10000
(e) 10101 **(f)** 11101 **(g)** 10111 **(h)** 11111

7. Converta cada número binário a seguir para decimal:
(a) 110011,11 **(b)** 101010,01 **(c)** 1000001,111
(d) 1111000,101 **(e)** 1011100,10101 **(f)** 1110001,0001
(g) 1011010,1010 **(h)** 1111111,11111

8. Qual o maior número decimal que pode ser representado pelas seguintes quantidades de dígitos binários (bits)?
(a) dois **(b)** três **(c)** quatro **(d)** cinco **(e)** seis
(f) sete **(g)** oito **(h)** nove **(i)** dez **(j)** onze

9. Quantos bits são necessários para representar os seguintes números decimais?
(a) 17 **(b)** 35 **(c)** 49 **(d)** 68
(e) 81 **(f)** 114 **(g)** 132 **(h)** 205

10. Determine a seqüência binária para cada seqüência decimal a seguir:
(a) 0 a 7 **(b)** 8 a 15 **(c)** 16 a 31
(d) 32 a 63 **(e)** 64 a 75

SEÇÃO 2–3 Conversão de Decimal para Binário

11. Converta cada número decimal a seguir para binário usando o método da soma dos pesos:

(a) 10 **(b)** 17 **(c)** 24 **(d)** 48

(e) 61 **(f)** 93 **(g)** 125 **(h)** 186

12. Converta cada fração decimal para binário usando o método da soma dos pesos:

(a) 0,32 **(b)** 0,246 **(c)** 0,0981

13. Converta cada número decimal para binário usando o método da divisão sucessiva por 2.

(a) 15 **(b)** 21 **(c)** 28 **(d)** 34

(e) 40 **(f)** 59 **(g)** 65 **(h)** 73

14. Converta cada fração decimal para binário usando o método da multiplicação sucessiva por 2:

(a) 0,98 **(b)** 0,347 **(c)** 0,9028

SEÇÃO 2–4 Aritmética Binária

15. Some os seguintes números binários:

(a) 11 + 01 **(b)** 10 + 10 **(c)** 101 + 11

(d) 111 + 110 **(e)** 1001 + 101 **(f)** 1101 + 1011

16. Use a subtração direta para os seguintes números binários:

(a) 11 − 1 **(b)** 101 − 100 **(c)** 110 − 101

(d) 1110 − 11 **(e)** 1100 − 1001 **(f)** 11010 − 10111

17. Realize as seguintes multiplicações binárias:

(a) 11 × 11 **(b)** 100 × 10 **(c)** 111 × 101

(d) 1001 × 110 **(e)** 1101 × 1101 **(f)** 1110 × 1101

18. Faça a operação de divisão binária conforme indicado:

(a) 100 ÷ 10 **(b)** 1001 ÷ 11 **(c)** 1100 ÷ 100

SEÇÃO 2–5 Complementos de 1 e de 2 de Números Binários

19. Determine o complemento de 1 de cada número binário:

(a) 101 **(b)** 110 **(c)** 1010

(d) 11010111 **(e)** 1110101 **(f)** 00001

20. Determine o complemento de 2 de cada número binário a seguir usando qualquer método:

(a) 10 **(b)** 111 **(c)** 1001 **(d)** 1101

(e) 11100 **(f)** 10011 **(g)** 10110000 **(h)** 00111101

SEÇÃO 2–6 Números Sinalizados

21. Expresse cada número decimal a seguir em um número binário do tipo sinal-magnitude de 8 bits:

(a) +29 **(b)** −85 **(c)** +100 **(d)** −123

22. Expresse cada número decimal a seguir como um número de 8 bits na forma do complemento de 1:

(a) −34 **(b)** +57 **(c)** −99 **(d)** +115

23. Expresse cada número decimal a seguir como um número de 8 bits na forma do complemento de 2:

(a) +12 **(b)** −68 **(c)** +101 **(d)** −125

24. Determine o valor decimal de cada número binário sinalizado a seguir na forma sinal-magnitude:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

25. Determine o valor decimal de cada número binário sinalizado a seguir na forma do complemento de 1:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

26. Determine o valor decimal de cada número binário sinalizado a seguir na forma do complemento de 2:

(a) 10011001 **(b)** 01110100 **(c)** 10111111

27. Expresse cada um dos seguintes números binários no formato de ponto flutuante de precisão simples:

(a) 0111110000101011 **(b)** 100110000011000

28. Determine os valores dos números em ponto flutuante de precisão simples a seguir:

(a) 1 10000001 01001001110001000000000

(b) 0 11001100 10000111110100100000000

SEÇÃO 2–7 Operações Aritméticas com Números Sinalizados

29. Converta cada par de números decimais para binário e some-os usando a forma do complemento de 2.

(a) 33 e 15 **(b)** 56 e −27 **(c)** −46 e 25 **(d)** −110 e −84

30. Realize cada adição a seguir na forma do complemento de 2:

(a) 00010110 + 00110011 **(b)** 01110000 + 10101111

31. Realize cada adição a seguir na forma do complemento de 2:

(a) 10001100 + 00111001 **(b)** 11011001 + 11100111

32. Realize cada subtração a seguir na forma do complemento de 2:

(a) 00110011 − 00010000 **(b)** 01100101 − 11101000

33. Multiplique 01101010 por 11110001 na forma do complemento de 2.

34. Divida 01000100 por 00011001 na forma do complemento de 2:

SEÇÃO 2–8 Números Hexadecimais

35. Converta para binário cada número hexadecimal a seguir:

(a) 38_{16} **(b)** 59_{16} **(c)** $A14_{16}$ **(d)** $5C8_{16}$

(e) 4100_{16} **(f)** $FB17_{16}$ **(g)** $8A9D_{16}$

36. Converta para hexadecimal cada número binário a seguir:

(a) 1110 **(b)** 10 **(c)** 10111

(d) 10100110 **(e)** 1111110000 **(f)** 100110000010

37. Converta para decimal cada número hexadecimal a seguir:

(a) 23_{16} **(b)** 92_{16} **(c)** $1A_{16}$ **(d)** $8D_{16}$

(e) $F3_{16}$ **(f)** EB_{16} **(g)** $5C2_{16}$ **(h)** 700_{16}

38. Converta para hexadecimal cada número decimal a seguir:

(a) 8 **(b)** 14 **(c)** 33 **(d)** 52

(e) 284 **(f)** 2890 **(g)** 4019 **(h)** 6500

39. Realize as seguintes adições:

(a) $37_{16} + 29_{16}$ **(b)** $A0_{16} + 6B_{16}$ **(c)** $FF_{16} + BB_{16}$

40. Realize as seguintes subtrações:

(a) $51_{16} - 40_{16}$ **(b)** $C8_{16} - 3A_{16}$ **(c)** $FD_{16} - 88_{16}$

SEÇÃO 2–9 Números Octais

41. Converta para decimal cada número octal a seguir:

(a) 12_8 **(b)** 27_8 **(c)** 56_8 **(d)** 64_8 **(e)** 103_8

(f) 557_8 **(g)** 163_8 **(h)** 1024_8 **(i)** 7765_8

42. Converta para octal cada número decimal a seguir fazendo divisões sucessivas por 8:

(a) 15 **(b)** 27 **(c)** 46 **(d)** 70

(e) 100 **(f)** 142 **(g)** 219 **(h)** 435

43. Converta para binário cada número octal a seguir:

(a) 13_8 **(b)** 57_8 **(c)** 101_8 **(d)** 321_8 **(e)** 540_8
(f) 4653_8 **(g)** 13271_8 **(h)** 45600_8 **(i)** 100213_8

44. Converta para octal cada número binário a seguir:

(a) 111 **(b)** 10 **(c)** 110111
(d) 101010 **(e)** 1100 **(f)** 1011110
(g) 101100011001 **(h)** 10110000011 **(i)** 111111101111000

SEÇÃO 2–10 Decimal Codificado em Binário (BCD)

45. Converta para BCD 8421 cada um dos seguintes números decimais:

(a) 10 **(b)** 13 **(c)** 18 **(d)** 21 **(e)** 25 **(f)** 36
(g) 44 **(h)** 57 **(i)** 69 **(j)** 98 **(k)** 125 **(l)** 156

46. Converta para binário direto cada um dos números do Problema 45 e compare o número de bits necessários nesses dois problemas.

47. Converta para BCD os seguintes números decimais:

(a) 104 **(b)** 128 **(c)** 132 **(d)** 150 **(e)** 186
(f) 210 **(g)** 359 **(h)** 547 **(i)** 1051

48. Converta para decimal os números BCD a seguir:

(a) 0001 **(b)** 0110 **(c)** 1001
(d) 00011000 **(e)** 00011001 **(f)** 00110010
(g) 01000101 **(h)** 10011000 **(i)** 100001110000

49. Converta para decimal cada um dos números BCD a seguir:

(a) 10000000 **(b)** 001000110111
(c) 001101000110 **(d)** 010000100001
(e) 011101010100 **(f)** 100000000000
(g) 100101111000 **(h)** 0001011010000011
(i) 1001000000011000 **(j)** 0110011001100111

50. Some os seguintes números BCD:

(a) 0010 + 0001 **(b)** 0101 + 0011
(c) 0111 + 0010 **(d)** 1000 + 0001
(e) 00011000 + 00010001 **(f)** 01100100 + 00110011
(g) 01000000 + 01000111 **(h)** 10000101 + 00010011

51. Some os seguintes números BCD:

(a) 1000 + 0110 **(b)** 0111 + 0101
(c) 1001 + 1000 **(d)** 1001 + 0111
(e) 00100101 + 00100111 **(f)** 01010001 + 01011000
(g) 10011000 + 10010111 **(h)** 010101100001 + 011100001000

52. Converta para BCD cada par de números decimais e faça a soma conforme indicado:

(a) 4 + 3 **(b)** 5 + 2 **(c)** 6 + 4 **(d)** 17 + 12
(e) 28 + 23 **(f)** 65 + 58 **(g)** 113 + 101 **(h)** 295 + 157

SEÇÃO 2–11 Códigos Digitais

53. Numa determinada aplicação, uma seqüência de 4 bits varia ciclicamente de 1111 a 0000. Existe uma alteração de 4 bits, e em função de atrasos no circuito, essas alterações podem não ocorrer no mesmo instante. Por exemplo, se o LSB mudar primeiro, o número aparecerá como 1110 durante a transição de 1111 para 0000 podendo ser interpretado erroneamente pelo sistema. Ilustre como o código Gray evita esse problema.

54. Converta para código Gray cada número binário a seguir:

(a) 11011 **(b)** 1001010 **(c)** 1111011101110

55. Converta para binário cada código Gray a seguir:

(a) 1010 **(b)** 00010 **(c)** 11000010001

56. Converta para ASCII cada um dos seguintes números decimais. Consulte a Tabela 2–7.

(a) 1 **(b)** 3 **(c)** 6 **(d)** 10 **(e)** 18

(f) 29 **(g)** 56 **(h)** 75 **(i)** 107

57. Determine cada caractere codificado a seguir em ASCII. Consulte a Tabela 2–7.

(a) 0011000 **(b)** 1001010 **(c)** 0111101

(d) 0100011 **(e)** 0111110 **(f)** 1000010

58. Decodifique a seguinte mensagem codificada em ASCII:

1001000 1100101 1101100 1101100 1101111 0101110
0100000 1001000 1101111 1110111 0100000 1100001
1110010 1100101 0100000 1111001 1101111 1110101
0111111

59. Escreva em hexadecimal a mensagem apresentada no Problema 58.

60. Converta para ASCII a seguinte linha comando de um programa de computador:

30 INPUT A,B

SEÇÃO 2–12 Códigos de Detecção e Correção de Erro

61. Determine qual dos seguintes códigos com paridade par apresenta erro:

(a) 100110010 **(b)** 011101010 **(c)** 10111111010001010

62. Determine qual dos seguintes códigos com paridade ímpar apresenta erro:

(a) 11110110 **(b)** 00110001 **(c)** 01010101010101010

63. Acrescente um bit de paridade par aos seguintes bytes de dados:

(a) 10100100 **(b)** 00001001 **(c)** 11111110

64. Determine o código de Hamming com paridade par para os bits de dados 1100.

65. Determine o código de Hamming com paridade ímpar para os bits de dados 1101.

66. Corrija qualquer erro em cada um dos seguintes códigos de Hamming com paridade par.

(a) 1110100 **(b)** 1000111

67. Corrija qualquer erro em cada um dos seguintes códigos de Hamming com paridade ímpar.

(a) 110100011 **(b)** 100001101

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 2–1 Números Decimais

1. **(a)** 1370: 10 **(b)** 6725: 100 **(c)** 7051: 1000 **(d)** 58,72: 0,1

2. **(a)** $51 = (5 \times 10) + (1 \times 1)$ **(b)** $137 = (1 \times 100) + (3 \times 10) + (7 \times 1)$ **(c)** $1492 = (1 \times 1000) + (4 \times 100) + (9 \times 10) + (2 \times 1)$ **(d)** $106{,}58 = (1 \times 100) + (0 \times 10) + (6 \times 1) + (5 \times 0{,}1) + (8 \times 0{,}01)$

SEÇÃO 2–2 Números Binários

1. $2^8 - 1 = 255$

2. O peso é 16.

3. 10111101,011 = 189,375

SEÇÃO 2–3 Conversão de Decimal para Binário

1. **(a)** $23 = 10111$ **(b)** $57 = 111001$ **(c)** $45{,}5 = 101101{,}1$
2. **(a)** $14 = 1110$ **(b)** $21 = 10101$ **(c)** $0{,}375 = 0{,}011$

SEÇÃO 2–4 Aritmética Binária

1. **(a)** $1101 + 1010 = 10111$ **(b)** $10111 + 01101 = 100100$
2. **(a)** $1101 - 0100 = 1001$ **(b)** $1001 - 0111 = 0010$
3. **(a)** $110 \times 111 = 101010$ **(b)** $1100 \div 011 = 100$

SEÇÃO 2–5 Complementos de 1 e de 2 de Números Binários

1. **(a)** Compl. de 1 de 00011010 = 11100101 **(b)** Compl. de 1 de 11110111 = 000010(
   (c) Compl. de 1 de 10001101 = 01110010
2. **(a)** Compl. de 2 de 00010110 = 11101010 **(b)** Compl. de 2 de 11111100 = 000001(
   (c) Compl. de 2 de 10010001 = 01101111

SEÇÃO 2–6 Números Sinalizados

1. Sinal-magnitude: +9 = 00001001
2. Complemento de 1: –33 = 11011110
3. Complemento de 2: –46 = 11010010
4. Bit de sinal, expoente e mantissa.

SEÇÃO 2–7 Operações Aritméticas com Números Sinalizados

1. Casos da adição: o número positivo é maior, o número negativo é maior, ambos são positivos, ambos são negativos.
2. $00100001 + 10111100 = 11011101$
3. $01110111 - 00110010 = 01000101$
4. O sinal do produto é positivo.
5. $00000101 \times 01111111 = 01001111011$
6. O sinal do quociente é negativo.
7. $00110000 \div 00001100 = 00000100$

SEÇÃO 2–8 Números Hexadecimais

1. **(a)** $10110011 = B3_{16}$ **(b)** $110011101000 = CE8_{16}$
2. **(a)** $57_{16} = 01010111$ **(b)** $3A5_{16} = 001110100101$
   (c) $F8OB_{16} = 1111100000001011$
3. $9B30_{16} = 39.728_{10}$
4. $573_{10} = 23D_{16}$
5. **(a)** $18_{16} + 34_{16} = 4C_{16}$ **(b)** $3F_{16} + 2A_{16} = 69_{16}$
6. **(a)** $75_{16} - 21_{16} = 54_{16}$ **(b)** $94_{16} - 5C_{16} = 38_{16}$

SEÇÃO 2–9 Números Octais

1. **(a)** $73_8 = 59_{10}$ **(b)** $125_8 = 85_{10}$
2. **(a)** $98_{10} = 142_8$ **(b)** $163_{10} = 243_8$
3. **(a)** $46_8 = 100110$ **(b)** $723_8 = 111010011$ **(c)** $5624_8 = 101110010100$
4. **(a)** $110101111 = 657_8$ **(b)** $1001100010 = 1142_8$ **(c)** $10111111001 = 2771_8$

SEÇÃO 2–10 Decimal Codificado em Binário (BCD)

1. **(a)** 0010: 2 **(b)** 1000: 8 **(c)** 0001: 1 **(d)** 0100: 4

2. **(a)** $6_{10} = 0110$ **(b)** $15_{10} = 00010101$ **(c)** $273_{10} = 001001110011$
(d) $849_{10} = 100001001001$

3. **(a)** $10001001 = 89_{10}$ **(b)** $001001111000 = 278_{10}$ **(c)** $000101010111 = 157_{10}$

4. Um resultado de 4 bits é inválido quando ele for maior que 9_{10}.

SEÇÃO 2–11 Códigos Digitais

1. **(a)** $1100_2 = 1010$ Gray **(b)** $1010_2 = 1111$ Gray **(c)** $11010_2 = 10111$ Gray

2. **(a)** 1000 Gray $= 1111_2$ **(b)** 1010 Gray $= 1100_2$ **(c)** 11101 Gray $= 10110_2$

3. **(a)** K: $1001011 \rightarrow 4B_{16}$ **(b)** r: $1110010 \rightarrow 72_{16}$
(c) \$: $0100100 \rightarrow 24_{16}$ **(d)** +: $0101011 \rightarrow 2B_{16}$

SEÇÃO 2–12 Códigos de Detecção e Correção de Erro

1. (c) 0101 tem um erro.

2. (d) 11111011 tem um erro.

3. **(a)** 1010100**1** **(b)** 0100000**1** **(c)** 1110111**0** **(d)** 1000110**1**

4. Quatro bits de paridade.

5. **100**0011 (os bits de paridade estão em cor)

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

2–1. 9 tem um valor de 900, 3 tem um valor de 30, 9 tem um valor de 9.

2–2. 6 tem um valor de 60, 7 tem um valor de 7, 9 tem um valor de 9/10 (0,9), 2 tem um valor de 2/100 (0,02), 4 tem um valor de 4/1000 (0,004).

2–3. $10010001 = 128 + 16 + 1 = 145$ **2–4.** $10{,}111 = 2 + 0{,}5 + 0{,}25 + 0{,}125 = 2{,}875$

2–5. $125 = 64 + 32 + 16 + 8 + 4 + 1 = 1111101$ **2–6.** $39 = 100111$

2–7. $1111 + 1100 = 11011$ **2–8.** $111 - 100 = 011$ **2–9.** $110 - 101 = 001$

2–10. $1101 \times 1010 = 10000010$ **2–11.** $1100 \div 100 = 11$ **2–12.** 00110101

2–13. 01000000 **2–14.** Veja a Tabela 2–16. **2–15.** $01110111 = +119_{10}$

▶ **TABELA 2–16**

	SINAL - MAGNITUDE	COMPL. DE 1	COMPL. DE 2
+19	00010011	00010011	00010011
−19	10010011	11101100	11101101

2–16. $11101011 = -20_{10}$ **2–17.** $11010111 = -41_{10}$

2–18. 110000100010100110000000000 **2–19.** 01010101 **2–20.** 00010001

2–21. 1001000110 **2–22.** $(83)(-59) = -4897$ (10110011011111 em complemento de 2)

2–23. $100 \div 25 = 4$ (0100) **2–24.** $4F79C_{16}$ **2–25.** 0110101111010011_2

2–26. $6BD_{16} = 011010111101 = 2^{10} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0$
$= 1024 + 512 + 128 + 32 + 16 + 8 + 4 + 1 = 1725_{10}$

2–27. $60A_{16} = (6 \times 256) + (0 \times 16) + (10 \times 1) = 1546_{10}$

2–28. $2591_{10} = A1F_{16}$ **2–29.** $4C_{16} + 3A_{16} = 86_{16}$

2–30. $BCD_{16} - 173_{16} = A5A_{16}$

2–31. **(a)** $001011_2 = 11_{10} = 13_8$ **(b)** $010101_2 = 21_{10} = 25_8$
(c) $001100000_2 = 96_{10} = 140_8$ **(d)** $111101010110_2 = 3926_{10} = 7526_8$

2–32. 1250762_8 **2–33.** 1001011001110011 **2–34.** $82{,}276_{10}$

2–35. 1001100101101000 **2–36.** 10000010 **2–37.** **(a)** 111011 (Gray) **(b)** 111010_2

2–38. A seqüência de códigos para 80 INPUT Y é $38_{16}30_{16}20_{16}49_{16}4E_{16}50_{16}55_{16}54_{16}20_{16}59_{16}$

2–39. 01001011 **2–40.** Sim **2–41.** *111*0000 **2–42.** *001*0100*0*1

2–43. O bit na posição 010 (2) está errado. O correto é 0011001.

2–44. O bit na posição 0010 (2) está errado. O correto é 111111000.

AUTOTESTE

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (c) **6.** (a) **7.** (d) **8.** (b)

9. (d) **10.** (a) **11.** (c) **12.** (d) **13.** (d) **14.** (b) **15.** (c) **16.** (a)

17. (c) **18.** (a) **19.** (b)

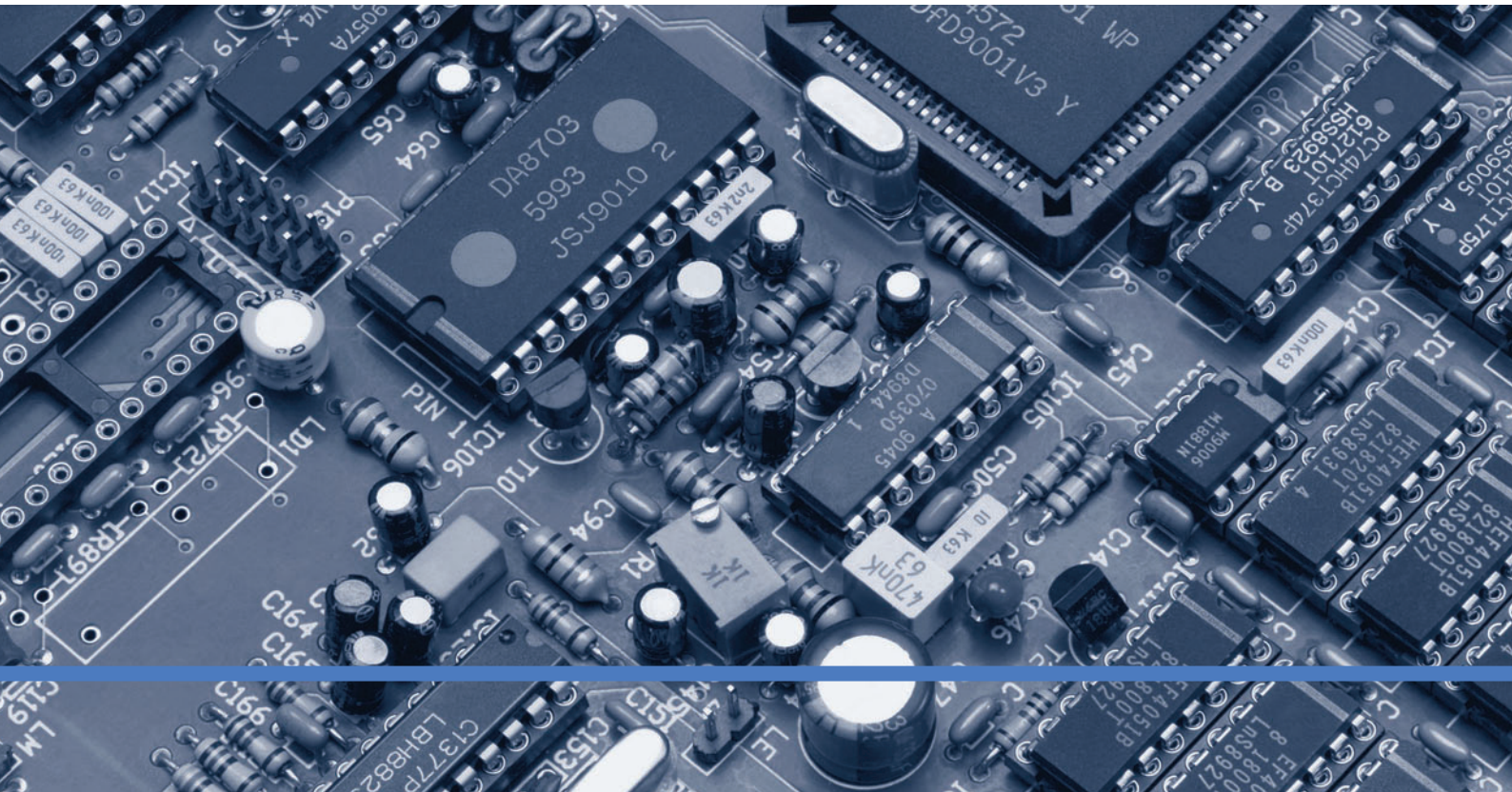
3

PORTAS LÓGICAS

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Descrever a operação do inversor, da porta AND e da porta OR
- Descrever a operação da porta NAND e da porta NOR
- Expressar a operação da função NOT e das portas AND, OR, NAND e NOR usando a álgebra Booleana
- Descrever a operação das portas EX-OR e EX-NOR
- Reconhecer e usar tanto os símbolos característicos de portas lógicas quanto os símbolos retangulares do padrão da 91-1984 ANSI/IEEE.



- Construir diagramas de temporização mostrando as relações de tempo entre as entradas e saídas de diversas portas lógicas
- Discutir os conceitos básicos da lógica programável
- Estabelecer comparações básicas entre as principais tecnologias de CIs (CMOS e TTL)
- Explicar as diferenças dentro das séries das famílias lógicas TTL e CMOS
- Definir *tempo de atraso de propagação*, *dissipação de potência*, *produto velocidade-potência* e *fan-out* relativo às portas lógicas
- Fazer uma lista especificando os circuitos integrados de funções fixas que contêm os diversos tipos de portas lógicas
- Usar cada uma das portas lógicas em aplicações simples
- Efetuar a análise de defeito em circuitos com portas lógicas para os casos de curto-circuito e circuito aberto usando um osciloscópio

TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- Inversor
- Tabela verdade
- Diagrama de temporização
- Álgebra Booleana
- Complemento
- Porta AND
- Habilitação
- Porta OR
- Porta NAND
- Porta NOR
- Porta EX-OR
- Arranjo AND
- Fusível
- Antifusível
- EPROM
- EEPROM
- SRAM
- Dispositivo de destino
- JTAG
- CMOS
- TTL
- Tempo de atraso de propagação
- Fan-out
- Unidade de carga

INTRODUÇÃO

A ênfase deste capítulo está na operação, aplicação e análise de defeito de portas lógicas. As formas de onda que relacionam as entradas com a saída de uma porta lógica usando diagrama de temporização são abordadas minuciosamente.

Os símbolos lógicos usados para representar as portas lógicas estão de acordo com o padrão 91-1984 da ANSI/IEEE. Esse padrão foi adotado pela indústria privada e militar para uso em documentações internas bem como na literatura publicada.

Tanto a lógica programável quanto a lógica de funções fixas são discutidas nesse capítulo. Devido aos circuitos integrados (CIs) serem usados em todas as aplicações, as funções lógicas de um dispositivo são geralmente mais importantes para o técnico ou tecnólogo do que os detalhes da operação do circuito em nível de componentes dentro do encapsulamento do CI. Portanto, a abordagem detalhada de dispositivos em nível de componente pode ser tratada como um tópico opcional. Para aqueles que necessitam desse conhecimento e têm tempo, o Capítulo 14 apresenta uma abordagem minuciosa da tecnologia de circuitos integrados digitais, no qual determinadas partes podem ser usadas como referência ao longo desse livro. Sugestão: reveja a Seção 1-3 antes de iniciar o estudo deste capítulo.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

(SÉRIES CMOS E TTL)

74XX00	74XX11	74XX32
74XX02	74XX20	74XX86
74XX04	74XX21	74XX266
74XX08	74XX27	
74XX10	74XX30	

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

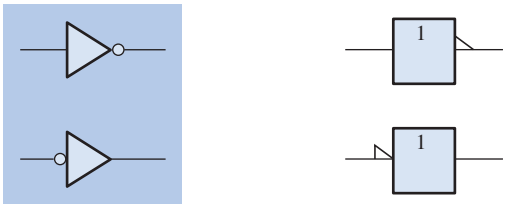
3-1 O INVERSOR

O inversor (circuito NOT) realiza a operação denominada *inversão* ou *complementação*. O inversor troca um nível lógico para o nível lógico oposto. Em termos de bit, ele troca 1 por 0 e 0 por 1.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar os indicadores de negação e polaridade ■ Identificar um inversor através do símbolo característico ou retangular ■ Gerar uma tabela-verdade para um inversor ■ Descrever a operação lógica de um inversor.

Os símbolos lógicos padronizados para um **inversor** são mostrados na Figura 3–1. A parte (a) mostra os símbolos *característicos* e a parte (b) mostra os símbolos *retangulares*. Nesse livro geralmente são usados os símbolos característicos; entretanto, os símbolos retangulares são encontrados em muitas publicações da indústria, sendo que o leitor deve se familiarizar com eles também (os símbolos lógicos estão de acordo com o Padrão 91-1984 da **ANSI/IEEE**).



(a) Símbolos característicos com indicadores de negação

(b) Símbolos retangulares com indicadores de polaridade

▶ **FIGURA 3–1**

Símbolos lógicos padrão para o inversor (padrão 91-1984 da ANSI/IEEE).

Os Indicadores de Negação e Polaridade

O indicador de negação é um pequeno círculo (○) que indica a **inversão** ou complementação quando ele aparece na entrada ou saída de qualquer elemento lógico, como mostra a Figura 3–1(a) para o inversor. Geralmente, as entradas estão à esquerda do símbolo lógico e a saída à direita. Quando o pequeno círculo aparece na entrada significa que um 0 é o estado de entrada ativo ou *acionado* e a entrada é denominada de entrada ativa em nível BAIXO. Quando o pequeno círculo aparece na saída significa que um 0 é o estado de saída ativo ou acionado e a saída é denominada de saída ativa em nível BAIXO. A ausência de um pequeno círculo na entrada ou saída significa que um 1 é o estado ativo ou acionado, sendo que, nesse caso, a entrada ou saída é denominada ativa em nível ALTO.

O indicador de polaridade ou nível é um triângulo (◺) que indica a inversão quando aparece na entrada ou saída de um elemento lógico, como mostra a Figura 3–1(b). Quando aparece na entrada significa que o nível BAIXO é o estado ativo ou acionado da entrada. Quando aparece na saída significa que o nível BAIXO é o estado ativo ou acionado da saída.

Qualquer um dos indicadores (pequeno círculo e triângulo) pode ser usado nos símbolos característicos ou retangulares. A Figura 3–1(a) indica os principais símbolos de inversores usados nesse livro. Observe que uma troca no local de inserção do indicador de negação ou polaridade não implica em uma mudança na forma do inversor operar.

▼ **TABELA 3–1**

Tabela-verdade do inversor

ENTRADA	SAÍDA
BAIXO (0)	ALTO (1)
ALTO (1)	BAIXO (0)

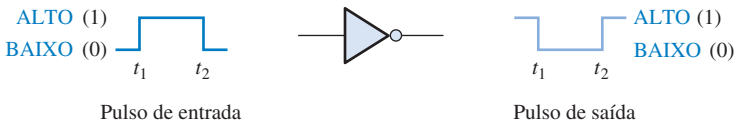
Tabela-Verdade do Inversor

Quando um nível ALTO é aplicado na entrada de um inversor, um nível BAIXO aparece na saída dele. Quando um nível BAIXO é aplicado na entrada de um inversor, um nível ALTO aparece na saída dele. Essa operação aparece resumida na Tabela 3–1, que mostra a saída para cada entrada possível em termos de níveis lógicos e os bits correspondentes. Uma tabela como essa é denominada **tabela-verdade**.

Operação do Inversor

A Figura 3–2 mostra a saída de um inversor para um pulso de entrada, onde t_1 e t_2 indicam os pontos correspondentes nas formas de onda de entrada e saída.

Quando a entrada for nível BAIXO, a saída será nível ALTO; quando a entrada for nível ALTO, a saída será nível BAIXO, gerando então um pulso de saída invertido.



Pulso de entrada Pulso de saída

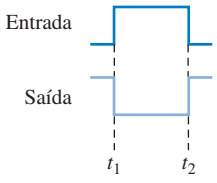
▲ **FIGURA 3–2**

Operação de um inversor com um pulso de entrada. Abrir o arquivo F03-02 para verificar a operação de um inversor.

Diagramas de Temporização

Lembre-se, do Capítulo 1, de que um **diagrama de temporização** é basicamente um gráfico que apresenta com precisão a relação temporal de duas ou mais formas de onda. Por exemplo, a relação temporal entre o pulso de saída e o de entrada na Figura 3–2 pode ser mostrado com um simples diagrama de temporização alinhando os dois pulsos de forma que as bordas dos pulsos apareçam numa relação temporal adequada. A borda de subida do pulso de entrada e a borda de descida do pulso de saída acontecem no mesmo instante (idealmente). De forma similar, a borda de descida do pulso de entrada e a borda de subida do pulso de saída ocorrem no mesmo instante (idealmente). Essa relação de temporização é mostrada na Figura 3-3. Os diagramas de temporização são especialmente úteis para ilustrar a relação temporal de formas de onda digital com múltiplos pulsos.

Um diagrama de temporização mostra a relação temporal de duas ou mais formas de onda.



◀ **FIGURA 3–3**

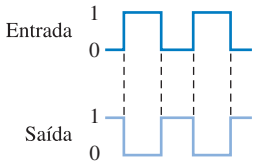
Diagrama de temporização para o caso da Figura 3–2.

EXEMPLO 3–1

Uma forma de onda é aplicada no inversor da Figura 3-4. Determine a forma de onda de saída correspondente à entrada e mostre o diagrama de temporização. De acordo com o posicionamento do pequeno círculo, qual é o estado ativo da saída?

▶ **FIGURA 3–4**

Solução A forma de onda de saída é exatamente oposta à de entrada (invertida), conforme mostra a Figura 3–5, que é um diagrama de temporização. O estado ativo ou acionado da saída é **0**.



▶ **FIGURA 3–5**

Problema relacionado* Se o inversor for mostrado com o indicador de negação (pequeno círculo) na entrada em vez da saída, em que isso afetaria o diagrama de temporização?

* As respostas estão no final do capítulo.

Expressão Lógica para um Inversor

A álgebra Booleana usa variáveis e operadores para descrever um circuito lógico.

Na **álgebra Booleana**, que é a matemática dos circuitos lógicos e será abordada minuciosamente no Capítulo 4, uma variável é representada por uma letra. O **complemento** de uma variável é representado por uma barra sobre a letra. Uma variável pode assumir um valor 1 ou 0. Se uma determinada variável for 1, o complemento dela será 0 e vice-versa.

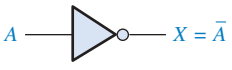
A operação de um inversor (circuito NOT) pode ser expressa como a seguir: se a variável de entrada é denominada A e a variável de saída é denominada X, então

$$X = \overline{A}$$

Essa equação diz que a saída é o complemento da entrada, assim, se $A = 0$, então $X = 1$ e se $A = 1$, X = 0. A Figura 3–6 ilustra isso. A variável complementada pode ser lida como "A barra" ou "A negado".

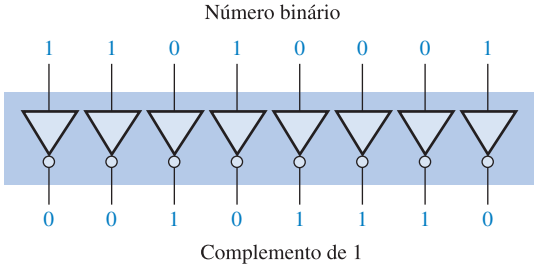
▶ **FIGURA 3–6**

O inversor complementa a variável de entrada.



Uma Aplicação

A Figura 3–7 mostra um circuito que produz o complemento de 1 de um número de 8 bits. Os bits do número são inseridos nas entradas dos inversores e o complemento de 1 do número aparece nas saídas.



▶ **FIGURA 3–7**

Exemplo de um circuito para gerar o complemento de 1 usando inversores.

SEÇÃO 3–1 REVISÃO
As respostas estão no final do capítulo.

1. Quando um 1 está na entrada de um inversor, qual é a saída?
2. Um pulso ativo em nível ALTO (nível ALTO quando acionado, e nível BAIXO em caso contrário) faz-se necessário na entrada de um inversor.
   (a) Desenhe o símbolo lógico apropriado, usando a forma característica e o indicador de negação, para o inversor dessa aplicação.
   (a) Descreva a saída quando um pulso positivo é aplicado na entrada do inversor.

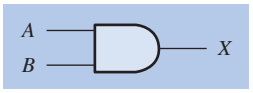
3-2 A PORTA AND

A porta AND é uma das portas básicas que pode ser combinada para formar qualquer função lógica. Uma porta AND pode ter duas ou mais entradas e realizar uma operação conhecida como multiplicação lógica.

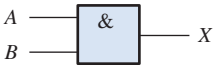
Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma porta AND pelo seu símbolo característico ou pelo seu símbolo retangular ■ Descrever a operação de uma porta AND ■ Gerar a tabela-verdade para uma porta AND com qualquer número de entradas ■ Desenhar um diagrama de temporização para uma porta AND com quaisquer formas de onda especificadas de entrada ■ Escrever a expressão lógica para uma porta AND com qualquer número de entradas ■ Discutir exemplos de aplicações com portas AND

O termo *porta* é usado para descrever um circuito que realiza uma operação lógica básica. A porta AND é composta de duas ou mais entradas e uma única saída, conforme indicado pelo símbolo lógico padrão mostrado na Figura 3–8. As entradas estão à esquerda e a saída está à direita de cada símbolo. A figura mostra portas com duas entradas; entretanto, uma porta AND pode ter qualquer número de entradas maior que um. Embora sejam apresentados como exemplos os símbolos característico e retangular, o símbolo característico, mostrado na parte (a), é usado predominantemente nesse livro.



(a) Formato característico



(b) Formato retangular com o símbolo de qualificação AND (&)

▲ **FIGURA 3–8**
Símbolos lógicos padrões para a porta AND de duas entradas (padrão 91-1984 da ANSI/IEEE).

NOTA: COMPUTAÇÃO

As portas lógicas são os blocos construtivos de computadores. A maioria das funções num computador, exceto certos tipos de memórias, são implementadas com portas lógicas usadas numa escala de integração muito ampla. Por exemplo, um microprocessador, a principal parte de um computador, é construído com centenas de milhares ou ainda milhões de portas lógicas.

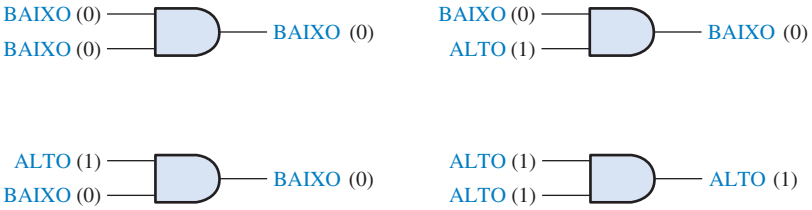
Operação de uma Porta AND

Uma **porta AND** produz uma saída de nível ALTO *apenas* quando *todas* as entradas forem nível ALTO. Quando qualquer uma das entradas for nível BAIXO, a saída será nível BAIXO. Portanto, o propósito básico da porta AND é determinar quando certas condições são simultaneamente verdadeiras, conforme indicado pelos níveis ALTOS em todas as entradas e para produzir um nível ALTO na saída para indicar que todas essas condições são verdadeiras. As entradas da porta AND de 2 entradas mostrada na Figura 3–8 são denominadas *A* e *B* e a saída é denominada *X*. A operação da porta pode ser expressa da seguinte forma:

Uma porta AND pode ter mais que duas entradas.

Para uma porta AND de 2 entradas, a saída *X* será nível ALTO apenas quando as entradas *A* e *B* forem nível ALTO; *X* será nível BAIXO quando *A* ou *B* for nível BAIXO, ou ainda quando *A* e *B* forem nível BAIXO.

A Figura 3–9 ilustra uma porta AND de 2 entradas com todas as quatro possibilidades de combinações de entrada e a saída resultante para cada uma.



▲ FIGURA 3–9

Todas as combinações possíveis de níveis lógicos para uma porta AND de 2 entradas. Abra o arquivo F03-09 para verificar a operação de uma porta AND.

Tabela-Verdade da Porta AND

Para uma porta AND, todas as entradas em nível ALTO fazem com que a saída seja nível ALTO.

A operação lógica de uma porta pode ser expressa com uma tabela-verdade que apresenta uma lista de todas as combinações de entrada com as correspondentes saídas, conforme ilustrado na Tabela 3–2 para uma porta AND de 2 entradas. A tabela-verdade pode ser expandida para qualquer número de entradas. Embora os termos ALTO e BAIXO tendem a dar um sentido "físico" aos estados de entrada e saída, a tabela-verdade é expressa com 1s e 0s; um nível ALTO é equivalente a um 1 e um nível BAIXO é equivalente a um 0 em lógica positiva. Para qualquer porta AND, independente do número de entradas, a saída é nível ALTO *apenas* quando *todas* as entradas forem níveis ALTOS.

▶ TABELA 3–2

Tabela-verdade para uma pota AND de 2 entradas

ENTRADAS		SAÍDA
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

1 = ALTO, 0 = BAIXO

O número total de combinações possíveis de entradas binárias para uma porta é determinado pela seguinte fórmula:

Equação 3–1

$$N = 2^n$$

onde N é o número de combinações de entrada possíveis e n é o número de variáveis de entrada. Para ilustrar,

Para duas variáveis de entrada: $N = 2^2 = 4$ combinações

Para três variáveis de entrada: $N = 2^3 = 8$ combinações

Para quatro variáveis de entrada: $N = 2^4 = 16$ combinações

Podemos determinar o número de combinações binárias de entrada com qualquer número de entradas usando a Equação 3–1.

EXEMPLO 3–2

(a) Desenvolva a tabela-verdade para uma porta AND de 3 entradas.

(b) Determine o número total de combinações de entrada possíveis para uma porta AND de 4 entradas.

Solução

(a) Existem oito combinações de entrada possíveis ($2^3 = 8$) para uma porta AND de 3 entradas. O lado da entrada da tabela-verdade (Tabela 3–3) mostra todas as oito combinações de três bits. O lado da saída é todo de 0s exceto quando todos os bits de entradas são 1s.

► **TABELA 3–3**

ENTRADAS			SAÍDA
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

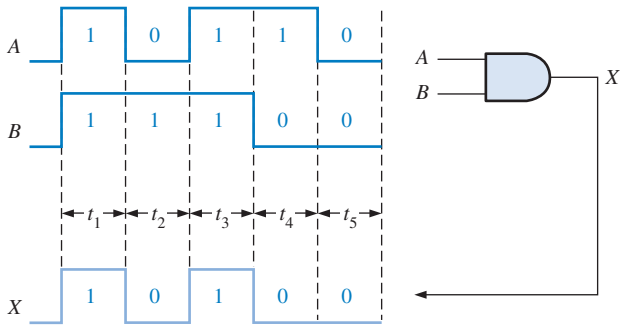
(b) $N = 2^4 = \mathbf{16}$. Existem 16 combinações binárias possíveis de entrada para uma porta AND de 4 entradas.

Problema relacionado Desenvolva a tabela-verdade para uma porta AND de 4 entradas.

Operação com Formas de Onda nas Entradas

Na maioria das aplicações, as entradas de uma porta não apresentam níveis estacionários mas são formas de onda de tensão que variam freqüentemente entre os níveis lógicos ALTO e BAIXO. Agora vamos analisar a operação das portas AND com formas de onda de pulsos nas entradas, tendo em mente que uma porta obedece a operação de uma tabela-verdade independente se as entradas dela são níveis constantes ou níveis que variam (ALTO e BAIXO).

Vamos examinar a operação com forma de onda nas entradas de uma porta AND observando as entradas uma relativa a outra para determinar o nível de saída num determinado instante. Na Figura 3–10, as entradas *A* e *B* são nível ALTO (1) durante o intervalo de tempo t_1, tornando a saída *X* nível ALTO (1) durante esse intervalo de tempo. Durante o intervalo de tempo t_2, a entrada *A* é



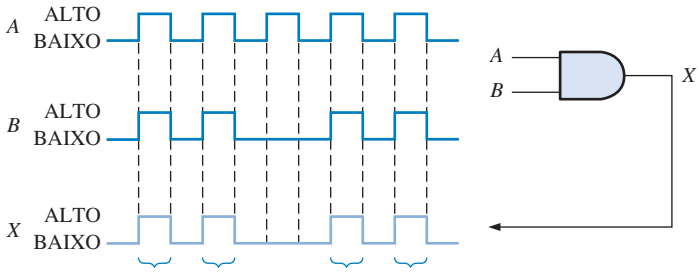
◄ **FIGURA 3–10**

Exemplo da operação de uma porta AND com um diagrama de temporização mostrando as relações entre entrada e saída.

nível BAIXO (0) e a entrada *B* é nível ALTO (1), de forma que a saída é nível BAIXO (0). Durante o intervalo de tempo t_3, as duas entradas são nível ALTO (1) novamente, portanto a saída é nível ALTO (1). Durante o intervalo de tempo t_4, a entrada *A* é nível ALTO (1) e a entrada *B* é nível BAIXO (0), resultando num nível BAIXO na saída. Finalmente, durante o intervalo de tempo t_5, a entrada *A* é nível BAIXO (0), a entrada *B* é nível BAIXO (0), sendo portanto a saída nível BAIXO (0). Assim como sabemos, um diagrama de formas de onda de entrada e saída que mostram as relações temporais é denominado de *diagrama de temporização*.

EXEMPLO 3–3

Se duas formas de onda, *A* e *B*, são aplicadas nas entradas de uma porta AND conforme mostrado na Figura 3–11, qual é a forma de onda de saída resultante?



A e *B* são nível ALTO durante esses quatro intervalos de tempo. Portanto, *X* é nível ALTO.

▲ FIGURA 3–11

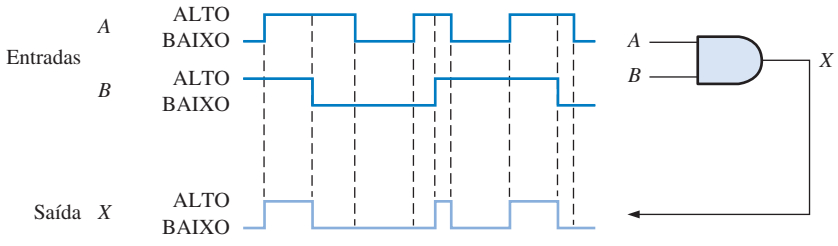
Solução A forma de onda na saída *X* é nível ALTO apenas quando as formas de onda em *A* e *B* forem nível ALTO conforme o diagrama de temporização visto na Figura 3–11.

Problema relacionado Determine a forma de onda de saída e mostre o diagrama de temporização no caso em que o segundo e o quarto pulsos na forma de onda *A* mostrada na Figura 3–11 forem substituídos por níveis BAIXOS.

Lembre-se, quando analisamos a operação de portas lógicas num diagrama de temporização, é importante prestar atenção nas relações temporais de todas as entradas entre si e com a saída.

EXEMPLO 3–4

Para as formas de onda de entrada, *A* e *B,* vistas na Figura 3–12, mostre a forma de onda de saída relacionando-a adequadamente às entradas.



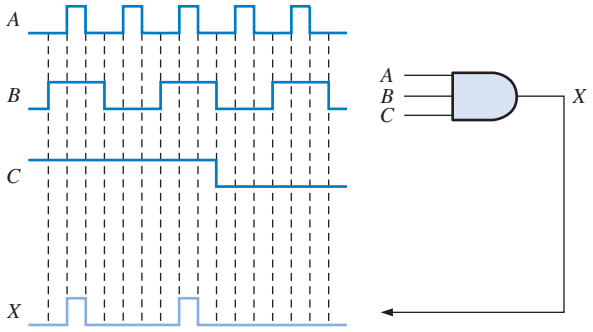
▲ FIGURA 3–12

Solução A forma de onda de saída é nível ALTO apenas quando as duas formas de onda de entrada estiverem em nível ALTO conforme mostra o diagrama de temporização.

Problema relacionado Mostre a forma de onda de saída se a entrada *B* da porta AND vista na Figura 3–12 for sempre nível ALTO.

EXEMPLO 3–5

Para a porta AND de 3 entradas mostrada na Figura 3–13, determine a forma de onda de saída em relação às entradas.



▲ **FIGURA 3–13**

Solução A forma de onda da saída *X* da porta AND de 3 entradas é nível ALTO apenas quando todas as três formas de onda de entrada (*A*, *B* e *C*) estiverem em nível ALTO.

Problema relacionado Qual é a forma de onda de saída da porta AND vista na Figura 3–13 se a entrada *C* estiver sempre em nível ALTO?

Expressões Lógicas para uma Porta AND

A função lógica AND de duas variáveis é representada matematicamente tanto colocando um ponto entre as duas variáveis, como $A \cdot B$, quanto simplesmente escrevendo as letras adjacentes sem o ponto, como AB. Normalmente usamos a representação por letras porque é mais fácil escrever.

A **multiplicação Booleana** segue as mesmas regras básicas que regem a multiplicação binária, que foi discutida no Capítulo 2 cujas regras são as seguintes:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

A multiplicação Booleana é o mesmo que a função AND.

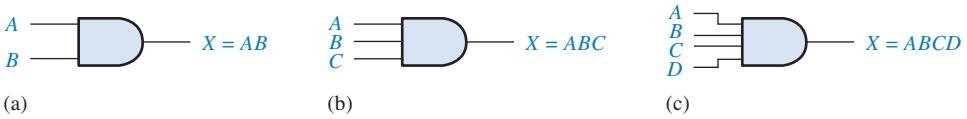
A operação realizada por uma porta AND de 2 entradas pode ser expressa na forma de equação como podemos ver a seguir: se uma variável de entrada for *A*, a outra variável for *B* e a variável de saída for *X*, então a expressão Booleana é:

$$X = AB$$

NOTA: COMPUTAÇÃO

Os computadores são capazes de utilizar todas as operações lógicas básicas quando é necessário manipular seletivamente certos bits em um ou mais bytes de dados. As manipulações seletivas de bits são feitas com máscaras. Por exemplo, para limpar (tornar todos os bits 0s) os quatro bits à direita num byte de dados, mantendo os quatro bits à esquerda, fazemos uma operação AND do byte de dados com 11110000 para obter o resultado desejado. Observe que a operação AND de qualquer bit com 1 resulta num bit que tem o mesmo valor do primeiro. Se fizermos a operação AND de 10101010 com a máscara 11110000, o resultado é 10100000.

A Figura 3–14(a) mostra o símbolo lógico da porta AND com as duas variáveis de entrada e a variável de saída indicadas.



(a) (b) (c)

▲ **FIGURA 3–14**

Expressões Booleanas para portas AND com duas, três e quatro entradas.

Quando variáveis são mostradas juntas, como em *ABC*, elas são inter-relacionadas por uma operação AND.

Para estender as expressões AND para mais de duas variáveis de entrada, use simplesmente uma nova letra para cada variável de entrada. A função de uma porta AND de três entradas, por exemplo, pode ser expressa como $X = ABC$, onde A, B e C são as variáveis de entrada. A expressão para uma porta AND de 4 entradas pode ser $X = ABCD$, e assim por diante. As partes (b) e (c) da Figura 3–14 mostram portas AND com três e quatro variáveis de entrada, respectivamente.

Podemos avaliar a operação de uma porta AND usando a expressão Booleana para a saída. Por exemplo, cada variável nas entradas pode ser 1 ou 0; assim para uma porta AND de 2 entradas, fazemos as substituições na equação de saída, $X = AB$, conforme mostra a Tabela 3–4. Essa avaliação mostra que a saída X de uma porta AND é um 1 (ALTO) apenas quando as duas entradas forem 1s (níveis ALTOS). Uma análise similar pode ser feita para qualquer número de variáveis de entrada.

▶ **TABELA 3–4**

A	B	AB = X
0	0	0 · 0 = 0
0	1	0 · 1 = 0
1	0	1 · 0 = 0
1	1	1 · 1 = 1

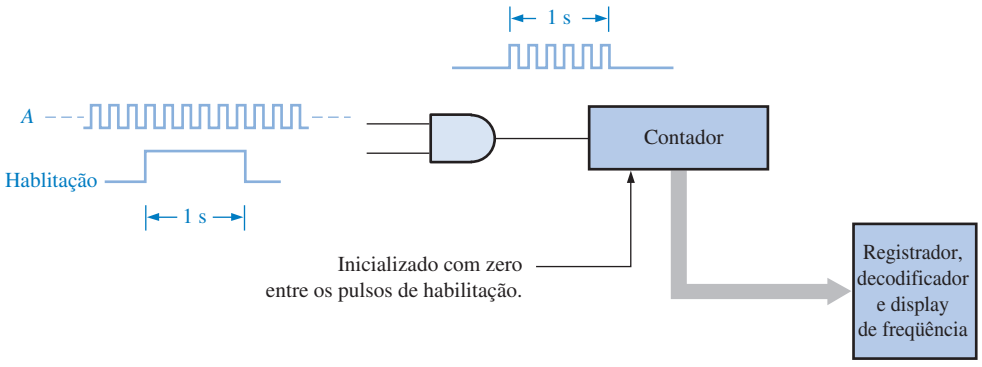
Aplicações

A Porta AND e um Dispositivo de Habilitação/Desabilitação Uma aplicação comum de uma porta AND é para **habilitar** (ou seja, permitir) a passagem de um sinal (forma de onda digital) de um ponto para outro em certos momentos e inibir (evitar) a passagem do sinal em outros momentos.

Um exemplo simples do uso particular de uma porta AND é mostrado na Figura 3–15, onde a porta AND controla a passagem de um sinal (forma de onda A) para um contador digital. A finalidade desse circuito é medir a freqüência da forma de onda A. O pulso de habilitação tem uma largura (duração) de precisamente 1s. Quando o pulso de habilitação está em nível ALTO, a forma de onda A passa pela porta chegando ao contador; e quando o pulso de habilitação está em nível BAIXO, o sinal não passa através da porta (está desabilitado).

Durante o intervalo de 1 segundo (1 s) do pulso de habilitação, os pulsos da forma de onda A passam através da porta AND para o contador. O número de pulsos que passam durante o intervalo de 1 s é igual a freqüência da forma de onda A. Por exemplo, a Figura 3–15 mostra seis pulsos em um segundo, que correspondem a uma freqüência de 6 Hz. Se 1000 pulsos passam através da porta no intervalo de 1 s do pulso de habilitação, existem 1000 pulsos/s, ou uma freqüência de 1000 Hz.

O contador conta o número de pulsos por segundo e produz uma saída binária que vai para o circuito de decodificação e display para gerar a leitura da freqüência. O pulso de habilitação se repete em intervalos determinados e uma nova contagem atualizada é feita de forma que se a freqüência variar, o novo valor será mostrado no display. Entre os pulsos de habilitação, o contador

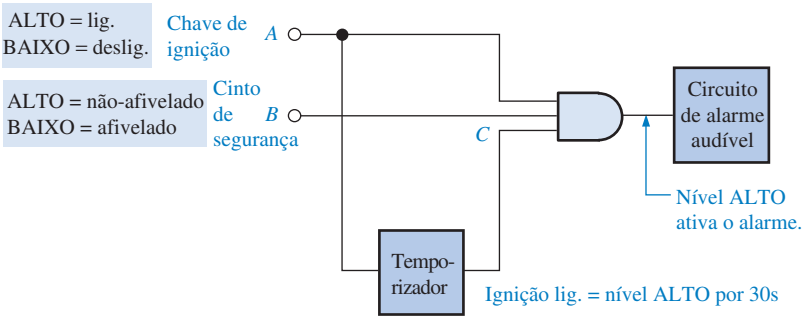


▲ **FIGURA 3–15**

Uma porta AND realizando a função de habilitação/desabilitação para um freqüencímetro.

é inicializado de forma a começar em zero cada vez que o pulso de habilitação ocorrer. O valor da freqüência atual é armazenado num registrador de forma que o display não é afetado pela inicialização do contador.

Um Sistema de Alarme para Cinto de Segurança Na Figura 3–16, uma porta AND é usada em um sistema simples de alarme para cinto de segurança de automóvel para detectar quando a chave de ignição está ligada *e* o cinto de segurança não está afivelado. Se a chave de ignição estiver ligada, um nível ALTO é produzido na entrada *A* da porta AND. Se o cinto de segurança não estiver afivelado adequadamente, um nível ALTO é produzido na entrada *B* da porta AND. Além disso, quando a chave de ignição é acionada, um temporizador é ativado produzindo um nível ALTO na entrada *C* por 30 s. Se todas as três condições estiverem presentes, ou seja, se a ignição estiver ligada e o cinto de segurança estiver afivelado e o temporizador estiver em operação, a saída da porta AND será nível ALTO e um alarme audível é acionado para lembrar o motorista.



▲ **FIGURA 3–16**

Um circuito simples de um alarme para cinto de segurança usando uma porta AND.

SEÇÃO 3–2 REVISÃO

1. Em que situação a saída de uma porta AND é nível ALTO?
2. Em que situação a saída de uma porta AND é nível BAIXO?
3. Descreva a tabela-verdade para uma porta AND de 5 entradas.

3-3 A PORTA OR

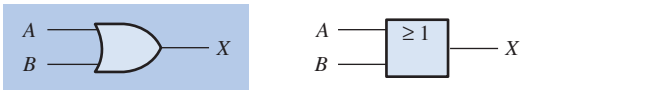
A porta OR é uma das portas básicas a partir das quais todas as funções lógicas são construídas. Uma porta OR pode ter duas ou mais entradas e realiza o que conhecemos como adição lógica.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma porta OR pelo símbolo característico ou pelo símbolo retangular ■ Descrever a operação de uma porta OR ■ Gerar a tabela-verdade para uma porta OR com um número qualquer de entradas ■ Desenhar o diagrama de temporização para uma porta OR com quaisquer formas de onda especificadas de entrada ■ Escrever a expressão lógica para um porta OR com um número qualquer de entradas ■ Discutir exemplos de aplicações da porta OR

Uma porta OR pode ter mais que duas entradas.

Uma **porta OR** tem duas ou mais entradas e uma saída, conforme indicado pelos símbolos lógicos padrão mostrados na Figura 3–17, onde estão ilustradas portas OR de duas entradas. Uma porta OR pode ter um número qualquer de entradas maior que um. Embora sejam mostrados os símbolos característico e retangular, o símbolo característico para a porta OR é o símbolo usado neste livro.



(a) Formato característico

(b) Forma retangular com o símbolo de qualificação OR (≥ 1)

▶ **FIGURA 3–17**

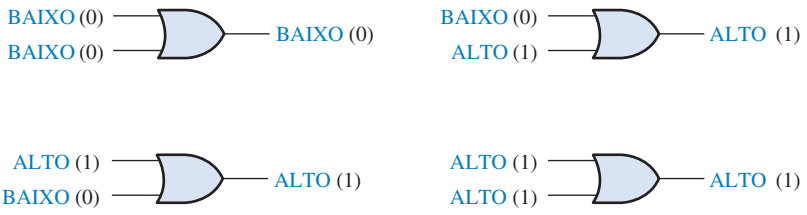
Símbolos lógicos padrão para a porta OR de duas entradas (padrão 91-1984 da ANSI/IEEE).

Operação de uma Porta OR

Uma porta OR produz um nível ALTO na saída quando *qualquer* das entradas for nível ALTO. A saída será nível BAIXO apenas quando todas as entradas estiverem em nível BAIXO. Portanto, uma porta OR determina quando uma ou mais de suas entradas estiverem em nível ALTO e produz um nível ALTO na saída dela para indicar essa condição. A Figura 3–17 mostra uma porta OR de 2 entradas, as quais são identificadas por *A* e *B* e uma saída identificada por *X*. A operação dessa porta pode ser expressa da seguinte forma:

> **Para uma porta OR de 2 entradas, a saída *X* será nível ALTO quando a entrada *A* ou a entrada *B* estiverem em nível ALTO, ou quando tanto *A* quanto *B* estiverem em nível ALTO; a saída *X* será nível BAIXO apenas quando as duas entradas *A* e *B* estiverem em nível BAIXO.**

O nível ALTO é o nível de saída ativo ou acionado para a porta OR. A Figura 3–18 ilustra a operação de uma porta OR de 2 entradas para todas as quatro combinações de entrada possíveis.



▲ **FIGURA 3–18**

Todas as combinações de níveis lógicos possíveis para uma porta OR de 2 entradas. Abra o arquivo F03-18 para verificar a operação da porta OR.

Tabela-Verdade da Porta OR

A operação de uma porta OR de 2 entradas é descrita na Tabela 3–5. Essa pode ser expandida para um número qualquer de entradas; porém independente do número de entradas, a saída será nível ALTO quando uma ou mais entradas estiverem em nível ALTO.

Para uma porta OR, se pelo menos uma entrada for nível ALTO, a saída será nível ALTO.

TABELA 3–5

Tabela-verdade para uma porta OR de 2 entradas

ENTRADAS		SAÍDA
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

1 = ALTO, 0 = BAIXO

Operação com Formas de Onda nas Entradas

Agora vamos analisar a operação de uma porta OR com formas de onda digitais nas entradas, tendo em mente a operação lógica dessa porta. Novamente, o mais importante na análise da operação da porta com formas de onda digitais é a relação temporal de todas as formas de onda envolvidas. Por exemplo, na Figura 3–19, as entradas *A* e *B* estão em nível ALTO (1) durante o intervalo de tempo t_1, fazendo com que a saída *X* seja nível ALTO (1). Durante o intervalo de tempo t_2, a entrada *A* está em nível BAIXO, porém, devido à entrada *B* estar em nível ALTO (1), a saída está em nível ALTO (1). Durante o intervalo t_3, as duas entradas estão em nível BAIXO (0), de forma que a saída está em nível BAIXO durante esse intervalo de tempo. Durante o intervalo de tempo t_4, a saída é nível ALTO (1), pois a entrada *A* está em nível ALTO (1).

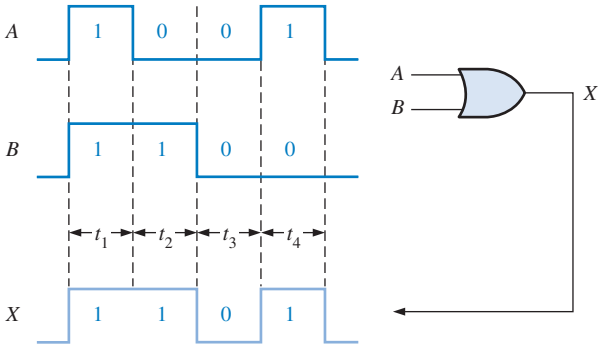


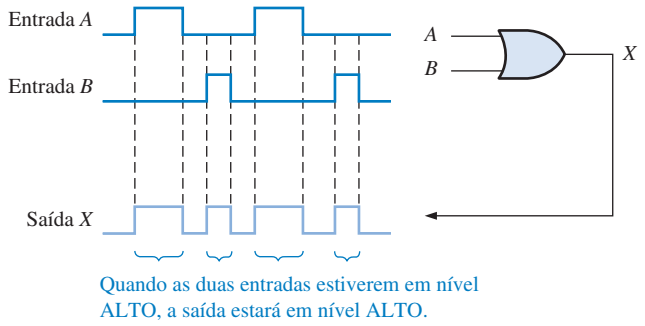
FIGURA 3–19

Exemplo da operação de uma porta OR com um diagrama de temporização mostrando as relações temporais entre entradas e saída.

Nessa ilustração, aplicamos a operação da porta OR segundo a sua tabela-verdade para cada um dos intervalos de tempo nos quais os níveis permanecem estáveis (não mudam). Os Exemplos 3–6 a 3–8 ilustram a operação da porta OR com formas de onda nas entradas.

EXEMPLO 3–6

Se as duas formas de onda de entrada, *A* e *B* (Figura 3–20), forem aplicadas na porta OR mostrada, qual é a forma de onda resultante na saída?



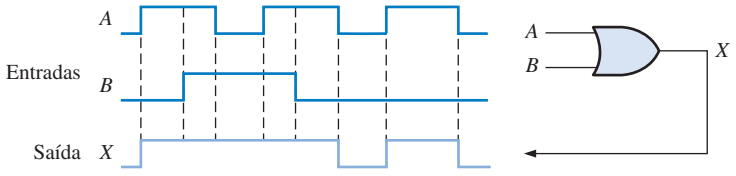
▲ FIGURA 3–20

Solução A forma de onda na saída *X* de uma porta OR de 2 entradas é nível ALTO quando uma das entradas, ou ainda ambas, estiverem em nível ALTO conforme mostra o diagrama de temporização. Nesse caso, as formas de onda das duas entradas nunca estão em nível ALTO simultaneamente.

Problema relacionado Determine a forma de onda de saída e mostre o diagrama de temporização se a entrada *A* for alterada de forma que ela seja nível ALTO a partir do início do primeiro pulso até o término do segundo pulso.

EXEMPLO 3–7

Para as formas de onda *A* e *B*, vistas na Figura 3–21, mostre a forma de onda de saída relacionando-a adequadamente as das entradas.



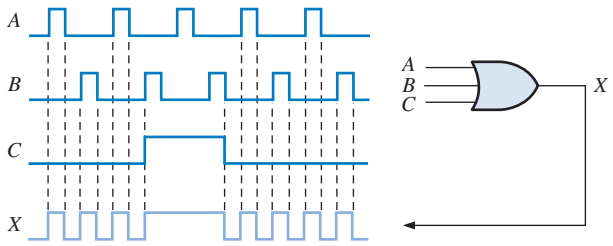
▲ FIGURA 3–21

Solução Quando uma ou ambas as entradas estiverem em nível ALTO, a saída estará em nível ALTO conforme mostra a forma de onda da saída *X* no diagrama de temporização.

Problema relacionado Determine a forma de onda de saída e mostre o diagrama de temporização se o pulso intermediário da entrada *A* for substituído por um nível BAIXO.

EXEMPLO 3–8

Para a porta OR de 3 entradas mostrada na Figura 3–22, determine a forma de onda de saída relacionando-a adequadamente com as formas de onda das entradas.



▲ **FIGURA 3–22**

Solução A saída será nível ALTO quando uma ou mais formas de onda nas entradas estiverem em nível ALTO conforme indicado pela forma de onda da saída *X* no diagrama de temporização.

Problema relacionado Determine a forma e onda de saída e mostre o diagrama de temporização se a entrada *C* estiver sempre em nível BAIXO.

Expressões Lógicas para uma Porta OR

A função lógica OR de duas variáveis é representada matematicamente por um sinal "+" entre as duas variáveis, por exemplo, $A + B$.

Quando variáveis são separadas pelo sinal +, elas estão submetidas à operação OR.

A adição na álgebra Booleana envolve variáveis cujos valores são o binário 1 ou o binário 0. As regras básicas para a **adição Booleana** são:

$$0 + 0 = 0$$
$$0 + 1 = 0$$
$$1 + 0 = 0$$
$$1 + 1 = 1$$

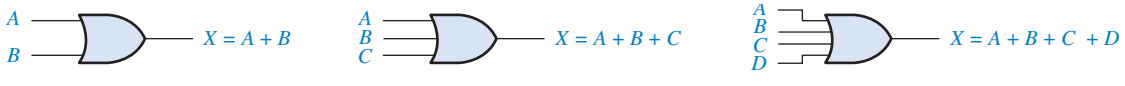
A adição Booleana é o mesmo que a função OR.

Observe que a adição Booleana difere da adição binária no caso em que dois 1s são somados. Não existe carry na adição Booleana.

A operação de uma porta OR de 2 entradas pode ser expressa como segue: se uma variável de entrada for *A*, se a outra variável for *B* e se a variável de saída dor *X*, a expressão Booleana é:

$$X = A + B$$

A Figura 3–23(a) mostra o símbolo lógico da porta OR com as identificações das duas variáveis de entrada e da variável de saída.



(a) (b) (c)

▲ **FIGURA 3–23**

Expressões Booleanas para portas OR de duas, três e quatro entradas.

NOTA: COMPUTAÇÃO

Uma outra operação de máscara usada na programação de computadores para tornar igual a 1 (setar) determinados bits de um byte de dados sem afetar quaisquer outros bits, é feita com uma operação OR. Utiliza-se uma máscara que contém 1s nas posições onde os bits de dados devem ser setados. Por exemplo, se queremos forçar para que o bit mais significativo de um byte de dados seja igual a 1, porém mantendo os bits restantes inalterados, podemos implementar uma operação OR do byte de dados com a máscara 10000000.

Para estender a operação OR para mais que duas variáveis de entrada, um nova letra é usada para cada variável adicional. Por exemplo, a função de uma porta OR de 3 entradas pode ser expressa como $X = A + B + C$. A expressão para uma porta OR de 4 entradas pode ser escrita como $X = A + B + C + D$, e assim por diante. As partes (b) e (c) da Figura 3–23 mostram portas OR com três e quatro variáveis de entrada, respectivamente.

A operação da porta OR pode ser determinada usando as expressões Booleanas para a saída X substituindo todas as combinações possíveis de 1 e 0 nas variáveis de entrada, conforme mostra a Tabela 3–6 para uma porta OR de 2 entradas. Esse cálculo mostra que a saída X de uma porta OR é um 1 (ALTO) quando qualquer uma ou mais de uma das entradas for 1 (ALTO). Uma análise similar pode ser estendida para portas OR com um número qualquer de variáveis de entrada.

TABELA 3–6

A	B	$A + B = X$
0	0	$0 + 0 = 0$
0	1	$0 + 1 = 1$
1	0	$1 + 0 = 1$
1	1	$1 + 1 = 1$

Uma Aplicação

Uma seção simplificada de um sistema de alarme para detecção de intrusão é mostrado na Figura 3–24. Esse sistema pode ser usado num ambiente de uma casa (um ambiente com duas janelas e uma porta). Os sensores são chaves magnéticas que produzem uma saída em nível ALTO quando abertas e em nível BAIXO quando fechadas. Enquanto as janelas e a porta estiverem fechadas, as chaves estão fechadas e todas as três entradas da porta OR estarão em nível BAIXO. Quando uma das janelas ou a porta for aberta, será produzido um nível ALTO numa entrada da porta OR e a saída dessa porta vai para nível ALTO. Isso gera a ativação e memorização num circuito de alarme para advertir a intrusão.

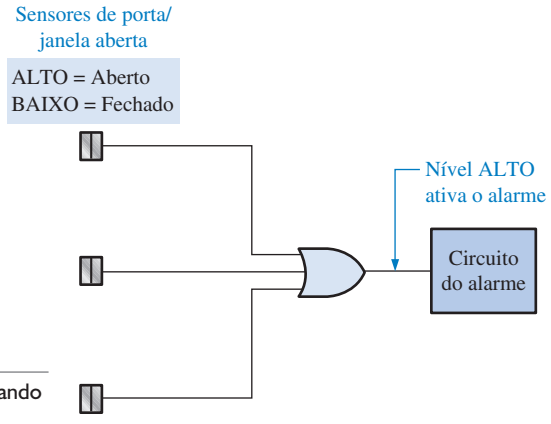


FIGURA 3–24

Um sistema simplificado de detecção de intrusão usando uma porta OR.

SEÇÃO 3–3 REVISÃO

1. Em que situação a saída de uma porta OR é nível ALTO?
2. Em que situação a saída de uma porta OR é nível BAIXO?
3. Descreva a tabela-verdade de uma porta OR de 3 entradas.

3-4 A PORTA NAND

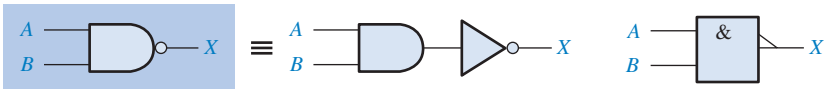
A porta NAND é um elemento lógico popular porque ela pode ser usada como uma porta universal; ou seja, as portas NAND podem ser usadas em combinação para realizarem operações AND, OR e inversão. A propriedade universal da porta NAND será analisada em detalhes no Capítulo 5.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma porta NAND pelo seu símbolo característico ou pelo seu símbolo retangular ■ Descrever a operação de uma porta NAND ■ Desenvolver a tabela-verdade para uma porta NAND com um número qualquer de entradas ■ Desenhar um diagrama de temporização para uma porta NAND com quaisquer formas de onda de entrada especificadas ■ Escrever a expressão lógica para uma porta NAND com um número qualquer de entradas ■ Descrever a operação da porta NAND em termos do seu equivalente OR negativo ■ Discutir exemplos de aplicações de portas NAND

O termo NAND é uma contração de NOT-AND e implica numa função AND com uma saída complementada (invertida). O símbolo lógico padrão para uma porta NAND de duas entradas e o seu equivalente com uma porta AND seguida de um inversor são mostrados na Figura 3–25(a), onde o símbolo ≡ significa equivalência. O símbolo retangular é mostrado na parte (b).

A porta NAND é o mesmo que uma AND exceto que a saída é invertida.



(a) Formato característico de uma porta NAND de 2 entradas e o seu equivalente NOT/AND

(b) Formato retangular de uma porta NAND de 2 entradas com o indicador de polaridade

▲ **FIGURA 3–25**

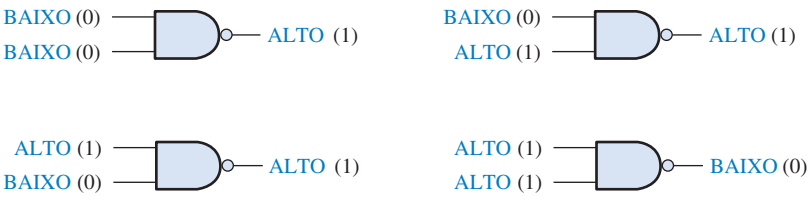
Símbolos lógicos padrões para a porta AND de duas entradas (padrão 91-1984 da ANSI/IEEE).

Operação de uma Porta NAND

Uma **porta NAND** produz uma saída de nível BAIXO apenas quando todas as entradas estiverem em nível ALTO. Quando qualquer uma das entradas for nível BAIXO, a saída será nível ALTO. Para o caso específico de uma porta NAND de 2 entradas, conforme mostra a Figura 3–25 com as entradas indicadas por *A* e *B* e a saída por *X*, a operação dela pode ser expressa como a seguir:

Para uma porta NAND de 2 entradas, a saída *X* será nível BAIXO apenas quando as entradas *A* e *B* estiverem em nível ALTO; *X* será nível ALTO quando *A* ou *B* for nível BAIXO ou ainda quando *A* e *B* estiverem em nível BAIXO.

Observe que essa operação é oposta a da AND em termos do nível lógico de saída. Numa porta NAND, o nível BAIXO (0) é o nível ativo ou acionado da saída, conforme indicado pelo pequeno círculo na saída. A Figura 3–26 ilustra a operação de uma porta NAND para todas as quatro combinações de entrada e a Tabela 3–7 é a tabela-verdade que resume a operação lógica de uma porta NAND de 2 entradas.



▲ **FIGURA 3–26**

Operação de uma porta NAND de 2 entradas. Abra o arquivo F03-26 para verificar a operação da porta NAND.

▶ **TABELA 3–7**

Tabela-verdade para uma porta NAND de 2 entradas

ENTRADAS		SAÍDA
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

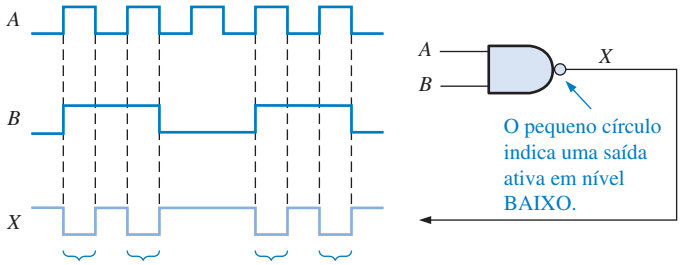
1 = ALTO, 0 = BAIXO

Operação com Formas de Onda nas Entradas

Agora analisaremos a operação de uma porta NAND com formas de onda digitais nas entradas. Lembre-se, da tabela-verdade, de que a única vez que a saída é nível BAIXO ocorre quando todas as entradas estão em nível ALTO.

EXEMPLO 3–9

Se as duas formas de onda *A* e *B* mostradas na Figura 3–27 forem aplicadas nas entradas de uma porta NAND, determine a forma de onda de saída resultante.



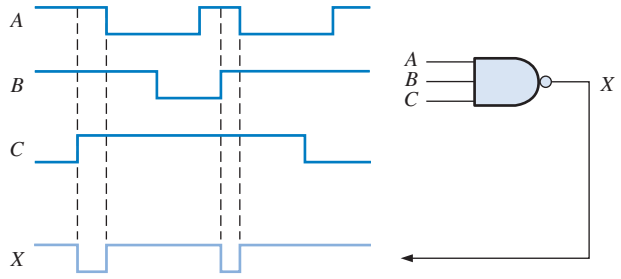
▲ **FIGURA 3–27**

Solução A forma de onda na saída *X* é nível BAIXO apenas durante os quatro intervalos de tempo em que as duas formas de onda *A* e *B* estão em nível ALTO conforme mostra o diagrama de temporização.

Problema relacionado Determine a forma de onda de saída e mostre o diagrama de temporização se a forma de onda da entrada *B* for invertida.

EXEMPLO 3–10

Mostre a forma de onda de saída para uma porta NAND de 3 entradas, conforme a Figura 3–28, estabelecendo a relação temporal com as entradas.



▲ FIGURA 3–28

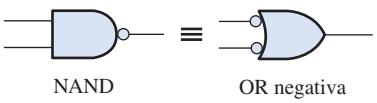
Solução A forma de onda da saída *X* é nível BAIXO apenas quando todas as três formas de onda das entradas estiverem em nível ALTO como mostra o diagrama de temporização.

Problema relacionado Determine a forma de onda de saída e mostre o diagrama de temporização se a forma de onda da entrada *A* for invertida.

Operação Equivalente OR Negativa de uma Porta NAND Inerente à operação da porta NAND é o fato de que uma ou mais entradas em nível BAIXO, produzem uma saída de nível ALTO. A Tabela 3–7 mostra que a saída *X* será nível ALTO (1) quando qualquer uma das entradas, *A* e *B*, for nível BAIXO (0). A partir dessa análise, uma porta NAND pode ser usada para implementar uma operação OR que necessita que uma ou mais entradas de nível BAIXO produza uma saída de nível ALTO. Esse aspecto da operação NAND é referenciado como **OR negativa**. O termo negativa nesse contexto significa que as entradas são definidas para estarem no estado ativo ou acionado quando em nível BAIXO.

Para uma porta NAND de 2 entradas realizando uma operação OR negativa, a saída *X* será nível ALTO quando a entrada *A* ou *B* for nível BAIXO ou quando ambas as entradas estiverem em nível BAIXO.

Quando uma porta NAND é usada para detectar uma ou mais entradas de nível BAIXO em vez de ser todas em nível ALTO, ela está realizando a operação OR negativa e é representada pelo símbolo lógico padrão mostrado na Figura 3–29. Embora os dois símbolos mostrados na Figura 3–29 representem a mesma porta física, eles servem para definir o seu papel ou modo de operação numa aplicação particular, conforme ilustrado pelos Exemplos 3–11 a 3–13.



◀ FIGURA 3–29

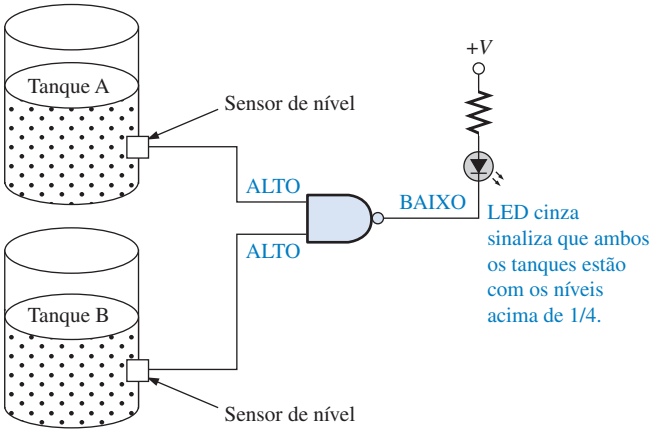
Símbolos-padrão representando duas operações equivalentes de uma porta NAND.

EXEMPLO 3–11

Uma planta de fabricação usa dois tanques para armazenar certos produtos químicos líquidos necessários num processo de fabricação. Cada tanque tem um sensor que detecta quando o nível do produto cai para 25% do nível máximo. Os sensores produzem um nível ALTO de 5 V quando os tanques estão com mais que 1/4 da capacidade. Quando o volume do produto no tanque cai para 1/4 do máximo, o sensor gera um nível BAIXO de 0 V.

É necessário que um diodo emissor de luz (LED – *light emitting diode*) cinza num painel indicador mostre quando ambos os tanques estão acima de 1/4 da capacidade. Mostre como uma porta NAND pode ser usada para implementar essa função.

Solução A Figura 3–30 mostra uma porta NAND com as suas duas entradas conectadas aos sensores de nível do tanque e a sua saída conectada a um painel indicador. A operação pode ser descrita como a seguir: Se o tanque *A* e o tanque *B* estão com os níveis acima de 1/4, o LED está ligado.



▲ **FIGURA 3–30**

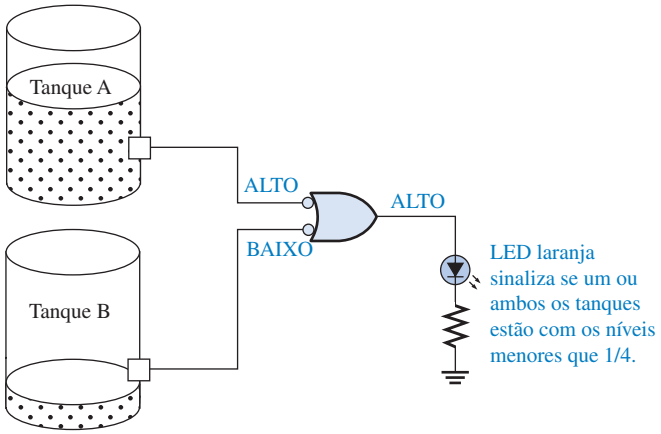
Enquanto as saídas dos dois sensores estão em nível ALTO (5 V), indicando que os dois tanques estão com mais que 1/4 da capacidade, a saída da porta NAND estará em nível BAIXO (0 V). O circuito do LED cinza está configurado de forma que uma tensão de nível BAIXO ative-o.

Problema relacionado Como o circuito visto na Figura 3–30 poderia ser modificado para monitorar os níveis de três tanques em vez de dois?

EXEMPLO 3–12

O supervisor do processo produtivo descrito no Exemplo 3–11 preferiria ter um LED laranja para indicar quando pelo menos um dos tanques estiver abaixo de 1/4 da capacidade em vez de usar um LED cinza para indicar quando os dois tanques estão acima de 1/4 da capacidade. Mostre como essa solicitação pode ser implementada.

Solução A Figura 3–31 mostra uma porta NAND operando como uma porta OR negativa para detectar a ocorrência de pelo menos um nível BAIXO nas entradas. O sensor produz uma tensão de nível BAIXO se o volume do tanque, ao qual está conectado, for igual ou menor que 1/4 da capacidade. Quando isso acontece, a saída da porta vai para nível ALTO. O circuito do LED laranja no painel está configurado de forma que uma tensão de nível ALTO liga o LED. A operação pode ser descrita da seguinte forma: Se o tanque *A* ou o tanque *B*, ou ainda ambos, estiverem abaixo de 1/4 da capacidade, o LED ligará.



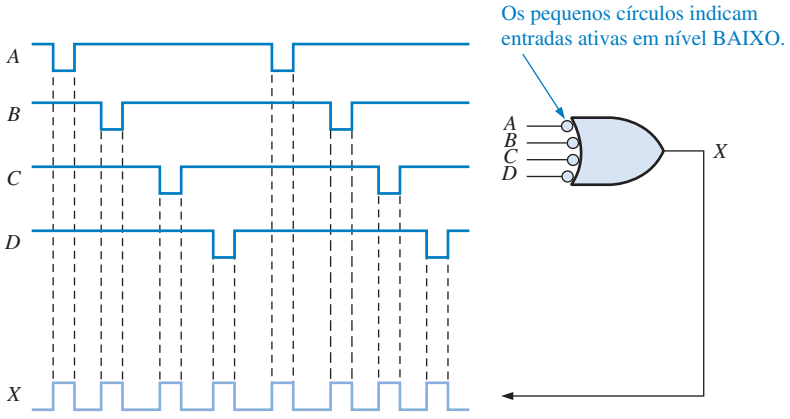
▲ **FIGURA 3–31**

Observe que, aqui e no Exemplo 3–11, a mesma porta NAND de 2 entradas é utilizada, porém, é usado no diagrama um símbolo de porta diferente, ilustrando as formas nas quais as operações NAND e OR negativa equivalente são usadas.

Problema relacionado Como o circuito mostrado na Figura 3–31 pode ser modificado para monitorar quatro tanques em vez de dois?

EXEMPLO 3–13

Para a porta NAND de 4 entradas vista na Figura 3–32, operando como uma OR negativa, determine a saída em relação às entradas dadas.



▲ **FIGURA 3–32**

Solução A forma de onda na saída *X* será nível ALTO em qualquer momento que a forma de onda numa entrada for nível BAIXO conforme mostra o diagrama de temporização.

Problema relacionado Determine a forma de onda de saída se a forma de onda na entrada *A* for invertida antes de ser aplicada na porta.

Expressões Lógicas para uma Porta NAND

Uma barra sobre uma variável ou variáveis indica uma inversão.

A expressão Booleana para a saída de uma porta NAND de 2 entradas é

$$X = \overline{AB}$$

Essa expressão diz que as duas variáveis e entrada, *A* e *B*, são submetidas a uma operação AND e em seguida esta é complementada, conforme indicado pela barra sobre a expressão AND. Essa é a descrição na forma de equação da operação de uma porta NAND de duas entradas. Calculando essa expressão para todos os valores possíveis para as variáveis de entrada, obtemos o resultado mostrado na Tabela 3–8.

▶ **TABELA 3–8**

A	*B*	$\overline{AB} = X$
0	0	$\overline{0 \cdot 0} = \overline{0} = 1$
0	1	$\overline{0 \cdot 1} = \overline{0} = 1$
1	0	$\overline{1 \cdot 0} = \overline{0} = 1$
1	1	$\overline{1 \cdot 1} = \overline{1} = 0$

Uma vez determinada a expressão para uma dada função lógica, essa função pode ser calculada para todos os valores possíveis das variáveis. Esse cálculo nos diz exatamente qual é a saída do circuito lógico para cada uma das condições de entrada, nos dando portanto a descrição completa da operação lógica do circuito. A expressão NAND pode ser estendida para mais de duas variáveis de entrada incluindo letras adicionais para representar as outras variáveis.

SEÇÃO 3–4 REVISÃO

1. Quando uma saída de uma porta NAND é nível BAIXO?
2. Quando uma saída de uma porta NAND é nível ALTO?
3. Descreva as diferenças funcionais entre uma porta NAND e uma porta OR negativa. Elas têm a mesma tabela-verdade?
4. Escreva a expressão de saída para uma porta NAND com entradas *A*, *B* e *C*.

3-5 A PORTA NOR

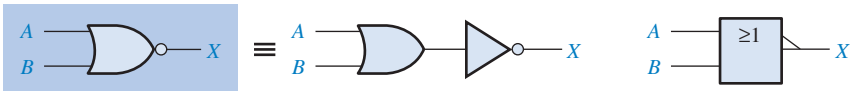
A porta NOR, assim como a porta NAND, é um elemento lógico útil porque ela também pode ser usada como uma porta universal; ou seja, as portas NOR podem ser usadas em combinação para realizarem as operações AND, OR e inversão. A propriedade universal da porta NOR será examinada detalhadamente no Capítulo 5.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma porta NOR pelo seu símbolo característico ou pelo seu símbolo retangular ■ Descrever a operação de uma porta NOR ■ Desenvolver a tabela-verdade para uma porta NOR com um número qualquer de entradas ■ Desenhar um diagrama de temporização para uma porta NOR para quaisquer formas de onda de entrada especificadas ■ Escrever a expressão lógica para uma porta NOR com um número qualquer de entradas ■ Descrever a operação de uma porta NOR em termos da AND negativa equivalente ■ Discutir exemplos de aplicações de porta NOR

O termo NOR é a contração de NOT-OR e implica numa função OR com a saída invertida (complementada). O símbolo lógico padrão para uma porta NOR de 2 entradas e o circuito equivalente com uma OR seguida de um inversor são mostrados na Figura 3–33(a). O símbolo retangular é mostrado na parte (b).

O NOR é o mesmo que o OR, exceto que a saída é invertida.



(a) Formato característico de uma porta NOR de 2 entradas e o seu equivalente NOT/OR

(b) Formato retangular de uma porta NOR de 2 entradas com indicador de polaridade

▲ **FIGURA 3–33**

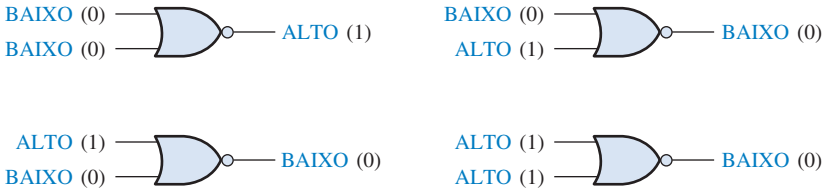
Símbolos lógicos padrão da porta NOR (padrão 91-1984 da ANSI/IEEE).

Operação de uma Porta NOR

Uma **porta NOR** produz uma saída de nível BAIXO quando qualquer uma de suas entradas for nível ALTO. Apenas quando todas as suas entradas estiverem em nível BAIXO é que a saída será nível ALTO. Para o caso específico de uma porta NOR de 2 entradas, conforme mostra a Figura 3–33 com as entradas identificadas por *A* e *B* e a saída identificada por *X*, a operação dessa porta é a seguinte:

Para uma porta NOR de 2 entradas, a saída *X* será nível BAIXO quando a entrada *A* ou a entrada *B* for nível ALTO, ou quando ambas as entradas estiverem em nível ALTO; a saída *X* será nível ALTO apenas quando as entradas *A* e *B* estiverem em nível BAIXO.

Essa operação resulta numa saída de nível oposto ao da porta OR. Numa porta NOR, uma saída de nível BAIXO está no nível de saída ativo ou acionado conforme indicado pelo pequeno círculo na saída. A Figura 3–34 ilustra a operação de uma porta NOR de 2 entradas para todas as quatro combinações possíveis de entrada e a Tabela 3–9 mostra a tabela-verdade para uma porta NOR de 2 entradas.



▲ **FIGURA 3–34**

Operação de uma porta NOR de 2 entradas. Abra o arquivo F03-34 para verificar a operação da porta NOR

◀ **TABELA 3–9**

Tabela-verdade para uma porta NOR de 2 entradas

ENTRADAS		SAÍDAS
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

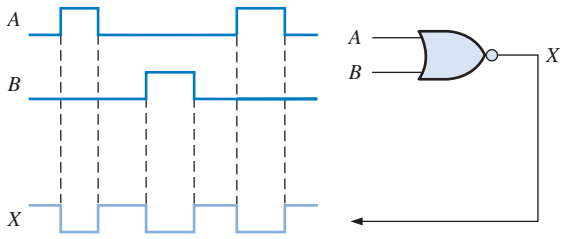
1 = nível ALTO, 0 = nível BAIXO

Operação com Formas de Onda nas Entradas

Os próximos dois exemplos ilustram a operação de uma porta NOR com formas de onda digitais nas entradas. Novamente, assim como com outros tipos de portas, seguiremos a operação da tabela-verdade para determinar as formas de onda de saída relacionando-as adequadamente no tempo com as formas de onda das entradas.

EXEMPLO 3–14

Se as duas formas de onda mostradas na Figura 3–35 são aplicadas em uma porta NOR, qual é a forma de onda de saída resultante?



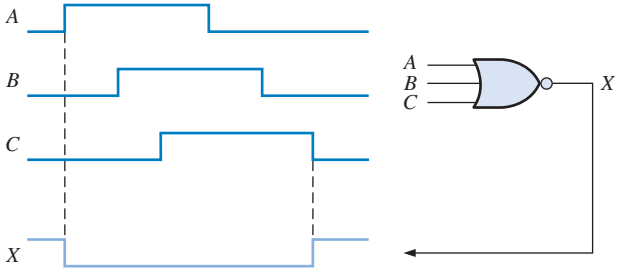
▲ FIGURA 3–35

Solução Todas as vezes que qualquer entrada de uma porta NOR for nível ALTO, a saída será nível BAIXO, conforme mostra a forma de onda da saída *X* no diagrama de temporização.

Problema relacionado Inverta a entrada *B* e determine a forma de onda da saída em relação às entradas.

EXEMPLO 3–15

Mostre a forma de onda de saída para a porta NOR de 3 entradas vista na Figura 3–36 com a relação temporal adequada com as entradas.



▲ FIGURA 3–36

Solução A saída *X* é nível BAIXO quando qualquer entrada for nível ALTO, conforme mostra a forma de onda da saída *X* no diagrama de temporização.

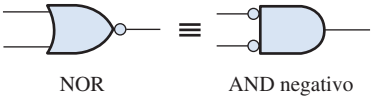
Problema relacionado Com as entradas *B* e *C* invertidas, determine a saída e mostre o diagrama de temporização.

Operação Equivalente AND Negativa de uma Porta NOR Uma porta NOR, assim como uma NAND, tem um outro aspecto de sua operação que é inerente à forma com que ela funciona logicamente. A Tabela 3–9 mostra que um nível ALTO é produzido na saída da porta apenas quando todas as entradas estiverem em nível BAIXO. A partir dessa análise, uma porta NOR pode ser usada para uma operação AND que necessita que todas as entradas estejam em nível BAIXO para produzir uma saída em nível ALTO. Esse aspecto da operação NOR é denominado **AND nega-**

tiva. O termo *negativa* nesse contexto significa que as entradas são definidas para estarem no estado ativo ou acionado quando em nível BAIXO.

Para uma porta NOR de 2 entradas realizando uma operação AND negativa, a saída *X* será nível ALTO apenas quando as entradas *A* e *B* estiverem em nível baixo.

Quando uma porta NOR é usada para detectar todos os níveis BAIXOs em suas entradas, em vez de um ou mais níveis ALTOs, ela está realizando uma operação AND negativa e é representada pelo símbolo padrão mostrado na Figura 3–37. É importante lembrar que os dois símbolos mostrados na Figura 3–37 representam fisicamente a mesma porta servindo apenas para fazer distinção entre os dois modos de operação. Os próximos três exemplos ilustram isso.

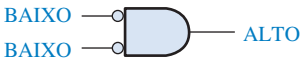


◀ **FIGURA 3–37**
Símbolos padrão que representam as duas operações equivalentes de uma porta NOR.

EXEMPLO 3–16

Um dispositivo é necessário para indicar quando dois níveis BAIXOs ocorrem simultaneamente nas entradas para produzir uma saída de nível ALTO como uma indicação. Especifique o dispositivo.

▶ **FIGURA 3–38**



Solução Uma porta NOR de 2 entradas operando como uma porta AND negativa faz-se necessária para produzir uma saída de nível ALTO quando as duas entradas estiverem em nível BAIXO, conforme mostra a Figura 3–38.

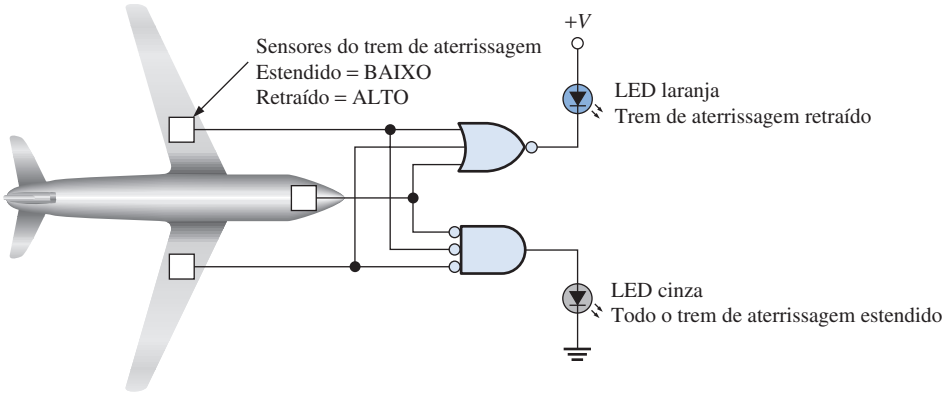
Problema relacionado Um dispositivo é necessário para indicar quando ocorre um ou dois níveis ALTOs em suas entradas e produzir um nível BAIXO na saída como uma indicação. Especifique o dispositivo.

EXEMPLO 3–17

Como parte de um sistema de monitoramento funcional de aeronaves, é necessário um circuito para indicar o estado do trem de aterrissagem antes da aterrissagem. Um LED cinza liga se os três trens de aterrissagem estiverem adequadamente estendidos quando a chave de redução de velocidade for ativada em preparação para a aterrissagem. Um LED laranja liga se algum dos trens de aterrissagem não for adequadamente estendido antes da aterrissagem. Quando o trem de aterrissagem está estendido, o seu sensor produz uma tensão de nível BAIXO. Quando o trem de aterrissagem está retraído, o seu sensor produz uma tensão de nível ALTO. Implemente um circuito que atenda a esse requisito.

Solução O circuito é energizado apenas quando a chave de acionamento do trem de aterrissagem for ativada. Use uma porta NOR para cada um dos dois requisitos conforme mostra a Figura 3–39. Uma porta NOR opera como uma AND negativa para detectar um nível BAIXO a partir de cada um dos três sensores situados nos três trens de aterrissagem. Quando as três entradas da porta estiverem em nível BAIXO, os três trens de aterrissagem estão adequadamente estendidos resultando numa saída de nível ALTO a partir da porta AND negativa ligando o LED cinza. A outra porta NOR opera como uma NOR para detectar se um ou mais trens de aterrissagem permanecem retraídos quando a chave de acionamento do trem de aterrissagem estiver ativada. Quando um ou mais trens de ater-

rissagem permanecerem retraídos, o nível ALTO resultante a partir do sensor é detectado pela porta NOR, a qual produz uma saída de nível BAIXO para ligar o LED laranja de advertência.



▲ FIGURA 3–39

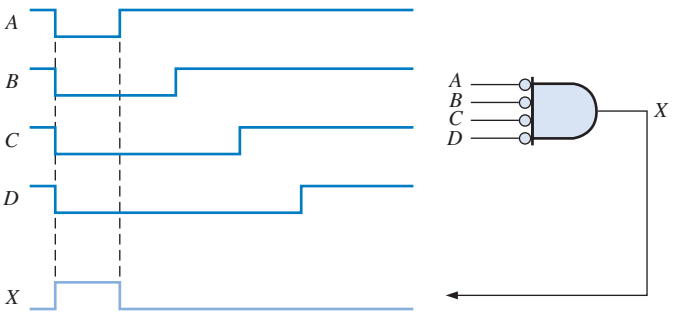
Problema relacionado Que tipo de porta deve ser usada para detectar se todos os três trens de aterrissagem estão retraídos após levantar vôo, considerando que é necessário uma saída em nível BAIXO para ativar um LED?

DICA PRÁTICA

Quando for acionar uma carga como um LED através de uma porta lógica, consulte as folhas de dados do fabricante para saber a capacidade máxima de acionamento (corrente de saída) do dispositivo. Uma porta lógica num CI comum pode não ser capaz de operar a corrente necessária requerida por certas cargas tais como LEDs. Existem disponíveis muitos tipos de CIs de portas lógicas nos quais as saídas possuem buffers como saídas de coletor aberto (OC – *open colector*) e dreno aberto (OD – *open drain*). A capacidade de corrente de saída de um CI típico de portas lógicas é limitada a μA ou uma faixa relativamente pequena de mA. Por exemplo, a tecnologia TTL padrão pode operar com correntes de saída de até 16 mA. A maioria dos LEDs requer correntes na faixa de 10 mA a 50 mA.

EXEMPLO 3–18

Para a porta NOR de 4 entradas operando como uma AND negativa (Figura 3–40), determine a saída relacionando-a às entradas.



▲ FIGURA 3–40

Solução Todas as vezes que todas as entradas estiverem em nível BAIXO, a saída estará em nível ALTO como podemos ver na forma de onda da saída *X* no diagrama de temporização.

Problema relacionado Determine a saída para a porta vista na Figura 3–40, sendo que a entrada *D* é invertida, e mostre o diagrama de temporização.

Expressões Lógicas para uma Porta NOR

A expressão Booleana para a saída de uma porta NOR de 2 entradas pode ser escrita como a seguir:

$$X = \overline{A + B}$$

Essa equação diz que as duas variáveis de entrada são submetidas a uma operação OR e então esta é complementada, conforme indicado pela barra sobre a expressão OR. Calculando essa expressão, obtemos o resultado mostrado na Tabela 3–10. A expressão NOR pode ser estendida para mais de duas variáveis de entrada incluindo letras adicionais para representar as outras variáveis.

◀ **TABELA 3–10**

A	*B*	$\overline{A+B}=X$
0	0	$\overline{0+0} = \overline{0} = 1$
0	1	$\overline{0+1} = \overline{1} = 0$
1	0	$\overline{1+0} = \overline{1} = 0$
1	1	$\overline{1+1} = \overline{1} = 0$

SEÇÃO 3–5 REVISÃO

1. Quando a saída de uma porta NOR é nível ALTO?
2. Quando a saída de uma porta NOR é nível BAIXO?
3. Descreva a diferença funcional entre uma porta NOR e uma porta AND negativa. Ambas têm a mesma tabela-verdade?
4. Escreva a expressão de saída para uma porta NOR de 3 entradas sendo as variáveis de entrada *A*, *B* e *C*.

3-6 AS PORTAS OR EXCLUSIVO E NOR EXCLUSIVO

As portas OR exclusivo (EX-OR) e NOR exclusivo (EX-NOR) são formadas pela combinação de outras portas já estudadas, conforme veremos no Capítulo 5. Entretanto, devido à importância fundamental dessas portas em muitas aplicações, elas são tratadas como elementos lógicos básicos tendo seus próprios símbolos lógicos.

Ao final do estudo desta seção você deverá ser capaz de:

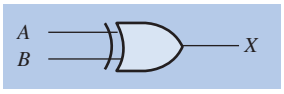
■ Identificar as portas EX-OR e EX-NOR pelos seus símbolos característicos ou pelos seus símbolos retangulares ■ Descrever a operação das portas EX-OR e EX-NOR ■ Mostrar a tabela-verdade para as portas EX-OR e EX-NOR ■ Desenhar um diagrama de temporização para uma porta EX-OR e EX-NOR com quaisquer formas de onda de entrada especificadas ■ Discutir exemplos de aplicações de portas EX-OR e EX-NOR

NOTA: COMPUTAÇÃO

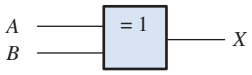
As portas EX-OR conectadas para formar um circuito somador permite um computador realizar adição, subtração, multiplicação e divisão numa unidade de lógica e aritmética (ALU – *arithmetic logic unit*). Uma porta EX-OR combina as lógicas básicas AND, OR e NOT.

A Porta EX-OR

Os símbolos padrão para a porta OR exclusivo (EX-OR) são mostrados na Figura 3–41. A porta EX-OR tem apenas duas entradas.



(a) Formato característico



(b) Formato retangular com a EX-OR

▲ **FIGURA 3–41**

Símbolos lógicos padrão para a porta EX-OR.

Para uma porta EX-OR, entradas opostas fazem com que a saída seja nível ALTO.

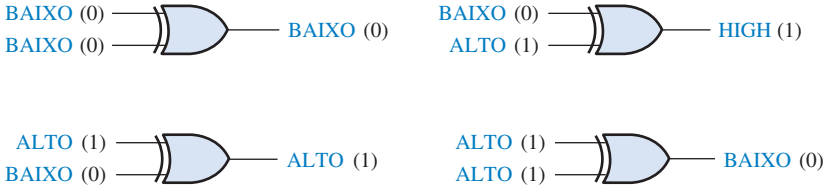
A saída de uma **porta OR exclusivo** (EX-OR) é nível ALTO *apenas* quando as duas entradas estão em níveis lógicos opostos. Essa operação pode ser expressa, com referência às entradas *A* e *B* e à saída *X*, como a seguir:

Para uma porta EX-OR, a saída *X* é nível ALTO quando a entrada A for nível BAIXO e a entrada *B* for nível ALTO ou quando a entrada *A* for nível ALTO e a entrada *B* for nível BAIXO; a saída *X* é nível BAIXO quando *A* e *B* forem ambas nível ALTO ou ambas nível BAIXO.

As quatro combinações possíveis de entrada e as saídas resultantes para uma porta EX-OR são ilustradas na Figura 3–42. O nível ALTO é o nível ativo ou acionado da saída e ocorre apenas quando as entradas estão em níveis opostos. A operação de uma porta EX-OR está resumida na tabela-verdade mostrada na Tabela 3–11.

▶ **FIGURA 3–42**

Todos os níveis lógicos possíveis para uma porta EX-OR. Abra o arquivo F03-42 para verificar a operação da porta EX-OR.



▶ **TABELA 3–11**

Tabela-verdade para uma porta EX-OR

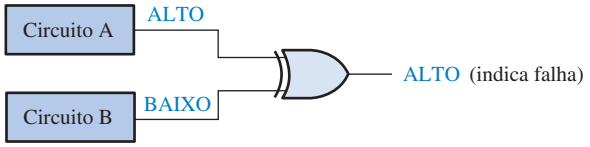
ENTRADAS		SAÍDA
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

EXEMPLO 3–19

Um certo sistema contém dois circuitos idênticos operando em paralelo. Enquanto operam adequadamente, as saídas de ambos os circuitos são sempre as mesmas. Caso um dos circuitos tenha algum problema, as saídas terão níveis opostos em algum momento. Projete um sistema para detectar que uma falha ocorreu em um dos circuitos.

Solução As saídas dos circuitos são conectadas às entradas de uma porta EX-OR, conforme mostra a Figura 3–43. Uma falha em qualquer um dos circuitos produz saídas diferentes, fa-

zendo com que as entradas da EX-OR tenham níveis opostos. Essa condição produz um nível ALTO na saída da porta EX-OR, indicando uma falha num dos circuitos.



▲ **FIGURA 3–43**

Problema relacionado Considerando o sistema acima, a porta EX-OR sempre detectará falhas simultâneas nos circuitos? Caso contrário, sob que condição?

A Porta EX-NOR

Os símbolos padrão para uma **porta NOR exclusivo** (EX-NOR) são mostrados na Figura 3–44. O pequeno círculo na saída da porta EX-NOR indica que sua saída é oposta a da porta EX-OR. Quando os dois níveis lógicos de entrada são opostos, a saída de uma porta EX-NOR é nível BAIXO. A operação pode ser expressa como segue (*A* e *B* são as entradas e *X* é a saída):

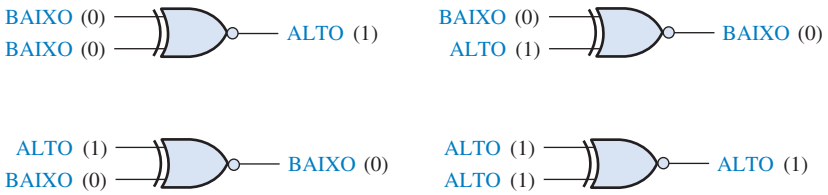
Para uma porta EX-NOR, a saída *X* é nível BAIXO quando a entrada *A* for nível BAIXO e a entrada *B* for nível ALTO, ou quando *A* for nível ALTO e *B* for nível BAIXO. A saída *X* é nível ALTO quando *A* e *B* estiverem ambas em nível ALTO ou ambas em nível BAIXO.

(a) Formato característico (b) Formato retangular

▲ **FIGURA 3–44**

Símbolos lógicos padrão para a porta EX-NOR.

As quatro combinações possíveis de entrada e as saídas resultantes para uma porta EX-NOR são mostradas na Figura 3–45. A operação de uma porta EX-NOR está resumida na Tabela 3–12. Observe que a saída é nível ALTO quando níveis iguais estão nas entradas.



▲ **FIGURA 3–45**

Todos os níveis lógicos possíveis para uma porta EX-NOR. Abra o arquivo F03-45 para verificar a operação da porta EX-NOR.

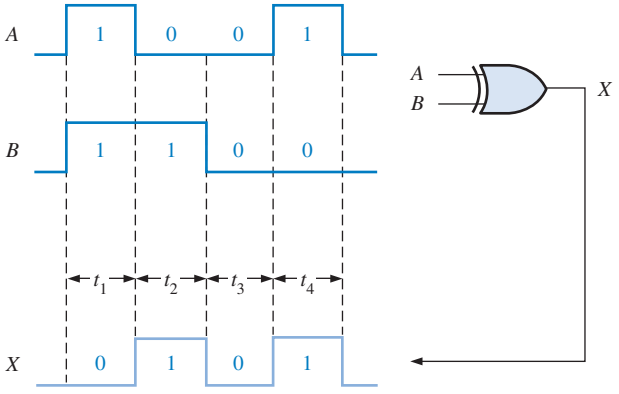
▶ **TABELA 3–12**

Tabela-verdade para uma porta EX-NOR

ENTRADAS		SAÍDA
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

Operação com Formas de Ondas nas Entradas

Assim como fizemos com as outras portas, vamos examinar a operação das portas EX-OR e EX-NOR com formas de onda nas entradas. Assim como fizemos, vamos aplicar a operação da tabela-verdade durante cada intervalo de tempo distinto das formas de onda digitais de entrada, conforme ilustrado na Figura 3–46 para uma porta EX-OR. Podemos ver que as formas de onda nas entradas *A* e *B* têm níveis opostos durante os intervalos de tempo t_2 e t_4. Portanto, a saída *X* está em nível ALTO durante esses dois intervalos. Como as duas entradas estão no mesmo nível lógico (ambas em nível ALTO ou ambas em nível BAIXO), durante os intervalos de tempo t_1 e t_3, a saída está em nível BAIXO durante esses intervalos conforme mostra o diagrama de temporização.

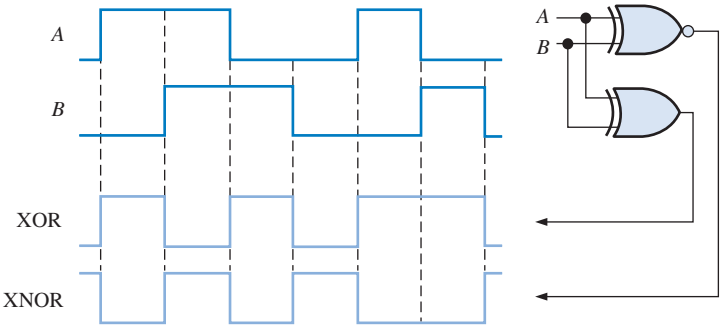


▶ **FIGURA 3–46**

Exemplo de uma porta EX-OR operando com formas de onda digitais nas entradas.

EXEMPLO 3–20

Determine as formas de onda das saídas das portas EX-OR e EX-NOR, a partir das formas de onda nas entradas (*A* e *B*), conforme a Figura 3–47.



▲ **FIGURA 3–47**

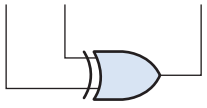
Solução As formas de onda nas saídas são mostradas na Figura 3–47. Observe que a saída da EX-OR está nível ALTO apenas quando as duas entradas estão em níveis opostos. Observe também que a saída da EX-NOR está em nível ALTO apenas quando as duas entradas estão no mesmo nível lógico.

Problema relacionado Determine as formas de onda das saídas (Figura 3–47) se as duas formas de onda nas entradas *A* e *B* forem invertidas.

Uma Aplicação

Uma porta EX-OR pode ser usada como um somador de dois bits. Lembre, do Capítulo 2, que as regras básicas para a adição binária são as seguintes: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 – 1 e 1 + 1 = 10. Um exame na tabela-verdade de uma porta EX-OR nos mostra que a sua saída é o resultado da soma binária dos dois bits das entradas. No caso em que as duas entradas são 1s, a saída é 0, porém não temos o carry de 1. No Capítulo 6 veremos como as portas EX-OR são combinadas para implementar circuitos somadores-completos. A Figura 3–48 ilustra uma porta EX-OR usada como um somador básico.

Bits de entrada		Saída (resultado da soma)	
A	*B*	Σ	
0	0	0	
0	1	1	
1	0	1	
1	1	0	(sem o carry que é 1)



◀ **FIGURA 3–48**
Uma porta EX-OR usada como um somador de dois bits.

SEÇÃO 3–6 REVISÃO

1. Quando a saída de uma porta EX-OR é nível ALTO?
2. Quando a saída de uma porta EX-OR é nível BAIXO?
3. Como uma porta EX-OR pode ser usada para detectar quando dois bits são diferentes?

3-7 LÓGICA PROGRAMÁVEL

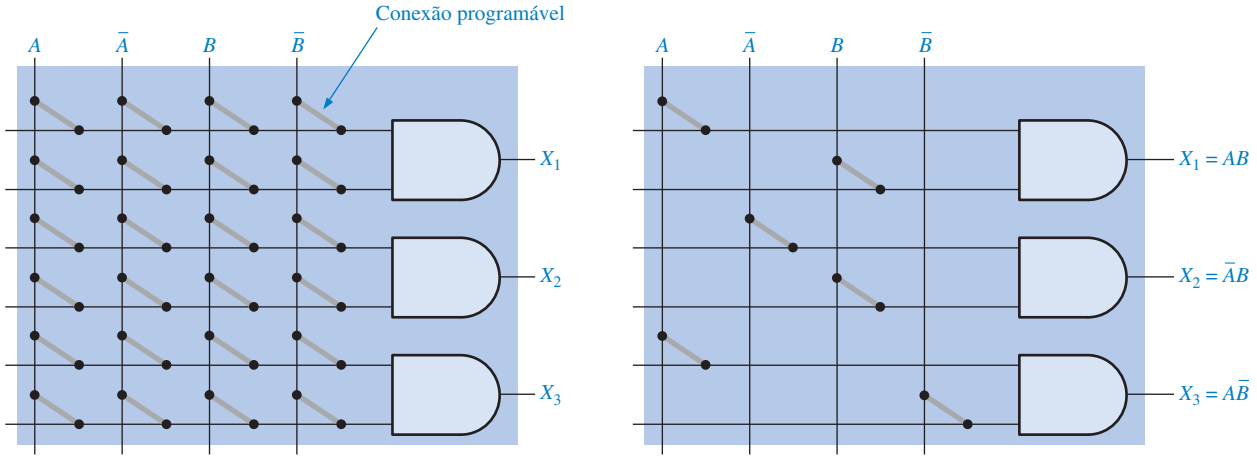
A lógica programável foi introduzida no Capítulo 1. Nesta seção, os conceitos básicos do arranjo AND programável, o qual representa a base para a maior parte da lógica programável, são discutidos e as principais tecnologias de processo são abordadas. Um dispositivo lógico programável (PLD – *programmable logic device*) é um dispositivo que não tem uma função lógica fixa, mas que pode ser programado para implementar qualquer projeto lógico. Conforme já estudamos, os dois tipos de PLDs são SPLD e CPLD. Além do PLD, um dispositivo de uma outra importante categoria da lógica programável é o FPGA. Por questão de simplicidade, todos esses dispositivos são referidos com PLDs. Além disso, alguns conceitos importantes sobre programação são discutidos.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever o conceito de um arranjo AND programável ■ Discutir várias tecnologias de processos ■ Discutir inserção por texto e inserção gráfica com os dois métodos para projeto em lógica programável ■ Descrever os métodos para fazer o download de um projeto para um dispositivo de lógica programável ■ Explicar a programação dentro do próprio sistema

Conceitos Básicos de um Arranjo AND

A maioria dos tipos de PLDs usa alguma forma de **arranjo AND**. Basicamente, esse arranjo consiste de portas AND e uma matriz de interconexões com conexões programáveis em dada ponto de cruzamento, conforme mostra a Figura 3–49(a). A finalidade das conexões programáveis é de fazer ou desfazer uma conexão entre uma linha e uma coluna na matriz de interconexões. Para cada entrada de uma porta AND, apenas uma conexão programável é deixada intacta para conectar a variável desejada à entrada da porta. A Figura 3–49(b) ilustra um arranjo após a programação.



(a) Não programável (b) Programável

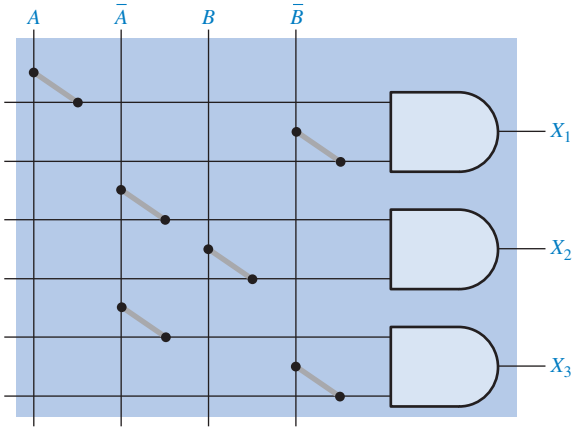
▲ **FIGURA 3–49**

Conceito básico de um arranjo AND.

EXEMPLO 3–21

Mostre como fica o arranjo AND visto na Figura 3–49(a) após ser programado para obter as seguintes saídas: $X_1 = A\overline{B}$, $X_2 = \overline{A}B$, e $X_3 = \overline{A}\,\overline{B}$

Solução Veja a Figura 3–50.



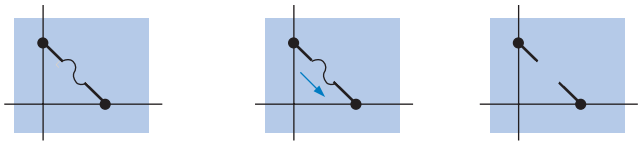
▲ **FIGURA 3–50**

Problema relacionado Quantas linhas, colunas e entradas de porta AND são necessárias para três variáveis de entrada num arranjo de 3 portas AND?

Tecnologia de Processos para Conexões Programáveis

Algumas diferentes tecnologias de processos usadas para implementar conexões programáveis em PLDs.

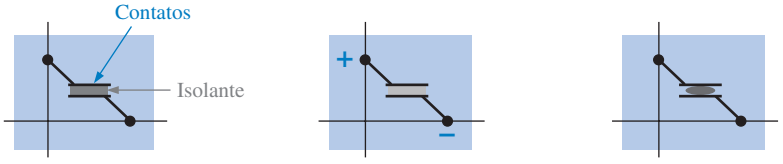
Tecnologia de Fusível Essa foi a tecnologia de conexões programáveis original. Ela ainda é usada em algumas SPLDs. O **fusível** é uma conexão metálica que conecta uma linha com uma coluna numa matriz de interconexões. Antes da programação, existe uma conexão à fusível em cada interconexão. Para programar um dispositivo, os fusíveis selecionados são abertos passando-se uma corrente através deles que seja suficiente para "queimar" os fusíveis e abrir a conexão. Os fusíveis mantidos intactos provêem as conexões entre linhas e colunas. A conexão à fusível é ilustrada na Figura 3–51. Os dispositivos de lógica programável que usam a tecnologia de fusível são programados apenas uma vez (**OTP** – *one-time programmable*).



(a) Fusível intacto antes da programação (b) Corrente de programação (c) Fusível aberto após a programação

◀ **FIGURA 3–51**
A conexão à fusível programável.

Tecnologia Antifusível Uma conexão programável **antifusível** é o oposto de uma conexão à fusível. Em vez de quebrar a conexão, uma conexão é feita durante a programação. Um antifusível começa como um circuito aberto enquanto que um fusível começa como um curto-circuito. Antes da programação, não existem conexões entre as linhas e colunas na matriz de interconexões. Um antifusível consiste basicamente de dois condutores separados por um isolante. Para programar um dispositivo com a tecnologia antifusível, um equipamento de programação aplica uma tensão suficiente no antifusível selecionado para romper com a isolação entre os dois materiais condutores, fazendo com que o isolante se torne uma conexão de baixa resistência. A conexão antifusível é ilustrada na Figura 3–52. Um dispositivo antifusível também é um dispositivo OTP (programável apenas uma vez).

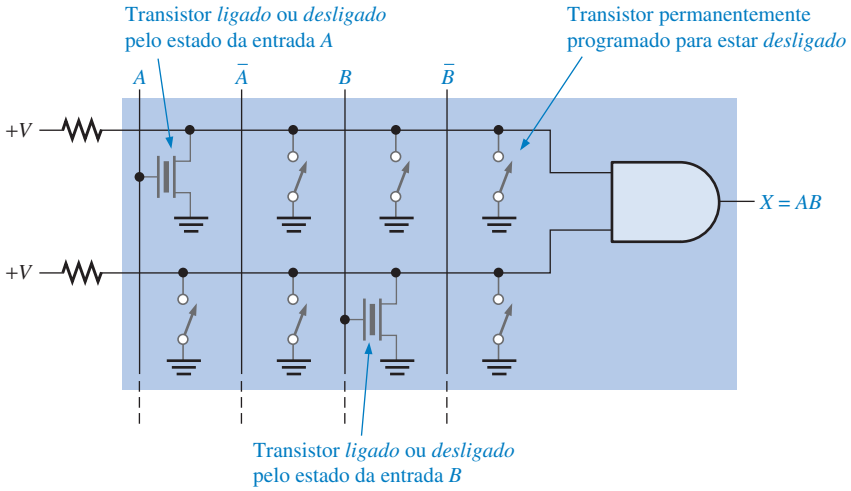


(a) O antifusível está aberto antes da programação. (b) A tensão de programação rompe a camada isolante e estabelece contato. (c) O antifusível entra efetivamente em curto-circuito após a programação.

◀ **FIGURA 3–52**
A conexão antifusível programável.

Tecnologia EPROM Em certos dispositivos lógicos programáveis, as conexões programáveis são semelhantes às células de memória nas **EPROMs** (*electrically programmable read-only memories*). Esse tipo de PLD é programado usando uma ferramenta especial conhecida como programador de dispositivo. O dispositivo é inserido no programador, o qual é conectado a um computador que executa um software de programação. A maioria dos PLDs baseados em EPROM é do tipo OTP. Entretanto, os que apresentam um encapsulamento com "janela" podem ser apagados com luz ultravioleta (UV) e reprogramados usando um equipamento padrão de programação de PLD. A tecnologia do processo EPROM usa um tipo especial de transistor MOS, conhecido como transistor de porta flutuante, como a conexão programável. O dispositivo de porta flutuante utiliza um processo denominado tunelamento Fowler-Nordheim para colocar elétrons na estrutura de porta flutuante.

Em um arranjo AND programável, o transistor de porta flutuante funciona como uma chave para conectar a linha ao nível ALTO ou ao nível BAIXO, dependendo da variável de entrada. Para variáveis de entrada que não são usadas, o transistor é programado para estar permanentemente desligado (aberto). A Figura 3–53 mostra uma porta AND num arranjo simples. A variável *A* controla o estado do transistor na primeira coluna, e a variável *B* controla o transistor na terceira coluna. Quando um transistor está desligado, assim como uma chave aberta, a linha de entrada da porta AND está em +V (ALTO). Quando um transistor está ligado, assim como uma chave fechada, a linha de entrada está conectada em GND (BAIXO). Quando a variável *A* ou *B* for 0 (BAIXO), o transistor está ligado, mantendo a linha de entrada da porta AND em nível BAIXO. Quando *A* ou *B* for 1 (ALTO), o transistor está desligado, mantendo a linha de entrada da porta AND em nível ALTO.



▶ **FIGURA 3–53**
Um arranjo AND simples com tecnologia EPROM. Por questão de simplicidade, apenas uma porta do arranjo é mostrada.

Tecnologia EEPROM A tecnologia de memória apenas de leitura programável e apagável eletricamente é semelhante à EPROM por usar também um tipo de transistor de porta flutuante em células E^2CMOS. A diferença é que **EEPROM** pode ser apagada e reprogramada eletricamente sem a necessidade de luz UV ou equipamentos especiais. Um dispositivos E^2CMOS pode ser programado após ser instalado na placa de circuito impresso, e muitos deles podem ser reprogramados enquanto operam num sistema. Isso é denominado de programação dentro do sistema (**ISP** – *in-system programming*). A Figura 3–53 também pode ser usada como um exemplo para representar um arranjo AND com a tecnologia EEPROM.

Um arranjo flash é um tipo de arranjo EEPROM que pode não apenas ser apagado de forma mais rápida que a tecnologia EEPROM padrão, mas que também pode resultar em dispositivos de maior densidade.

NOTA: COMPUTAÇÃO

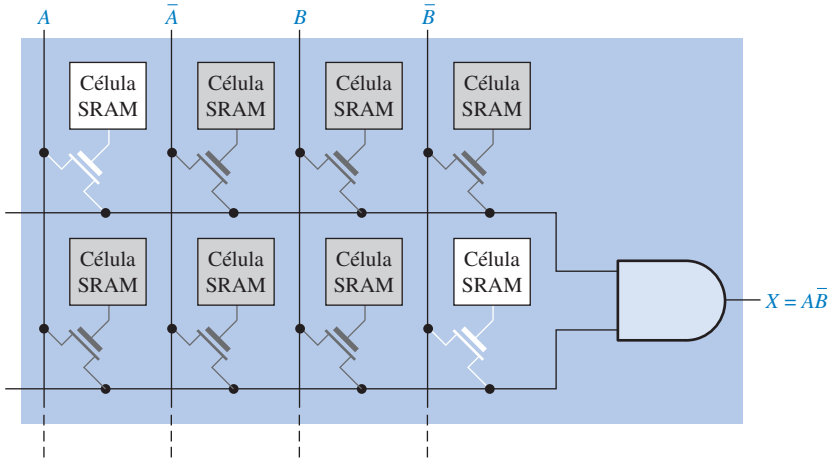
A maioria dos projetos em nível de sistema incorpora uma variedade de dispositivos tais como RAMs, ROMs, controladores e processadores que são interconectados por uma grande quantidade de dispositivos lógicos de propósitos gerais frequentemente referidos como lógica de interface. Os PLDs vieram para substituir muitos dos dispositivos de interface SSI e MSI. O uso de PLDs proporciona uma redução na quantidade de encapsulamentos.

Por exemplo, num sistema de memória de um computador, PLDs podem ser usados para decodificação de endereços e para gerar sinais de escrita bem como outras funções.

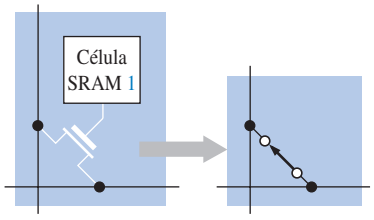
Tecnologia SRAM Muitos FPGAs e alguns CPLDs usam uma tecnologia de processo similar a que é usada em memórias de acesso aleatório estáticas (**SRAMs** – *statics random-access memories*). O conceito básico de arranjos lógicos programáveis baseado em SRAM é ilustrado na Figura 3–54(a). Uma célula de memória do tipo SRAM é usada para *ligar* ou *desligar* um transistor conectando ou desconectando linhas e colunas. Por exemplo, quando a célula de memória contém um 1 (branca), o transistor está *ligado* e quando contém um 0 (cinza) o transistor está *desligado* de forma que não existe conexão entre a linha e a coluna correspondente, como mostra a parte (c).

A tecnologia SRAM é diferente das outras tecnologias de processos discutidas porque ela é uma tecnologia volátil. Isso significa que uma célula SRAM não retém o dado quando a alimentação é *desligada*. Os dados da programação têm que ser carregados na memória; e quando a alimentação é ligada, os dados da memória reprogramam o PLD baseado em SRAM.

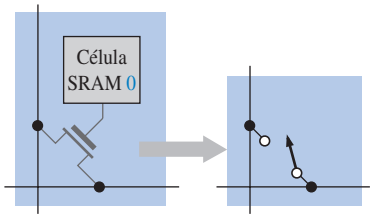
As tecnologias de processos do tipo fusível, antifusível, EPROM e EEPROM são não-voláteis, de forma que elas retêm as suas programações quando a alimentação é *desligada*. Um fusível é permanentemente aberto, um antifusível é permanentemente fechado e transistores de porta flutuante usados em arranjos baseados em EPROM e EEPROM podem reter a condição de estado *ligado* ou *desligado* indefinidamente.



(a) Arranjo programável baseado em SRAM



(b) Transistor *ligado*

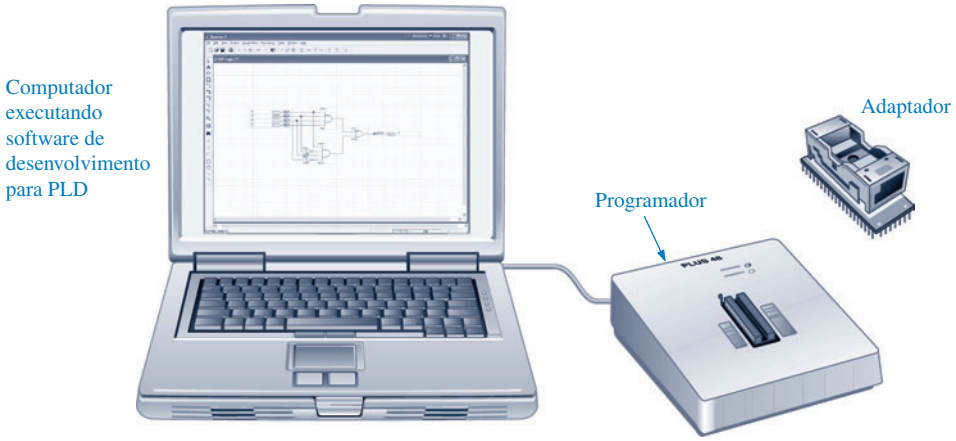


(c) Transistor *desligado*

◀ **FIGURA 3–54**
Conceito básico de um arranjo AND com tecnologia SRAM.

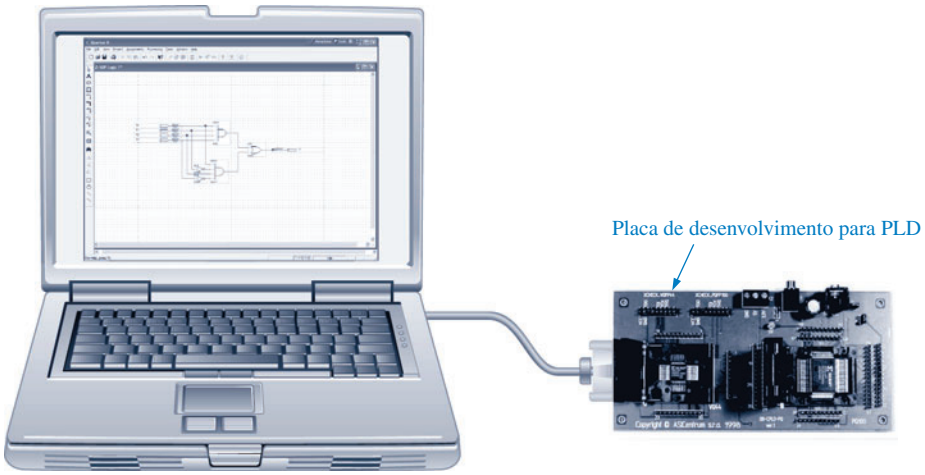
Programação de um Dispositivo

O conceito geral de programação foi introduzido no Capítulo 1 e já vimos como interconexões podem ser feitas num arranjo simples abrindo ou fechando as conexões programáveis. SPLDs, CPLDs e FPGAs são programados essencialmente da mesma forma. Os dispositivos com tecnologias de processos OTP (fusível, antifusível ou EPROM) têm que ser programados com um equipamento de hardware especial denominado *programador*. O programador é conectado a um computador por meio de um cabo de interface padrão, conforme mostra a Figura 3–55. O software de desenvolvimento é instalado no computador e o dispositivo é inserido no soquete do programador. A maioria dos programadores tem adaptadores, como o que é mostrado na figura, o qual permite o uso de diferentes tipos de encapsulamentos.



◀ **FIGURA 3–55**
Sistema para programação de um PLD num equipamento de programação (programador).

Dispositivos lógicos baseados em EEPROM e SRAM são reprogramados e podem ser reconfigurados muitas vezes. Embora um dispositivo programador possa ser usado para esse tipo de dispositivo, ele é geralmente programado numa placa de desenvolvimento de PLD, como mostra a Figura 3–56. Um projeto lógico pode ser desenvolvido usando essa abordagem porque qualquer alteração necessária durante o processo de projeto pode ser facilmente realizada simplesmente reprogramando o PLD. Um PLD para o qual o projeto lógico em software pode ser transferido (operação de download) é denominado de **dispositivo destino**. Além do dispositivo destino, as placas de desenvolvimento propiciam tipicamente outros circuitos e conectores para interface com um computador e outros circuitos periféricos. Além disso, são incluídos nas placas de desenvolvimento pontos de teste e dispositivos display para observar a operação do dispositivo programado.



► **FIGURA 3–56**
Sistema de programação para dispositivos lógicos reprogramáveis.

Inserção do Projeto Conforme estudamos no Capítulo 1, a inserção do projeto é a etapa onde o projeto lógico é programado no software de desenvolvimento. As duas principais formas de inserção de um projeto são a entrada via texto e a entrada gráfica (esquemático), sendo que os fabricantes de lógica de programação fornecem pacotes de softwares que dão suporte aos seus dispositivos que trabalham com ambas as formas.

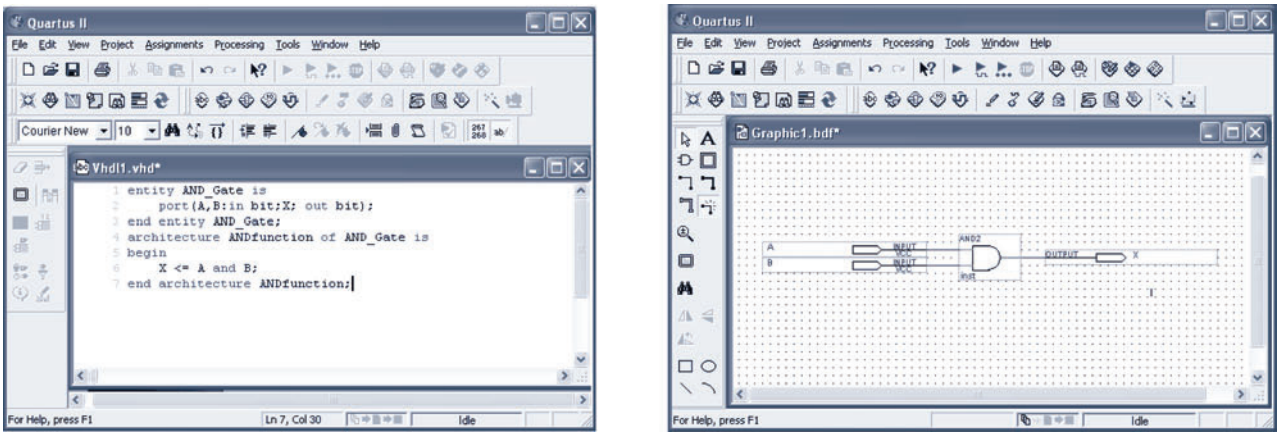
A **inserção via texto** na maioria dos softwares de desenvolvimento, independente do fabricante, suporta duas ou mais linguagens de desenvolvimento de hardware (**HDLs**). Por exemplo, todos os pacotes de software suportam os dois padrões HDL da IEEE, VHDL e verilog. Alguns pacotes de softwares também suportam certas linguagens proprietárias tais como ABEL, CUPL e AHDL.

No modo de **inserção gráfico (esquemático)**, os símbolos lógicos tais como das portas AND e OR são colocados na tela e interconectados para formar o circuito desejado. Nesse método usamos os símbolos que nos são familiar; porém, na realidade, o software converte cada símbolo e as conexões num arquivo texto para o computador usar; é um processo transparente para nós. Um exemplo simples mostrando uma tela com a inserção via texto e uma tela com a inserção gráfica para uma porta AND é mostrado na Figura 3–57. Como regra geral, a inserção gráfica é usada para circuitos lógicos menos complexos e a inserção via texto, embora também possa ser usada para lógicas bastante simples, é usado para implementações maiores e mais complexas.

Programação Dentro do Sistema

Certos CPLDs e FPGAs podem ser programados após serem instalados na placa de circuito impresso (PCB – *printed circuit board*). Após o desenvolvimento do projeto lógico e o teste completo numa placa de desenvolvimento, o dispositivo pode então ser programado mesmo já estando soldado na placa do sistema no qual vai operar. Além disso, se for necessário realizar uma alteração no projeto, o dispositivo na placa do sistema pode ser reconfigurado para incorporar as modificações do projeto.

Num sistema de produção, a programação de um dispositivo na própria placa do sistema minimiza o manuseio e elimina a necessidade de manter estoques de dispositivos programados. Isso



(a) Inserção de texto VHDL.

(b) Inserção gráfica (esquemático) equivalente.

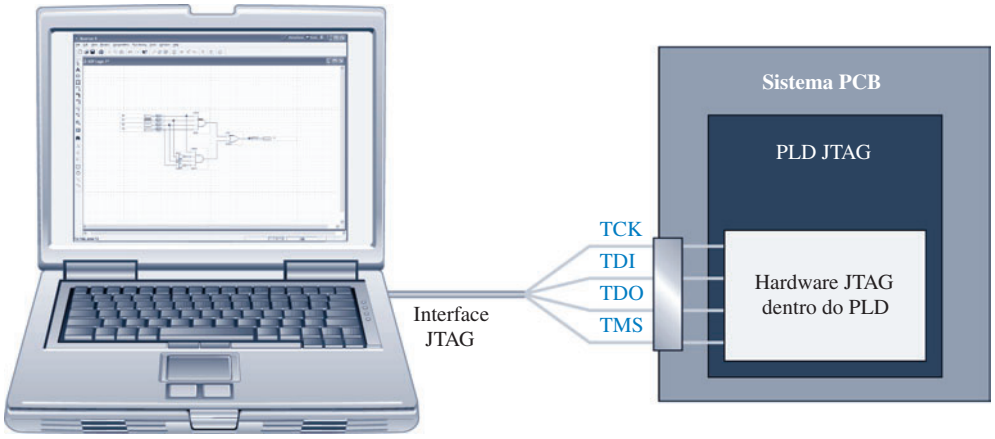
▲ **FIGURA 3–57**

Exemplo da inserção do projeto de uma porta AND.

também evita a possibilidade de partes com defeito serem colocadas no produto. Dispositivos não programados podem ser mantidos no almoxarifado e programados *on-board* (na placa do sistema) conforme a necessidade. Isso minimiza custos necessários para inventários e melhora a qualidade dos seus produtos.

JTAG O padrão JTAG estabelecido pela Joint Test Action Group é o nome normalmente usado para o padrão 1149.1 da IEEE. O padrão **JTAG** foi desenvolvido para prover um método simples, denominado varredura de limites, para teste da funcionalidade de dispositivos programáveis bem como a identificação de conexões ruins em placas de circuitos, tais como curto-circuito entre pinos, pinos abertos, problemas em trilhas, entre outros. Mais recentemente, o padrão JTAG tem sido usado como uma forma conveniente de configurar dispositivos programáveis on-board. De acordo com o aumento da demanda por produtos programáveis por campo, o uso do padrão JTAG como uma forma conveniente de reprogramar CPLDs e FPGAs continua aumentando.

Dispositivos JTAG tem um hardware interno dedicado que interpreta instruções e dados fornecidos por quatro sinais dedicados. Esses sinais são definidos pelo padrão JTAG como Teste de Entrada de Dados (TDI – *Test Data In*), Teste de Saída de Dados (TDO – *Test Data Out*), Teste de Seleção de Modo (TMS – *Test Mode Select*) e Teste de Clock (TCK – *Test Clock*). O hardware dedicado JTAG interpreta instruções e dados nos sinais TDI e TMS e controla os dados de saída no sinal TDO. O sinal TCK é usado como clock do processo. Uma placa de circuito impresso JTAG é representada na Figura 3–58.



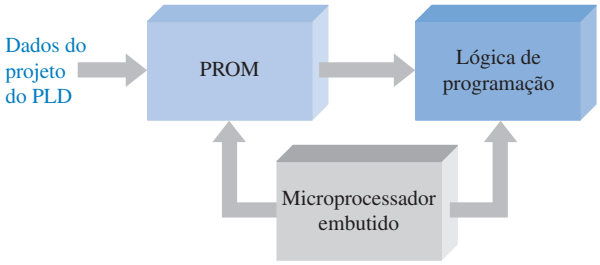
▲ **FIGURA 3–58**

Ilustração simplificada da programação dentro do sistema via interface JTAG.

Processador Embutido Uma outra abordagem para programação dentro do sistema é o uso de microprocessador e memória embutidos. O processador é embutido no sistema juntamente com o CPLD ou FPGA e outros circuitos, sendo o processador dedicado à finalidade de configuração do dispositivo programável dentro do sistema.

Conforme estudamos, os dispositivos baseados em SRAM são voláteis perdendo os dados programados quando a alimentação é desligada. É necessário armazenar os dados da programação numa memória apenas de leitura programável (PROM – *programmable read-only memory*), que é não-volátil. Quando a alimentação é ligada, o processador embutido toma o controle da transferência de dados da PROM para o CPLD ou FPGA.

Além disso, um processador embutido é usado algumas vezes para reconfiguração de um dispositivo programável enquanto o sistema está em operação. Nesse caso, as alterações de projeto são feitas com software, sendo que os novos dados são carregados numa PROM sem perturbar a operação do sistema. O processador controla a transferência de dados diretamente para o dispositivo no momento apropriado. Um diagrama em bloco simples de um processador/sistema lógico programável é mostrado na Figura 3–59.

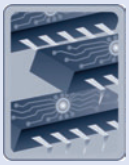


► **FIGURA 3–59**
Diagrama em bloco simplificado de um PLD com processador embutido e memória.

SEÇÃO 3–7 REVISÃO

1. Faça uma lista com as cinco tecnologias de processo usadas para conexões programáveis em lógica programável.
2. O que significa o termo volátil em relação a PLDs e qual tecnologia de processo é volátil?
3. Quais são os dois métodos de inserção de projeto para a programação de PLDs e FPGAs?
4. Defina JTAG.

3-8 LÓGICA DE FUNÇÕES FIXAS



As duas principais tecnologias de circuito integrado (CI) usadas para implementar portas lógicas são CMOS e TTL. As operações lógicas NOT, AND, OR, NAND, NOR e EX-OR são as mesmas independente da tecnologia de CI usada; ou seja, uma porta AND tem a mesma função lógica se for implementada com CMOS ou TTL.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar as séries mais comuns de CMOS e TTL ■ Comparar CMOS com TTL em termos de tipos de dispositivos e performance de parâmetros ■ Definir *tempo de atraso de propagação* ■ Definir *dissipação de potência* ■ Definir *fan-out* ■ Definir *produto velocidade-potência* ■ Interpretar informações básicas de folhas de dados

A tecnologia de Semicondutor de Óxido Metálico Complementar (**CMOS** – *Complementary Metal-Oxide Semiconductor*) é implementada com um tipo de transistor de efeito de campo. A tecnologia da Lógica Transistor-Transistor (**TTL** – *Transistor-Transistor Logic*) é implementada com transistores de junção bipolar. Tenha em mente que diferem apenas nos tipos de componentes de circuito e valores de parâmetros e não na operação lógica básica. Uma porta AND CMOS tem a mesma operação lógica que uma porta AND TTL. Isso é válido para todas as outras funções lógicas básicas. A diferença entre TTL e CMOS está nas características de performance tais como velocidade de comutação (atraso de propagação), dissipação de potência, imunidade a ruído entre outros parâmetros.

CMOS

Existe uma pequena divergência em relação a qual tecnologia de circuito, CMOS ou TTL, é a mais amplamente usada. Parece que a tecnologia CMOS se tornou dominante e pode eventualmente substituir CIs TTL de pequena e média escala de integração. Embora a tecnologia TTL tenha dominado por muitos anos principalmente por ter uma velocidade de comutação maior e por ter uma maior diversidade de tipos de dispositivos, a tecnologia CMOS sempre teve a vantagem de ter uma menor dissipação de potência embora esse parâmetro seja dependente da freqüência de operação. A velocidade de comutação de dispositivos CMOS tem melhorado bastante sendo agora competitiva com dispositivos TTL, enquanto que a baixa dissipação de potência e outros fatores desejáveis se mantiveram com o progresso da tecnologia.

Séries CMOS As categorias de dispositivos CMOS em termos de tensão de alimentação são 5 V, 3,3 V, 2,5 V e 1,8 V. As famílias CMOS de baixa tensão são os desenvolvimentos mais recentes e são o resultado de um esforço para reduzir a dissipação de potência. Como a dissipação de potência é proporcional ao quadrado da tensão, a redução de 5 V para 3,3 V, por exemplo, reduz a potência em 34% sendo que os outros fatores permanecem os mesmos.

Dentro de cada categoria de tensão de alimentação, várias séries de portas CMOS estão disponíveis. Essas séries dentro da família CMOS diferem nas suas características de performance e são identificadas pelos prefixos 74 ou 54 seguido por uma letra ou letras que indicam a série e um número que indica o tipo de dispositivo lógico. O prefixo 74 indica a classe comercial para uso geral e o prefixo 54 indica a classe militar para condições ambientais mais severas. Nesse livro nos referimos sempre aos dispositivos de prefixo 74. As séries básicas CMOS para a categoria de 5 V e suas designações incluem:

- 74HC e 74HCT – CMOS de alta velocidade (o “T” indica compatibilidade com TTL)
- 74AC e 74ACT – CMOS avançado
- 74AHC e 74AHCT – CMOS de alta velocidade avançado

As séries CMOS básicas para a categoria de 3,3 V e suas designações incluem:

- 74LV – CMOS de baixa tensão
- 74LVC – CMOS de baixa tensão
- 74ALVC – CMOS de baixa tensão avançado

Além das séries 74 existe a série 4000, que é uma tecnologia CMOS mais antiga de baixa velocidade ainda disponível, embora de uso limitado. Além do CMOS “puro”, existem séries que combinam CMOS e TTL denominadas BiCMOS. As séries BiCMOS básicas são:

- 74BCT – BiCMOS
- 74ABT – BiCMOS avançado
- 74LVT – BiCMOS de baixa tensão
- 74ALB – BiCMOS de baixa tensão avançado

TTL

A tecnologia TTL foi por muitos anos a mais popular tecnologia de CIs digitais. Uma vantagem da tecnologia TTL é que ela não é sensível à descarga eletrostática como CMOS, portanto, é mais prática para uso na maioria dos laboratórios de experimentação e prototipagem porque não precisamos nos preocupar com precauções para manuseio dos dispositivos.

Séries TTL Assim como na família CMOS, algumas séries de portas lógicas TTL estão disponíveis, sendo que todas operam com tensão de alimentação de 5 V cc. Essas séries dentro da família TTL diferem em suas características de desempenho e são identificadas pelos prefixos 74 ou 54 seguido por uma ou mais letras de indicam a série e um número que indica o tipo de dispositivo lógico dentro da série. Um CI TTL pode ser distinguido de um CMOS pelas letras que seguem o prefixo 74 ou 54.

As séries básicas TTL e as suas designações são:

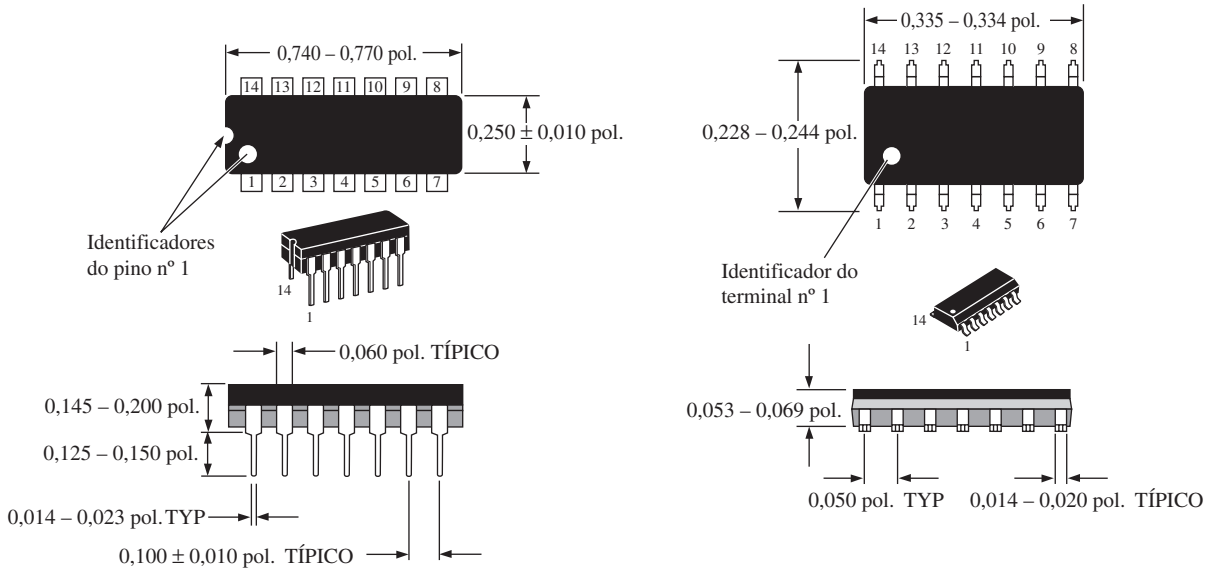
- 74 – TTL padrão (sem letra)
- 74S – TTL schottky
- 74AS – TTL schottky avançado
- 74LS – TTL schottky de baixa potência
- 74ALS – TTL schottky de baixa potência avançado
- 74F – TTL rápido

Tipos de Portas Lógicas de Funções Fixas

Todas as operações lógicas básicas, NOT, AND, OR, NAND, NOR, EX-OR e EX-NOR estão disponíveis nas tecnologias CMOS e TTL. Além disso, portas com buffers nas saídas também são disponibilizadas para acionar cargas que necessitam de correntes maiores. Os tipos de configurações de portas tipicamente disponíveis na forma de CIs são identificados pelos dois últimos dígitos na designação da série. Por exemplo, 74LS04 é um CI contendo seis inversores **Schottky** de baixa potência. Algumas das configurações de portas lógicas e os seus dígitos padrão de identificação são:

- Quatro portas NAND de 2 entradas – **00**
- Quatro portas NOR de 2 entradas – **02**
- Seis inversores – **04**
- Quatro portas AND de 2 entradas – **08**
- Três portas NAND de 3 entradas – **10**
- Três portas AND de 3 entradas – **11**
- Duas portas NAND de 4 entradas – **20**
- Duas portas AND de 2 entradas – **21**
- Três portas NOR de 3 entradas – **27**
- Uma porta NAND de 8 entradas – **30**
- Quatro portas OR de duas entradas – **32**
- Quatro EX-OR – **86**
- Quatro EX-NOR – **266**

Encapsulamentos de CIs Todas as séries 74 CMOS são compatíveis pino-a-pino com os mesmos tipos de dispositivos TTL. Isso significa que um CI digital CMOS tal como um 74HC00 (quatro portas NAND de 2 entradas) tem uma pinagem (numeração dos pinos) idêntica ao dispositivo correspondente TTL. A Figura 3–60 mostra encapsulamentos de CIs de portas lógicas do tipo DIP, para montagem em placas com furos, e SOIC, para montagem em superfície. Em alguns casos são disponibilizados outros tipos de encapsulamentos. O encapasulamento SOIC é significativamente menor que o DIP. Os diagramas de configuração de pino para a maioria dos dispositivos de funções lógicas fixas apresentados acima são mostrados na Figura 3–61.

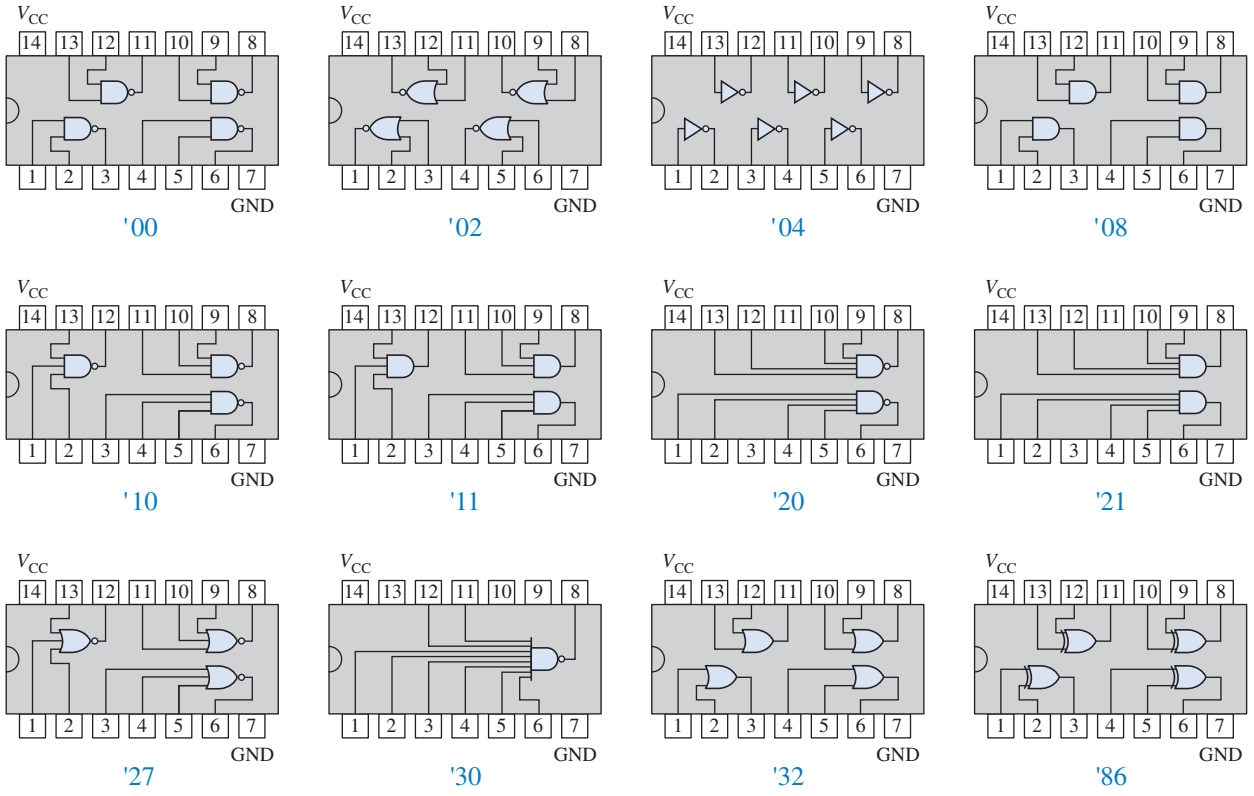


(a) Encapsulamento DIP do tipo PTH de 14 pinos.

(b) Encapsulamento SOIC para montagem em superfície.

▲ **FIGURA 3–60**

Encapsulamentos DIP e SOIC típicos mostrando os números dos pinos e as dimensões básicas.



▲ **FIGURA 3–61**

Diagramas de configuração de pinos para algumas configurações de CIs de funções lógicas fixas comuns.

Lógica de Porta Única Uma seleção limitada de portas lógicas CMOS é disponibilizada num encapsulamento de porta única. Com uma porta por encapsulamento, essa série tem um pequeno encapsulamento de 5 pinos com o objetivo de ser usada em modificações de última hora em circuitos lógicos usados em lugares pequenos onde o espaço disponível é limitado.

Símbolos Lógicos Os símbolos lógicos para circuitos integrados de funções fixas usam os símbolos de porta padrão e mostram o número de portas no encapsulamento do CI e os números dos pinos associados à cada porta bem como os números dos pinos de V_{CC} e GND. Um exemplo é mostrado na Figura 3–62 para um CI com seis inversores (*hex inverter*) e um outro com quatro portas NAND de 2 entradas (*quad 2-input NAND*). Tanto o símbolo característico quanto o símbolo retangular são mostrados. Independente da família lógica, todos os dispositivos com o mesmo sufixo são compatíveis pino-a-pino; em outras palavras, eles têm a mesma configuração interna. Por exemplo, o 7400, 74S00, 74LS00, 74F00, 74HC00 e 74AHC00 são encapsulamentos que possuem quatro portas NAND de 2 entradas e são compatíveis pino-a-pino.

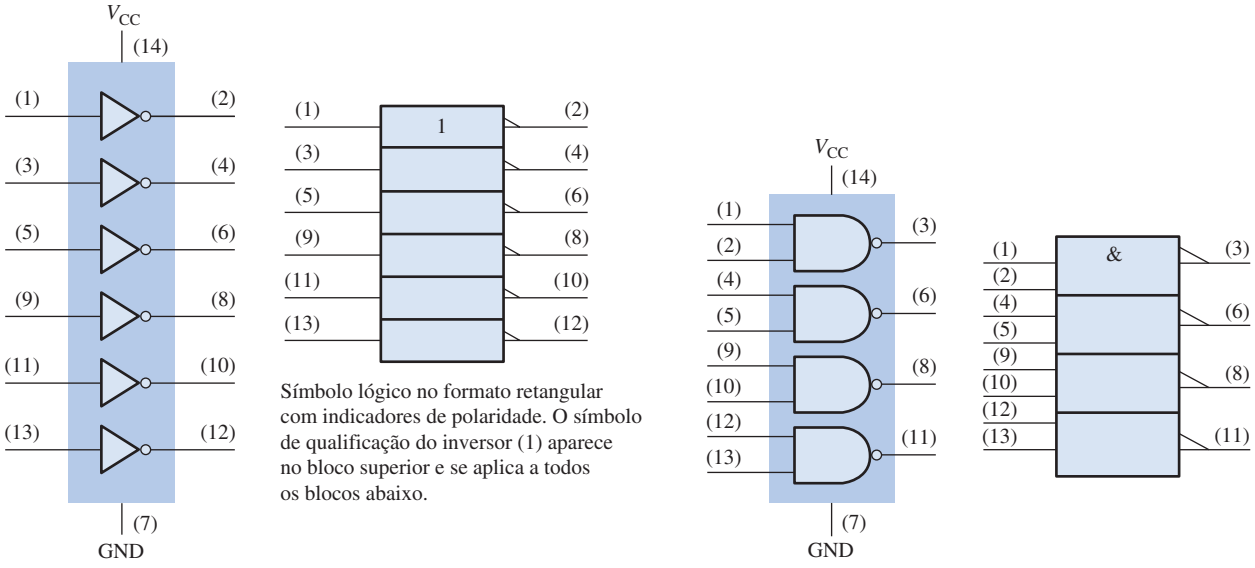


Diagrama lógico com símbolo característico

(a) Seis inversores

(b) Quatro portas NAND de 2 entradas

▲ **FIGURA 3–62**

Símbolos lógicos para um CI contendo seis inversores (sufixo 04) e um outro com quatro portas NAND de 2 entradas (sufixo 00). O símbolo se aplica aos mesmos dispositivos nas séries CMOS e TTL.

Características de Performance e Parâmetros

Circuitos lógicos de alta velocidade têm um tempo de atraso de propagação curto.

Algumas características definem a performance de um circuito lógico. Essas características são a velocidade de comutação medida em termos do tempo de atraso de propagação, a dissipação de potência, o fan-out ou capacidade de acionamento, o produto velocidade-potência, a tensão de alimentação cc e os níveis lógicos de entrada e saída.

Tempo de Atraso de Propagação Esse parâmetro é o resultado de uma limitação na velocidade ou freqüência na qual o circuito lógico pode operar. Os termos *baixa velocidade* e *alta velocidade*, aplicados aos circuitos lógicos se referem ao tempo de atraso de propagação. Quanto menor o atraso de propagação, maior a velocidade do circuito e maior a freqüência na qual ele pode operar.

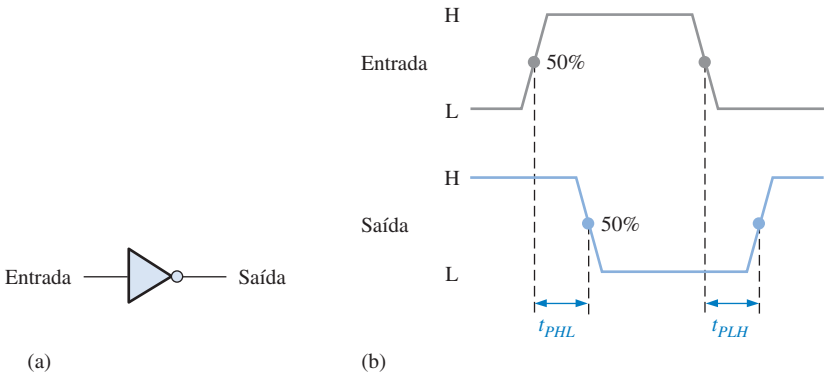
O **tempo de atraso de propagação**, t_p, de uma porta lógica é o intervalo de tempo entre a aplicação de um pulso na entrada e a ocorrência de um pulso resultante na saída. Existem duas for-

mas diferentes de medir o tempo de atraso de propagação associado a uma porta lógica que se aplica a todos os tipos de portas básicas:

- t_{PHL}: O tempo entre um ponto de referência especificado no pulso de entrada e o correspondente ponto de referência no pulso de saída resultante, com a saída mudando de nível ALTO para nível BAIXO (HL).
- t_{PLH}: O tempo entre um ponto de referência especificado no pulso de entrada e o correspondente ponto de referência no pulso de saída resultante, com a saída mudando de nível BAIXO para nível ALTO (LH).

EXEMPLO 3–22

Mostre o tempo de atraso de propagação para o inversor da Figura 3–63(a).



▲ FIGURA 3–63

Solução Os tempos de atraso de propagação, t_{PHL} e t_{PLH}, são indicados na parte (b) da figura. Nesse caso, os atrasos são medidos entre os pontos de 50% das bordas correspondentes dos pulsos de entrada e saída. Os valores de t_{PHL} e t_{PLH} não são necessariamente iguais mas em muitos casos eles são iguais.

Problema relacionado Um tipo de porta lógica tem uma especificação máxima para t_{PHL} e t_{PLH} de 10 ns. Para um outro tipo de porta o valor é de 4 ns. Qual porta pode operar numa freqüência maior?

Para portas da série TTL padrão, o atraso típico de propagação é de 11 ns e para portas da série F é de 3,3 ns. Para a série CMOS HCT, o atraso de propagação é de 7 ns, para a série AC é de 5 ns e para a série ALVC é de 3 ns. Todos os valores especificados dependem de certas condições de operação conforme especificadas nas folhas de dados.

Fonte de Tensão CC (V_{CC}) As tensões de alimentação cc típicas para CMOS são 5 V, 3,3 V, 2,5 V ou 1,8 V, dependendo da categoria. Uma vantagem da CMOS é que a tensão de alimentação pode variar ao longo de uma faixa maior que para TTL. O CMOS de 5 V pode tolerar uma variação na tensão de alimentação de 2 V a 6 V e ainda operar adequadamente embora o tempo de atraso de propagação e a dissipação de potência sejam afetados significativamente. O CMOS de 3,3 V pode operar com tensões de alimentação de 2 V a 3,6 V. A tensão de alimentação cc típica para TTL é 5,0 V com um mínimo de 4,5 V e um máximo de 5,5 V.

Uma baixa dissipação de potência significa menos corrente drenada da fonte de alimentação cc.

Dissipação de Potência A **dissipação de potência**, P_D, de uma porta lógica é o produto da tensão de alimentação cc e a corrente de alimentação média. Normalmente, a corrente de alimentação quando a saída da porta é nível BAIXO é maior que quando a saída da porta é nível ALTO. As folhas de dados do fabricante geralmente especificam a corrente de alimentação para a saída em nível BAIXO como I_{CCL} e para o nível ALTO como I_{CCH}. A corrente de alimentação média é determinada baseada num ciclo de trabalho de 50% (a saída em nível BAIXO metade do tempo e em nível ALTO a outra metade do tempo), de forma que a dissipação de potência média de uma porta lógica é

Equação 3–2

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

As portas lógicas de séries CMOS têm baixas dissipações de potência comparadas com as séries TTL. Entretanto, a dissipação de potência CMOS depende da freqüência de operação. Na freqüência zero a potência quiescente é tipicamente na faixa de microwatts/porta e na freqüência máxima de operação ela está na faixa de miliwatts; portanto, a potência às vezes é especificada para uma dada freqüência. A série HC, por exemplo, tem uma potência de 2,75 μW/porta a 0 Hz (quiescente) e 600 μW/porta a 1 MHz.

A dissipação de potência para TTL é independente da freqüência. Por exemplo, a série ALS gasta 1,4 mW/porta independente da freqüência e a série F gasta 6 mW/porta.

Níveis Lógicos de Entrada e Saída V_{IL} é a tensão de entrada de nível BAIXO para uma porta lógica e V_{IH} é a tensão de entrada de nível ALTO. O CMOS de 5 V aceita uma tensão máxima de 1,5 V como V_{IL} e uma tensão mínima de 3,5 V como V_{IH}. TTL aceita uma tensão máxima de 0,8 V como V_{IL} e uma tensão mínima de 2 V como V_{IH}.

VOL é a tensão de saída de nível BAIXO e VOH é a tensão de saída de nível ALTO. Para o CMOS de 5 V, o V_{OL} máximo é 0,33 V e o V_{OH} mínimo é 4,4 V. Para TTL, o V_{OL} máximo é 0,4 V e o V_{OH} mínimo é 2,4 V. Todos os valores dependem das condições de operação conforme especificado nas folhas de dados.

Produto Velocidade-Potência Esse parâmetro (**produto velocidade-potência**) pode ser usado como uma medida de desempenho de um circuito lógico levando em conta o tempo de atraso de propagação e a dissipação de potência. Ele é especialmente útil na comparação entre diversas séries de portas lógicas dentro das famílias CMOS ou TTL ou para comparação entre portas CMOS e TTL.

O produto velocidade-potência de um circuito lógico é o produto do tempo de atraso de propagação pela dissipação de potência e é expresso em joules (J), que é a unidade de energia. A fórmula é a seguinte:

Equação 3–3

$$SPP = t_p P_D$$

EXEMPLO 3–23

Uma certa porta tem um atraso de propagação de 5 ns e $I_{CCH} = 1$ mA e $I_{CCL} = 2{,}5$ mA com uma tensão de alimentação cc de 5 V. Determine o produto velocidade-potência.

Solução

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right) = 5\text{ V}\left(\frac{1\text{ mA} + 2{,}5\text{ mA}}{2}\right) = 5\text{ V}(1{,}75\text{ mA}) = 8{,}75\text{ mW}$$

$$SPP = (5\text{ ns})\,(8{,}75\text{ mW}) = \mathbf{43{,}75\text{ pJ}}$$

Problema relacionado Se o atraso de propagação de uma porta é 15 ns e o produto velocidade-potência for 150 pJ, qual a dissipação média de potência?

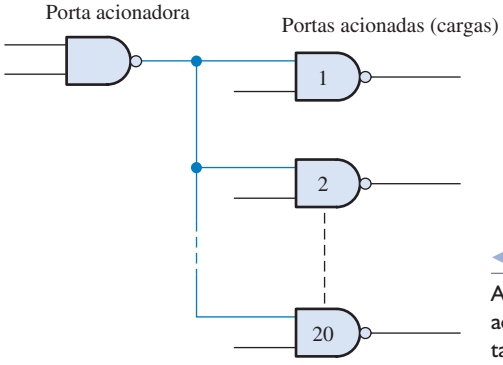
Fan-out e Acionamento de Cargas O **fan-out** de uma porta lógica é o número máximo de entradas que pode ser conectado na saída da porta mantendo ainda os níveis de tensão dentro dos limites especificados. O fan-out é um parâmetro significativo apenas para TTL por causa do tipo de tecnologia de circuito. Devido às altas impedâncias associadas com os circuitos CMOS, o fan-out é muito alto porém depende da freqüência devido os efeitos capacitivos.

O fan-out é especificado em termos de **unidades de cargas**. Uma unidade de carga para uma porta lógica é igual a entrada de um circuito. Por exemplo, uma unidade de carga para uma porta NAND 74LS00 é igual a uma entrada para uma outra porta lógica da série 74LS (não necessariamente uma porta NAND). Devido à corrente de entrada em nível BAIXO (I_{IL}) de um 74LS00 ser 0,4 mA e a capacidade de corrente de saída em nível BAIXO (I_{OL}) se 8,0 mA, o número de unidades de cargas que uma porta 74LS00 pode acionar em nível BAIXO é

Um fan-out mais alto significa que uma saída de porta pode ser conectada a mais entradas de porta.

$$\text{Unidades de cargas} = \frac{I_{OL}}{I_{IL}} = \frac{8{,}0\text{ mA}}{0{,}4\text{ mA}} = 20$$

A Figura 3–64 mostra uma porta lógica LS acionando um número de outras portas da mesma tecnologia de circuito, onde o número de portas depende da tecnologia de circuito em particular. Por exemplo, como já estudamos, o número máximo de entradas de portas (unidades de carga) que uma porta da série TTL 74LS pode acionar é 20.



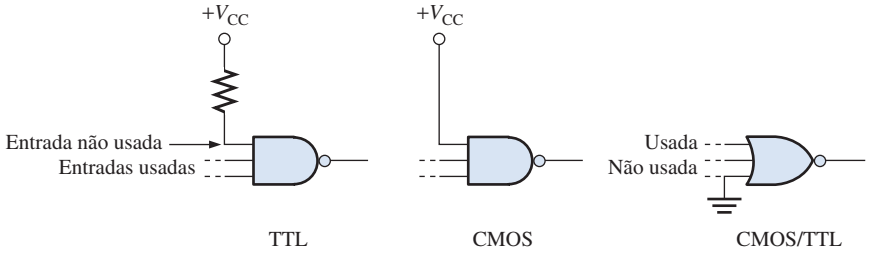
◀ **FIGURA 3–64**
A saída de uma porta NAND TTL LS aciona no máximo 20 entradas de portas TTL LS.

Folhas de Dados

Uma folha de dados consiste de uma página de informações que mostra, dentre outras coisas, o diagrama lógico e os encapsulamentos, as condições recomendadas de operação, as características elétricas e as características de comutação. As folhas de dados parciais para um 74LS00 e um 74HC00A são mostradas nas Figuras 3–65 e 3–66, respectivamente. A extensão das folhas de dados varia sendo que algumas delas têm muito mais informações que outras. Folhas de dados adicionais são fornecidas no CD-ROM que acompanha esse livro.

DICA PRÁTICA

As entradas não usadas de portas TTL e CMOS devem ser conectadas a um nível lógico apropriado (ALTO ou BAIXO). No caso de AND/NAND, é recomendado que entradas não usadas sejam conectadas à V_{CC} (através de um resistor de 1 kΩ em TTL) e para OR/NOR, as entradas não usadas devem ser conectadas a GND.

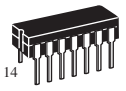


QUATRO PORTAS NAND DE 2 ENTRADAS

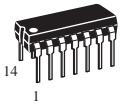
• ESD > 3500 Volts

SN54/74LS00

QUATRO PORTAS NAND DE 2 ENTRADAS
SCHOTTKY DE BAIXA POTÊNCIA



SUFIXO J
CERÂMICO
ENCAPSULAMENTO
632-08

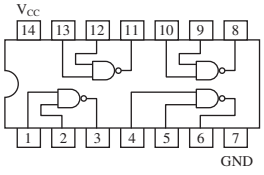


SUFIXO N
PLÁSTICO
ENCAPSULAMENTO
646-06

SUFIXO D
SOIC
ENCAPSULAMENTO
751A-02

SEQÜÊNCIA DA INFORMAÇÃO

SN54LSXXJ	Cerâmico
SN74LSXXN	Plástico
SN74LSXXD	SOIC



SN54/74LS00

CARACTERÍSTICAS CC EM FUNÇÃO DA FAIXA DE TEMPERATURA DE OPERAÇÃO (exceto se especificado o contrário)

Símbolo	Parâmetro		Limites Mín.	Típico	Mx.	Unidade	Condições de teste
V_{IH}	Tensão de entrada em nível ALTO		2,0			V	Garantida a tensão de entrada em nível ALTO para todas as entradas
V_{IL}	Tensão de entrada em nível BAIXO	54			0,7	V	Garantida a tensão de entrada em nível BAIXO para todas as entradas
		74			0,8		
V_{IK}	Tensão de Grampeamento de Entrada			–0,65	–1,5	V	V_{CC} = MIN, I_{IN} = –18 mA
V_{OH}	Tensão de Saída em Nível ALTO	54	2,5	3,5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} ou V_{IL} por Tabela-Verdade
		74	2,7	3,5		V	
V_{OL}	Tensão de Saída em Nível BAIXO	54, 74		0,25	0,4	V	I_{OL} = 4,0 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} ou V_{IH} por Tabela-Verdade
		74		0,35	0,5	V	I_{OL} = 8,0 mA
I_{IH}	Corrente de Entrada em Nível ALTO				20	μA	V_{CC} = MAX, V_{IN} = 2,7 V
					0,1	mA	V_{CC} = MAX, V_{IN} = 7,0 V
I_{IL}	Corrente de Entrada em Nível BAIXO				–0,4	mA	V_{CC} = MAX, I_{IN} = 0,4 V
I_{OS}	Corrente de Curto-Circuito (Nota 1)		–20		–100	mA	V_{CC} = MAX
I_{CC}	Corrente de Alimentação Total, Saída em Nível ALTO				1,6	mA	V_{CC} = MAX
	Total, Saída em Nível BAIXO				4,4		

NOTA 1: Não mais que uma saída deve estar em curto-circuito de cada vez, nem por mais de 1 segundo.

CARACTERÍSTICAS CA (T_A = 25°X)

Símbolo	Parâmetro	Limites Mín.	Típico	Máx.	Unidade	Condições de teste
t_{PLH}	Atraso de Desligamento da Entrada para Saída		9,0	15	ns	V_{CC} = 5,0 V, C_L = 15 pF
t_{PHL}	Atraso de Ligamento da Entrada para Saída		10	15	ns	

FAIXAS DE OPERAÇÃO SEGURA

Símbolo	Parâmetro		Mín.	Típico	Máx.	Unidade
V_{CC}	Tensão de Alimentação	54	4,5	5,0	5,5	V
		74	4,75	5,0	5,25	
T_A	Faixa de Temperatura Ambiente de Operação	54	–55	25	125	°C
		74	0	25	70	
I_{OH}	Corrente de Saída – Nível ALTO	54, 74			–0,4	mA
I_{OL}	Corrente de Saída – Nível BAIXO	54			4,0	mA
		74			8,0	

▲ **FIGURA 3–65**

Folhas de dados parciais para um 74LS00.*

SEÇÃO 3–8 REVISÃO

1. Faça uma lista contendo os dois tipos de tecnologias de CIs que são as mais usadas.
2. Especifique os seguintes identificadores lógicos de CIs:
   (a) LS **(b)** ALS **(c)** F **(d)** HC **(e)** AC **(f)** HCT **(g)** LV
3. Identifique os seguintes dispositivos de acordo com a função lógica:
   (a) 74LS04 **(b)** 74HC00 **(c)** 74LV08 **(d)** 74ALS10
   (E) 7432 **(F)** 74ACT11 **(G)** 74HC02
4. Qual tecnologia de CI tem geralmente a menor dissipação de potência?
5. O que significa o termo hex inverter? O que significa quad 2-input NAND?
6. Um pulso positivo é aplicado na entrada de um inversor. O tempo entre a borda de subida na entrada e a borda de subida na saída é 10 ns. O tempo entre a borda de descida na entrada e a borda de descida na saída é 8 ns. Quais são os valores de t_{PHL} e t_{PLH}?
7. Uma certa porta tem um atraso de propagação de 6 ns e uma dissipação de potência de 3 mW. Determine o produto velocidade-potência.
8. Defina I_{CCL} e I_{CCH}.
9. Defina V_{IL} e V_{IH}.
10. Defina V_{OL} e V_{OH}.

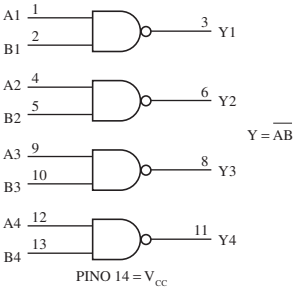
* N. de T.: Algumas folhas de dados são apresentadas em português. Assim, o leitor terá facilidade para identificar os parâmetros tratados no texto. Entretanto, o mais comum é encontrar folhas de dados em inglês. A sugestão para o leitor é pesquisar em folhas de dados em inglês os mesmos componentes apresentados em português neste livro com a finalidade de se familiarizar com os parâmetros em inglês.

Quatro Portas NAND de 2 Entradas Porta CMOS de silício de alto desempenho

O MC54/74HC00A é idêntico ao LS00 na configuração dos pinos. As entradas dos dispositivo são compatíveis com as saídas CMOS padrão; com resistores pullup, eles são compatíveis com saídas TTL LS.

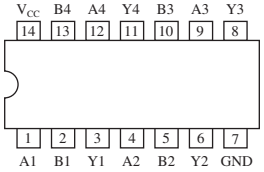
- Capacidade de Acionamento da Saída: 10 cargas TTL LS
- Saídas Interfaceadas Diretamente com CMOS, NMOS e TTL
- Faixa de Tensão de Operação: 2 a 6 V
- Corrente de Entrada em Nível Baixo: 1 μA
- Características de Alta Imunidade a Ruído de Dispositivos CMOS
- Em Concordância com os Requisitos do Padrão nº 7A da JEDEC
- Complexidade do Chip: 32 FETs ou 8 Portas Equivalentes

DIAGRAMA LÓGICO

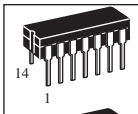


PINO 14 = V_{CC}
PINO 7 = GND

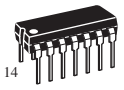
Pinagem: Encapsulamento de 14 Pinos (Vista Superior)



MC54/74HC00A



SUFIXO J
ENCAPSULAMENTO CERÂMICO
ENCAPSULAMENTO 632-08



SUFIXO N
ENCAPSULAMENTO PLÁSTICO
ENCAPSULAMENTO 646-06

SUFIXO D
ENCAPASULAMENTO SOIC
ENCAPSULAMENTO 751A-02

SUFIXO DT
ENCAPASULAMENTO TSSOP
ENCAPSULAMENTO 948G-01

SEQÜÊNCIA DA INFORMAÇÃO

MC54HCXXAJ	Cerâmico
MC74HCXXAN	Plástico
MC74HCXXAD	SOIC
MC74HCXXADT	TSSOP

TABELA DE FUNÇÕES

Entradas		Saídas
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

ESPECIFICAÇÕES MÁXIMAS*

Símbolo	Parâmetro	Valor	Unid.
V_{CC}	Tensão de Alimentação CC (Referenciada a GND)	–0,5 a + 7,0V	
V_{in}	Corrente de Entrada CC por Pino	–0,5 a V_{CC} + 0,5V	
V_{out}	Corrente de Saída CC por Pino	–0,5 a V_{CC} + 0,5V	
I_{in}	Corrente CC de Entrada por Pino	± 20	mA
I_{out}	Corrente CC de Saída por Pino	± 25	mA
I_{CC}	Corrente de Alimentação CC nos Pinos V_{CC} e GND	± 50	mA
P_D	Dissipação de Pot. sem Vent.: DIP Plástico ou Cerâmico †	750	mW
	Encapsulamento SOIC †	500	
	Encapsulamento TSSOP †	450	
T_{stg}	Temperatura de armazenamento	–65 a + 150	°C
T_L	Temp. do terminal a 1 mm do encapsulamento dur. 10 seg.		°C
	Encapsulamento DIP Plástico, SOIC ou TSSOP	260	
	DIP Cerâmico	300	

*Especificações Máximas são aqueles valores além dos quais podem ocorrer danos aos dispositivos. A operação funcional deve ser restrita às condições de operações recomendadas.

† Fator de diminuição – DIP Plástico: – 10 mW/ºC a partir de 65º a 125ºC
DIP Cerâmico – 10 mW/ºC a partir de 100º a 125ºC
Encapsulamento SOIC – 7 mW/ºC a partir de 65º a 125ºC
Encapsulamento TSSOP – 6,1 mW/ºC a partir de 65º a 125ºC

CONDIÇÕES DE OPERAÇÕES RECOMENDADAS

Símbolo	Parâmetro	Min.	Max.	Unid.
V_{CC}	Tensão de Alimentação CC (Referenciada a GND)	2,0	6,0	V
V_{in}, V_{out}	Voltagem CC de Entrada, Voltagem de Saída (Referenciada a GND)	0	V_{CC}	V
T_A	Temp. de Operação para Todos os Tipos de Encapsulamentos	–55	+125	°C
t_r, t_f	Tempo de Subida e Descida na Entrada V_{CC} = 2,0 V	0	1000	ns
	V_{CC} = 4,5 V	0	500	
	V_{CC} = 6,0 V	0	400	

CARACTERÍSTICAS CC (Tensões Referenciadas a GND) MC54/74HC00A

Símbolo	Parâmetro	Condição	V_{CC} V	Limites Garantidos –55 a 25°C	≤85°C	≤125°C	Unid.
V_{IH}	Mínima Tensão de Entrada em Nível Alto	V_{out} = 0,1V ou V_{CC} – 0,1V, $\|I_{out}\| \le 20\mu A$	2,0	1,50	1,50	1,50	V
			3,0	2,10	2,10	2,10	
			4,5	3,15	3,15	3,15	
			6,0	4,20	4,20	4,20	
V_{IL}	Máxima Tensão de Entrada em Nível Baixo	V_{out} = 0,1V ou V_{CC} – 0,1V, $\|I_{out}\| \le 20\mu A$	2,0	0,50	0,50	0,50	V
			3,0	0,90	0,90	0,90	
			4,5	1,35	1,35	1,35	
			6,0	1,80	1,80	1,80	
V_{OH}	Mínima Tensão de Saída em Nível Alto	V_{in} = V_{IH} ou V_{IL}, $\|I_{out}\| \le 20\mu A$	2,0	1,9	1,9	1,9	V
			4,5	4,4	4,4	4,4	
			6,0	5,9	5,9	5,9	
		V_{in} = V_{IH} ou V_{IL}, $\|I_{out}\| \le 2,4mA$	3,0	2,48	2,34	2,20	
		$\|I_{out}\| \le 4,0mA$	4,5	3,98	3,84	3,70	
		$\|I_{out}\| \le 5,2mA$	6,0	5,48	5,34	5,20	
V_{OL}	Máxima Tensão de Saída em Nível Baixo	V_{in} = V_{IH} ou V_{IL}, $\|I_{out}\| \le 20\mu A$	2,0	0,1	0,1	0,1	V
			4,5	0,1	0,1	0,1	
			6,0	0,1	0,1	0,1	
		V_{in} = V_{IH} ou V_{IL}, $\|I_{out}\| \le 2,4mA$	3,0	0,26	0,33	0,40	
		$\|I_{out}\| \le 4,0mA$	4,5	0,26	0,33	0,40	
		$\|I_{out}\| \le 5,2mA$	6,0	0,26	0,33	0,40	
I_{in}	Máxima corrente de Fuga de Entrada	V_{in} = V_{CC} ou GND	6,0	±0,1	±1,0	±1,0	μA
I_{CC}	Máxima Corrente de Alimentação Quiescente (por Encapsulamento)	V_{in} = V_{CC} ou GND, I_{out} = 0μA	6,0	1,0	10	40	μA

CARACTERÍSTICAS CA (C_L = 50 pF, Input t_r = t_f = 6 ns)

Símbolo	Parâmetro	V_{CC} V–	Limites Garantidos 55 a 25 °C	≤85°C	≤125°C	Unid.
t_{PLH}, t_{PHL}	Atraso de Propagação Máximo da Entrada A ou B para a Saída Y	2,0	75	95	110	ns
		3,0	30	40	55	
		4,5	15	19	22	
		6,0	13	16	19	
t_{TLH}, t_{THL}	Tempo de Transição de Saída Máximo para Qualquer Saída	2,0	75	95	110	ns
		3,0	27	32	36	
		4,5	15	19	22	
		6,0	13	16	19	
C_{in}	Capacitância Máxima de Entrada		10	10	10	pF

		Típico @ 25 °C, V_{CC} = 5,0 V, V_{EE} = 0 V	
C_{PD}	Capacitância de Dissipação de Potência (por Buffer)	22	pF

▲ **FIGURA 3–66**

Folha de dados parcial para um 74HC00A.

3-9 ANÁLISE DE DEFEITO

A análise de defeito é o processo de reconhecer, isolar e corrigir um defeito ou falha num circuito ou sistema. Para ser um técnico de manutenção efetivo, o leitor precisa entender como o circuito ou sistema deve funcionar e ser capaz de reconhecer os problemas de funcionamento. Por exemplo, para determinar se uma porta lógica específica está ou não com defeito, o técnico tem que saber qual deve ser a resposta de saída para determinadas entradas.

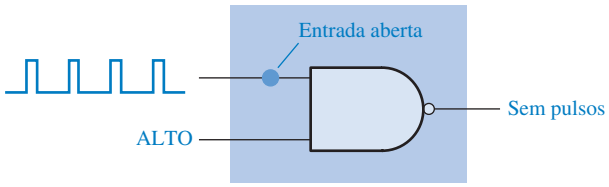
Ao final do estudo desta seção você deverá ser capaz de:

■ Testar CIs de portas lógicas pesquisando entradas e saídas abertas ■ Reconhecer os efeitos de uma entrada ou saída de CI em curto-circuito ■ Testar placas de circuito impresso pesquisando defeitos externos aos dispositivos ■ Fazer uma análise de defeito num freqüencímetro simples usando um osciloscópio

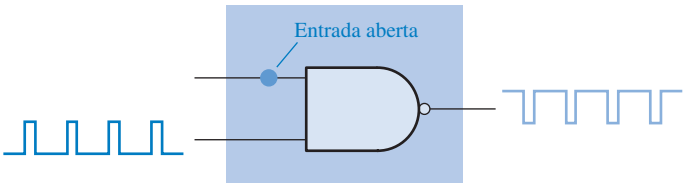
Falhas Internas em CIs de Portas Lógicas

Curtos-circuitos e circuitos abertos são os tipos mais comuns de defeitos internos às portas lógicas. Esses podem ocorrer nas entradas ou na saída de uma porta dentro do encapsulamento de um CI. *Antes de voltar a atenção em busca de qualquer defeito, verifique se as tensões de alimentação e GND são adequadas.*

Efeitos de uma Entrada Aberta Internamente Um circuito aberto internamente é o resultado de um componente aberto dentro do chip ou uma ruptura no pequeníssimo fio que interliga o chip do CI com o terminal do encapsulamento. Uma entrada aberta evita que um sinal na entrada chegue à saída da porta, conforme ilustra a Figura3–67(a) para o caso de uma porta NAND de 2 entradas. Uma entrada TTL aberta funciona efetivamente como um nível ALTO, de forma que os pulsos aplicados numa entrada boa chegam na saída de uma porta NAND como mostra a Figura 3–67(b).



(a) A aplicação de pulsos numa entrada aberta não produz pulsos na saída.



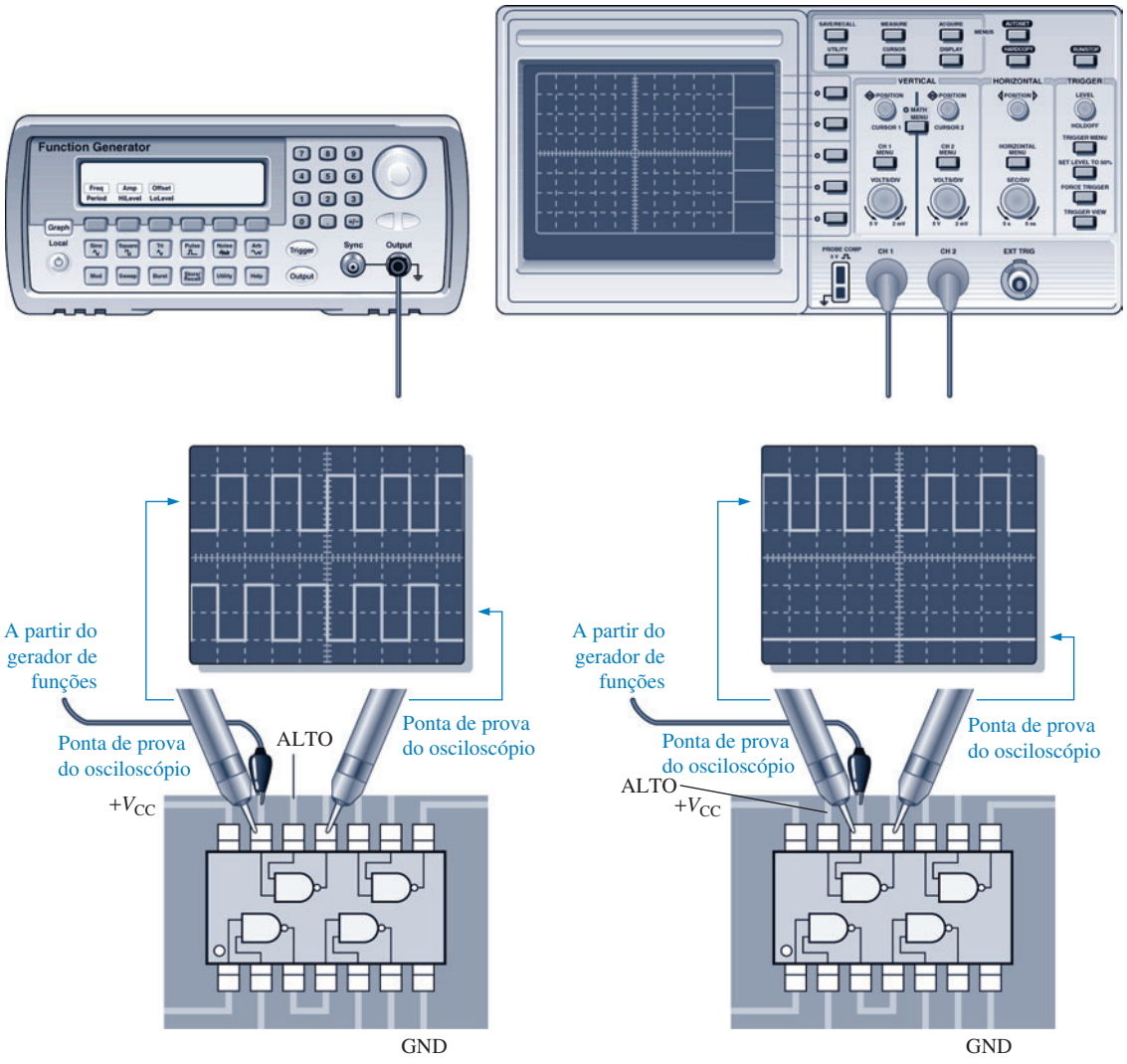
(b) A aplicação de pulsos numa entrada em perfeito estado produz pulsos na saída para portas AND e NAND TTL porque uma entrada aberta funciona tipicamente como se estivesse em nível ALTO. No caso de um dispositivo CMOS, o estado é incerto.

▲ **FIGURA 3–67**

O efeito de uma entrada aberta numa porta NAND.

Condições para o Teste de Portas No teste de uma porta NAND ou porta AND, certifique sempre se as entradas sem sinal estão em nível ALTO para habilitar a porta. Na verificação do funcionamento de uma porta NOR ou porta OR, certifique sempre se as entradas sem sinal estão em nível BAIXO. No teste de uma porta EX-OR ou EX-NOR, não importa o nível lógico na entrada sem sinal porque os pulsos na outra entrada forçam as entradas a se alternarem entre os mesmos níveis lógicos e níveis lógicos opostos.

Análise de Defeito para Entradas Abertas A análise de defeito desse tipo de falha é facilmente realizada com o uso de um osciloscópio e um gerador de funções, conforme demonstra a Figura 3–68 para o caso de um encapsulamento com portas NAND de 2 entradas. Ao medir os sinais digitais com um osciloscópio, sempre use o acoplamento cc (dc).



(a) A entrada no pino 13 e a saída no pino 11 estão OK.

(b) A entrada no pino 12 está aberta.

▲ **FIGURA 3–68**

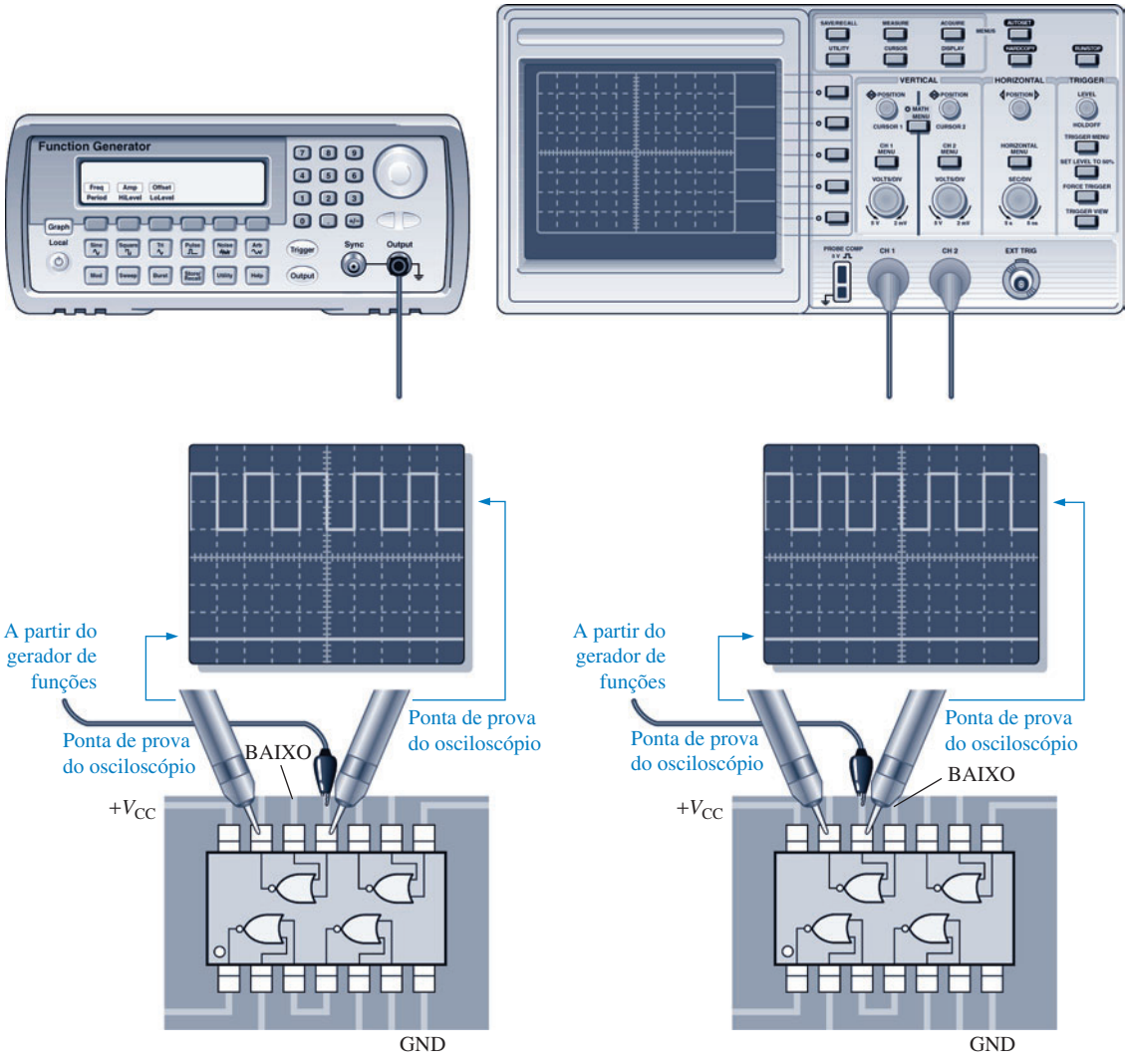
Análise de defeito numa porta NAND com uma entrada aberta.

O primeiro passo na análise de defeito num CI suspeito de estar com falhas é certificar que a tensão e alimentação (V_{CC}) e GND estão presentes nos pinos apropriados do CI. Em seguida, aplique pulsos numa das entradas da porta, certificando que a outra entrada esteja em nível ALTO (no caso de uma porta NAND). A Figura 3–68(a) mostra a aplicação de uma forma de onda digital no pino 13, que uma das entradas da porta suspeita. Se uma forma de onda digital aparecer na saída (pino 11 nesse caso), então a entrada no pino 13 não está aberta. A propósito, isso também prova que a saída não está aberta. Em seguida, aplique a forma de onda digital na outra entrada da porta (pino 12), garantindo que a outra entrada seja nível ALTO. Não existe forma de onda digital na saída (pino 11) e a saída está em nível BAIXO, indicando que a entrada no pino 12 está aberta, conforme mostra a Figura 3–68(b). A entrada sem pulsos tem que ser nível ALTO para o caso da porta NAND ou porta AND. Se fosse uma porta NOR, a entrada sem sinal teria que estar em nível BAIXO.

Efeitos de uma Saída Aberta Internamente A saída de uma porta aberta internamente evita que um sinal em qualquer das entradas chegue até ela. Portanto, não importa quais são as condições das entradas, a saída não é afetada. O nível na saída do pino do CI depende do que está co-

nectado externamente. O nível pode ser ALTO, BAIXO ou flutuação (sem referência fixa). De qualquer forma, não haverá sinal no pino de saída.

Análise de Defeito para uma Saída Aberta A Figura 3–69 ilustra a análise de defeito com a saída de uma porta NOR aberta. Na parte (a) da figura, uma das entradas da porta suspeita (nesse caso o pino 11) recebe os pulsos e a saída (pino 13) não apresenta uma forma de onda digital. Na parte (b) da figura, a outra entrada (pino 12) recebe pulsos e novamente não há uma forma de onda digital na saída. Sabendo que a entrada sem sinal está em nível BAIXO, esse teste mostra que a saída está aberta internamente.



(a) Entrada de pulsos no pino 11. Sem pulsos na saída.

(b) Entrada de pulsos no pino 12. Sem pulsos na saída.

▲ **FIGURA 3–69**

Análise de defeito de uma porta NOR com uma saída aberta.

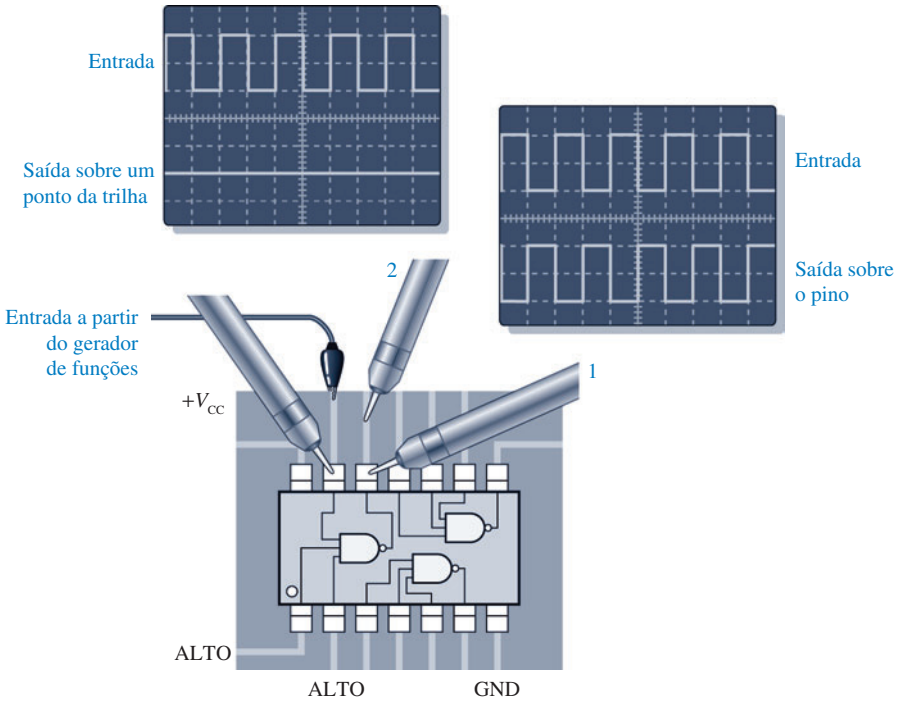
Entrada ou Saída em Curto-Circuito Embora não seja tão comum como um circuito aberto, pode ocorrer um curto-circuito interno para a tensão de alimentação cc, GND, outra entrada ou uma saída. Quando uma entrada ou saída está em curto-circuito para a tensão de alimentação, ela está presa ao nível ALTO. Se uma entrada ou saída estiver em curto-circuito com GND, ela estará presa no nível BAIXO (0 V). Se duas entradas ou uma entrada e uma saída estão em curto-circuito entre si, elas estarão sempre no mesmo nível.

Pontos Externos Abertos ou em Curto-Circuito

Muitos dos defeitos que envolvem os CIs digitais são devido a falhas externas ao encapsulamento do CI. Isso inclui conexões de solda fria, pingos de solda, pontes de fio, traçado inadequado da placa de circuito impresso e trincas ou interrupções em fios ou interconexões na placa de circuito. Essas condições de circuito aberto ou curto-circuito têm o mesmo efeito na porta lógica que as falhas internas e a análise de defeito é feita basicamente da mesma forma. Uma inspeção visual de qualquer circuito sob suspeita de defeito é a primeira coisa que um técnico deve fazer.

EXEMPLO 3–24

Considere que você esteja verificando o funcionamento de uma porta NAND de 3 entradas (74LS10) que é um dos diversos CIs de uma PCB. Você já verificou os pinos 1 e 2 e eles estão em nível ALTO. Agora você aplica uma forma de onda digital no pino 13 e coloca a ponta de prova do osciloscópio primeiro no pino 12 e em seguida na trilha de conexão na PCB, conforme indicado na Figura 3–70. Baseado em suas observações na tela do osciloscópio, qual é o defeito aparente?



▲ FIGURA 3–70

Solução

A forma de onda com a ponta de prova na posição 1 mostra que existem pulsos na saída da porta no pino 12, porém não existem pulsos na trilha da PCB conforme indicado pela ponta de prova na posição 2. A porta está funcionando adequadamente, mas o sinal não passa do pino 12 do CI para a trilha na placa PCB.

Isso sugere que existe uma solda fria entre o pino 12 do CI e a ilha na PCB, estabelecendo uma condição de circuito aberto. O ponto de solda deve ser refeito e verificado novamente.

Problema relacionado

Se não existem pulsos nos dois pontos indicados na Figura 3–70, qual(is) o(s) defeito(s) associado(s) a essa situação?

Na maioria das vezes, fazemos análise de defeito em CIs que estão montados em PCBs ou em protótipos e interconectados com outros CIs. À medida que avançarmos no estudo desse livro, aprenderemos como diferentes tipos de CIs digitais são usados em conjunto para formarem sistemas. Entretanto, nesse momento estamos concentrados em CIs de portas individuais. Essa limitação não nos impede de analisar conceitos de sistemas num nível bem básico e simplificado.

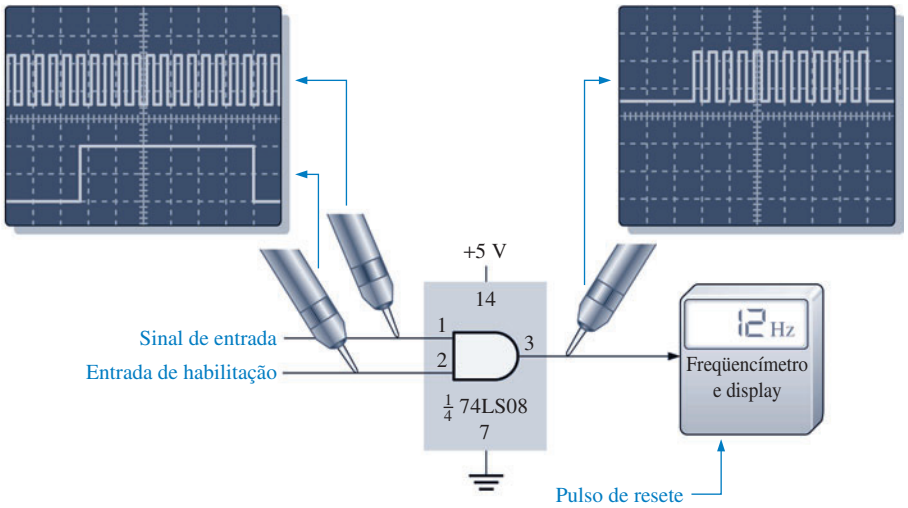
Dando continuidade na ênfase de sistemas, os Exemplos 3–25 e 3–26 tratam da análise de defeito num freqüencímetro apresentado na Seção 3–2.

EXEMPLO 3–25

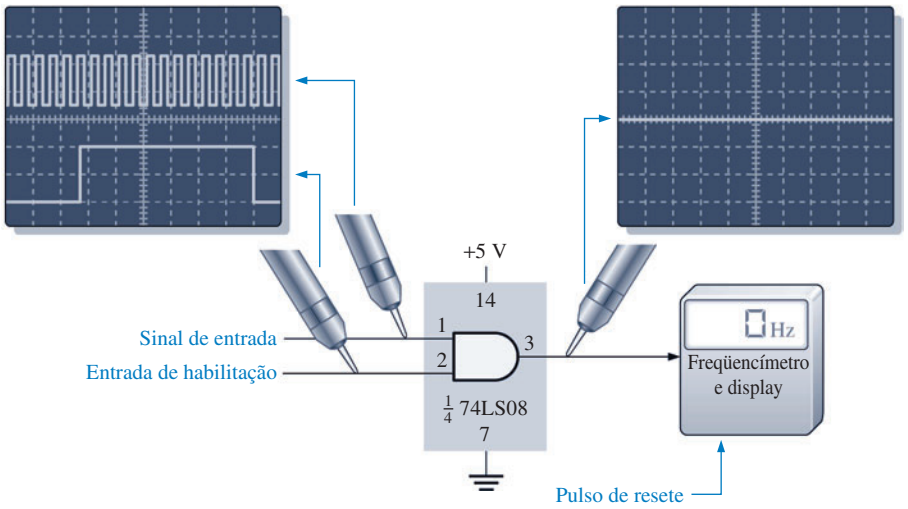
Após testar o funcionamento do freqüencímetro mostrado na Figura 3–71, o técnico constata que a leitura é sempre zero no display, independente da freqüência de entrada. Determine a causa desse mau funcionamento. O pulso de habilitação tem uma largura (duração) de 1 s.

A Figura 3–71(a) mostra um exemplo de como o freqüencímetro deveria funcionar com uma forma de onda digital de 12 Hz na entrada de uma porta AND. A parte (b) da mesma figura mostra que o display indica indevidamente 0 Hz.

► FIGURA 3–71



(a) O freqüencímetro está funcionando.



(b) O freqüencímetro indica que não há pulsos.

Solução Existem três causas possíveis:

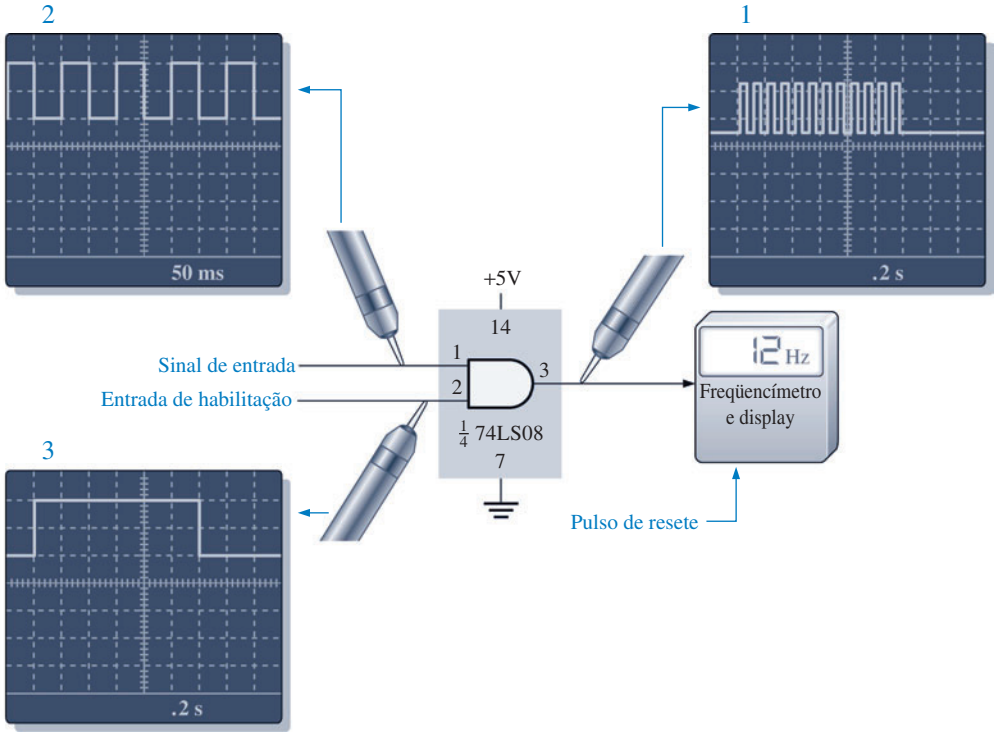
1. Um nível ativo ou acionado constante na entrada de resete, o que mantém o contador em zero.
2. Sem sinal na entrada do freqüencímetro devido a um circuito aberto ou curto-circuito interno ao freqüencímetro. Esse problema impediria o contador do freqüencímetro de avançar após sofrer um resete para zero.
3. Não existe sinal na entrada do contador devido a um circuito aberto na saída da porta AND ou devido a ausência de sinal na entrada, impedindo novamente o contador de avançar a partir do zero.

O primeiro passo é garantir que V_{CC} e GND estão conectados nos pontos certos; considere que eles estejam corretamente conectados. Em seguida, verifique os pulsos nas duas entradas da porta AND. O osciloscópio indica que os pulsos estão corretos nas duas entradas. Uma verificação na entrada de resete do contador mostra um nível BAIXO que é o nível não ativo e, portanto, esse não é o problema. A verificação do pino 3 do 74LS08 mostra que não existem pulsos na saída da porta AND, indicando que a saída da porta está aberta. Substitua o CI 74LS08 e verifique a operação novamente.

Problema relacionado Se o pino 2 da porta AND no CI 74LS08 estiver aberto, qual indicação aparecerá no display do freqüencímetro?

EXEMPLO 3–26

O freqüencímetro mostrado na Figura 3–72 mede incorretamente a freqüência do sinal de entrada. Verifica-se quando um sinal com uma freqüência de precisão conhecida é aplicado



▲ **FIGURA 3–72**

no pino 1 da porta AND, o osciloscópio indica uma freqüência maior. Determine o que está errado. A leitura na tela indica sec/div.

Solução Lembre-se, da Seção 3–2, de que é permitida a passagem dos pulsos de entrada através da porta AND durante exatamente 1 s. O número de pulsos contados em 1 s é igual a freqüência em hertz (ciclos por segundo). Portanto, o intervalo de 1 s, no qual é gerado o pulso de habilitação no pino 2 da porta AND, é muito crítico para a precisão da medida da freqüência. Os pulsos de habilitação são produzidos internamente por um circuito oscilador de precisão. Os pulsos têm que ter exatamente uma largura de 1 s e nesse caso o contador é atualizado a cada 3 s. Exatamente antes da habilitação de cada pulso, o contador é resetado para zero de forma que ele inicia uma nova contagem a cada vez. Como o contador se mostra estar contando mais pulsos, os quais deveriam ser produzido por uma freqüência maior, o pulso de habilitação é o principal suspeito. A medida exata do intervalo de tempo tem que ser feita com o osciloscópio.

Uma forma de onda digital na entrada de exatamente 10 Hz é aplicada no pino 1 da porta AND e o display mostra incorretamente 12 Hz. A primeira medição feita com o osciloscópio na saída da porta AND mostra que existe m 12 pulsos para cada pulso de habilitação. Na segunda medição com o osciloscópio, verifica-se que a freqüência de entrada é precisamente 10 Hz (período = 100 ms). Na terceira medição com o osciloscópio, a largura do pulso de habilitação medido é de 1,2 s em vez de 1 s.

A conclusão é que o pulso de habilitação está descalibrado por alguma razão.

Problema relacionado Do que você suspeitaria se a leitura da freqüência fosse menor que a freqüência real?

SEÇÃO 3–9 REVISÃO

1. Quais são os tipos mais comuns de defeitos em CIs?
2. Se duas formas de onda diferentes são aplicadas nas entradas de uma porta NAND TTL de 2 entradas e a forma de onda de saída é exatamente igual à forma de onda de uma das entradas, porém invertida, qual é o problema mais provável?
3. Cite duas características de formas de onda digitais que podem ser medidas no osciloscópio.

DICA PRÁTICA

O aterramento adequado é muito importante quando fazemos medições ou trabalhamos com um circuito. O aterramento apropriado do osciloscópio nos protege de choques e o nosso próprio aterramento protege o circuito de danos. Aterrar o osciloscópio significa conectá-lo à Terra através de uma tomada de três pinos. Aterrar a nós mesmos significa usar uma pulseira anti-estática, particularmente quando trabalhamos com circuitos CMOS.

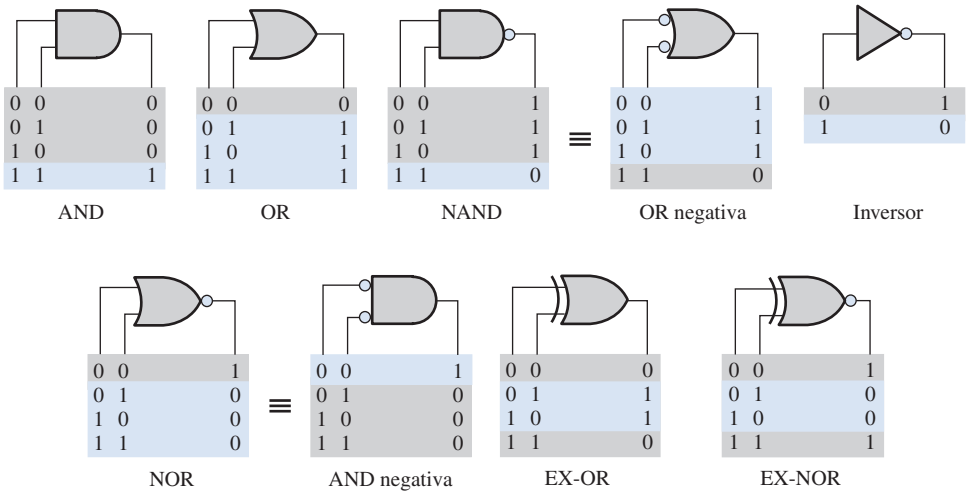
Além disso, para medições mais precisas, certifique-se de que o terra do circuito sob teste seja o mesmo do terra do osciloscópio. Isso pode ser feito conectando-se o terra da ponta de prova do osciloscópio num ponto de terra conhecido do circuito, tal como o chassi metálico ou um ponto de terra da placa de circuito. Podemos também conectar o terra do circuito ao conector GND no painel frontal do osciloscópio.

Os problemas de análise de defeito que são encontrados no CD-ROM estão disponíveis na seção Práticas de Análise de Defeito Usando o Multisim no final dos problemas propostos desse capítulo.

RESUMO

- A saída de um inversor é o complemento da entrada.
- A saída de uma porta AND é nível ALTO apenas quando todas as entradas são nível ALTO.
- A saída de uma porta OR é nível ALTO quando qualquer uma das entradas for nível ALTO.
- A saída de uma porta NAND é nível BAIXO apenas quando todas as entradas estiverem em nível ALTO.
- Uma NAND pode ser vista como uma OR negativa cuja saída é nível ALTO quando qualquer entrada for nível BAIXO.
- A saída de uma porta NOR é nível BAIXO quando qualquer uma das entradas for nível ALTO.
- Uma NOR pode ser vista como uma AND negativa cuja saída é nível ALTO apenas quando todas as entradas estiverem em nível BAIXO.
- A saída de uma porta EX-OR é nível ALTO quando as entradas estão em níveis diferentes.
- A saída de uma porta EX-NOR é nível BAIXO quando as entradas estão em níveis diferentes.
- Os símbolos característicos e as tabelas-verdade para várias portas lógicas (limitada a 2 entradas) são mostradas na Figura 3–73.

▶ **FIGURA 3–73**



Nota: Os estados ativos são mostrados em laranja.

- A maioria dos dispositivos lógicos programáveis (PLDs) se baseia em alguma forma de arranjo AND.
- A tecnologia de conexões programáveis são: fusível, antifusível, EPROM, EEPROM e SRAM.
- Um PLD pode ser programado num equipamento denominado de programador ou montado numa PCB.
- Os PLDs têm um pacote associado de software de desenvolvimento para programação.
- Os dois métodos de inserção de projeto que usam software para programação são: inserção via texto (HDL) e inserção gráfica (esquemático).
- Os PLDs tipo ISP podem ser programados após serem instalados no sistema.
- O padrão JTAG foi estabelecido pelo Joint Test Action Group e é uma interface padrão (padrão 1149.1 da IEEE) usada para programação e teste de PLDs.
- Um processador embutido é usado para facilitar a programação de PLDs dentro do sistema.
- A tecnologia CMOS é implementada com transistores de efeito de campo.
- A tecnologia TTL é implementada com transistores de junção bipolar.
- Como via de regra, os dispositivos CMOS têm um consumo de potência menor que os dispositivos TTL.

- A dissipação de potência média de uma porta lógica é dada por:

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

- O produto velocidade-potência de uma porta lógica é dado por:

$$SPP = t_p P_D$$

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Álgebra Booleana A matemática dos circuitos lógicos.

Antifusível Um tipo de conexão programável não volátil de PLD que pode ser deixada aberta ou pode ser colocada em curto-circuito uma vez conforme determinado pelo programa.

Arranjo AND Um arranjo de portas AND consiste em uma matriz de interconexões programáveis.

Carga unitária Uma medida do fan-out. Uma entrada de porta representa uma unidade de carga para a saída de uma porta lógica dentro da mesma família de CIs.

CMOS Semicondutor de óxido metálico complementar; uma classe de circuitos lógicos integrados implementados com um tipo de transistor de efeito de campo.

Complemento O inverso ou o oposto de um número. O nível BAIXO é o complemento do nível ALTO e 0 é o complemento de 1.

Diagrama de temporização Um diagrama de formas de onda que mostra a relação temporal de todas as formas de onda envolvidas.

Dispositivo destino Um PLD montado num equipamento de programação ou placa de desenvolvimento no qual se faz a transferência (download) do projeto lógico na forma de software.

EEPROM Tipo de conexão programável não volátil de PLD baseada em células de memória apenas de leitura programável e apagável eletricamente podendo ser ligada ou desligada repetidas vezes por programação.

Habilitação Para ativar ou colocar no modo de operação; uma entrada de um circuito lógico que habilita a operação dele.

EPROM Tipo de conexão programável não volátil de PLD baseada em células de memória apenas de leitura programável eletricamente podendo ser ligada ou desligada uma vez por programação.

Fan-out O número de entradas de portas equivalentes da mesma série de uma família que uma porta lógica é capaz de acionar.

Fusível Um tipo de conexão programável não volátil de PLD que pode ser deixada intacta (curto-circuito) ou aberta uma vez conforme determinado pelo programa.

Inversor Um circuito lógico que inverte ou complementa sua entrada.

JTAG Joint Test Action Group; uma interface padrão projetada pela IEEE (padrão 1149.1)

Porta AND Uma porta lógica que produz uma saída de nível ALTO apenas quando todas as entrada estiverem em nível ALTO.

Porta EX-NOR Uma porta lógica que produz uma saída de nível BAIXO apenas quando suas duas entradas estiverem em níveis opostos.

Porta EX-OR Uma porta lógica que produz uma saída de nível ALTO apenas quando suas duas entradas estiverem em níveis opostos.

Porta NAND Uma porta lógica que produz uma saída de nível BAIXO apenas quando todas as entradas estão em nível ALTO.

Porta NOR Uma porta lógica na qual a saída é nível BAIXO quando uma ou mais entradas estiverem em nível ALTO.

Porta OR Uma porta lógica que produz uma saída de nível ALTO quando uma ou mais entradas estiverem em nível ALTO.

SRAM Um tipo de conexão programável volátil de PLD baseada em células de memória de acesso aleatório estática e que pode ser ligada ou desligada repetidas vezes por programação.

Tabela-verdade Uma tabela que mostra as entradas e a(s) correspondente(s) saída(s) de um circuito lógico.

Tempo de atraso de propagação O intervalo de tempo entre a ocorrência de uma transição de entrada e a correspondente transição de saída num circuito lógico.

TTL Lógica transistor transistor; uma classe de circuitos lógicos integrados que usa transistores de junção bipolar.

AUTOTESTE

As respostas estão no final do capítulo.

1. Quando a entrada de um inversor é nível ALTO (1) a saída é
(a) nível ALTO ou 1 **(b)** nível BAIXO ou 1 **(c)** nível ALTO ou 0 **(d)** nível BAIXO ou 0

2. Um inversor realiza uma operação conhecida como
(a) complementação **(b)** acionamento
(c) inversão **(d)** as opções (a) e (c) estão corretas.

3. A saída de uma porta AND com entradas A, B e C é 1 (ALTO) quando
(a) $A = 1, B = 1, C = 1$ **(b)** $A = 1, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$

4. A saída de uma porta OR com entradas A, B e C é 1 (ALTO) quando
(a) $A = 1, B = 1, C = 1$ **(b)** $A = 0, B = 0, C = 1$ **(c)** $A = 0, B = 0, C = 0$
(d) as opções (a), (b) e (c) estão corretas. **(e)** apenas as opções (a) e (b) estão corretas.

5. Cada uma das entradas de uma porta NAND de 2 entradas recebe pulso. Um pulso vai para nível ALTO em $t = 0$ e retorna para nível BAIXO em $t = 1$ ms. O outro pulso vai para nível ALTO em $t = 0{,}8$ ms e retorna para nível BAIXO em $t = 3$ ms. O pulso de saída pode ser descrito como:
(a) Ele vai para nível BAIXO em $t = 0$ e retorna para nível ALTO em $t = 3$ ms.
(b) Ele vai para nível BAIXO em $t = 0{,}8$ ms e retorna para nível ALTO em $t = 3$ ms.
(c) Ele vai para nível BAIXO em $t = 0{,}8$ ms e retorna para nível ALTO em $t = 1$ ms.
(d) Ele vai para nível BAIXO em $t = 0{,}8$ ms e retorna para nível BAIXO em $t = 1$ ms.

6. Cada uma das entradas de uma porta NOR de 2 entradas recebe pulso. Um pulso vai para nível ALTO em $t = 0$ e retorna para nível BAIXO em $t = 1$ ms. O outro pulso vai para nível ALTO em $t = 0{,}8$ ms e retorna para nível BAIXO em $t = 3$ ms. O pulso de saída pode ser descrito como:
(a) Ele vai para nível BAIXO em $t = 0$ e retorna para nível ALTO em $t = 3$ ms.
(b) Ele vai para nível BAIXO em $t = 0{,}8$ ms e retorna para nível ALTO em $t = 3$ ms.
(c) Ele vai para nível BAIXO em $t = 0{,}8$ ms e retorna para nível ALTO em $t = 1$ ms.
(d) Ele vai para nível ALTO em $t = 0{,}8$ ms e retorna para nível BAIXO em $t = 1$ ms.

7. Cada uma das entradas de uma porta EX-OR recebe pulso. Um pulso vai para nível ALTO em $t = 0$ e retorna para nível BAIXO em $t = 1$ ms. O outro pulso vai para nível ALTO em $t = 0{,}8$ ms e retorna para nível BAIXO em $t = 3$ ms. O pulso de saída pode ser descrito como:
(a) Ele vai para nível ALTO em $t = 0$ e retorna para nível BAIXO em $t = 3$ ms.
(b) Ele vai para nível ALTO em $t = 0$ e retorna para nível BAIXO em $t = 0{,}8$ ms.
(c) Ele vai para nível ALTO em $t = 1$ ms e retorna para nível BAIXO em $t = 3$ ms.
(d) As opções (b) e (c) estão corretas.

8. Um pulso positivo é aplicado num inversor. O intervalo de tempo a partir da borda de subida na entrada até a borda de subida na saída é 7 ns. Esse parâmetro é
(a) o produto velocidade-potência
(b) o atraso de propagação t_{PHL}
(c) o atraso de propagação t_{PLH}
(d) largura de pulso

9. A finalidade de uma conexão programável num arranjo AND é
(a) conectar uma variável de entrada a uma entrada de porta.
(b) conectar uma linha a uma coluna numa matriz de arranjo.
(c) desconectar uma linha de uma coluna numa matriz de arranjo.
(d) todas as opções acima estão corretas.

10. O termo OTP significa
(a) ponto de teste aberto **(b)** programável apenas uma vez
(c) programa de teste de saída **(d)** terminal de saída positivo

11. São tipos de tecnologias de processo usadas em conexões programáveis de PLDs
(a) antifusível **(b)** EEPROM
(c) ROM **(d)** as opções (a) e (b) estão corretas.
(e) as opções (a) e (c) estão corretas.

12. Uma tecnologia de conexão programável volátil é
(a) fusível **(b)** EPROM
(c) SRAM **(d)** EEPROM

13. Duas das formas de inserção de projeto lógico usando software de desenvolvimento de PLD são:
(a) texto e numérica **(b)** texto e gráfica
(c) gráfica e codificada **(d)** compilação e classificação

14. O padrão JTAG foi criado por
(a) Joint Test Action Group **(b)** Java Top Array Group
(c) Joint Test Array Group **(d)** Joint Time Analysis Group

15. A programação dentro do sistema de uma PLD utiliza tipicamente
(a) um gerador de clock embutido. **(b)** um processador embutido.
(c) uma PROM embutida. **(d)** as opções (a) e (b) estão corretas.
(e) as opções (b) e (c) estão corretas.

16. Para medir o período de uma forma de onda digital temos que usar
(a) um DMM (multímetro digital) **(b)** uma ponta de prova lógica
(c) um osciloscópio **(d)** um pulsador lógico

17. Uma vez medido o período de uma forma de onda digital, a freqüência dela é determinada:
(a) usando um outro parâmetro. **(b)** medindo o ciclo de trabalho.
(c) calculando o recíproco do período. **(d)** usando um outro tipo de instrumento.

PROBLEMAS

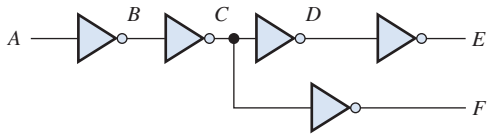
As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 3–1 O Inversor

1. A forma de onda de entrada mostrada na Figura 3–74 é aplicada num inversor. Desenhe o diagrama de temporização da forma e onda de saída devidamente relacionada com a entrada.

▶ FIGURA 3–74

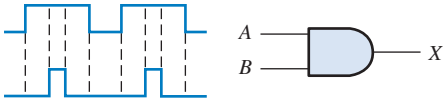
2. Um circuito de inversores em cascata é mostrado na Figura 3–75. Se um nível ALTO for aplicado no ponto *A*, determine os níveis lógicos nos pontos de *B* até *F*.



▶ FIGURA 3–75

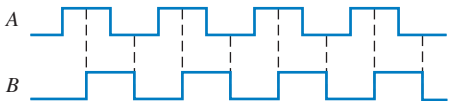
SEÇÃO 3–2 A Porta AND

3. Determine a saída, *X*, para uma porta AND de 2 entradas com as formas de onda de entrada mostradas na Figura 3–76. Mostre, através de um diagrama de temporização, a relação entre a saída e as entradas.



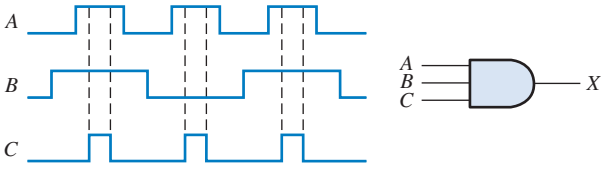
▶ FIGURA 3–76

4. Repita o Problema 3 considerando o diagrama de temporização mostrado na Figura 3–77.



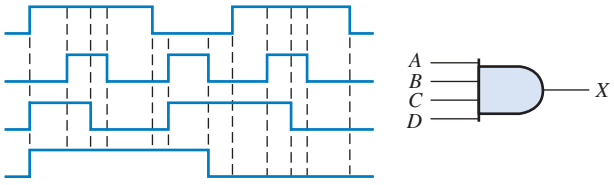
▶ FIGURA 3–77

5. As formas de onda de entrada aplicada numa porta AND de 3 entradas são mostradas na Figura 3–78. Mostre a forma de onda de saída relacionando-a adequadamente às entradas usando um diagrama de temporização.



▶ FIGURA 3–78

6. As formas de onda de entrada de uma porta AND de 4 entradas são vistas na Figura 3–79. Mostre a forma de onda de saída relacionando-a adequadamente às entradas usando um diagrama de temporização.



▶ FIGURA 3–79

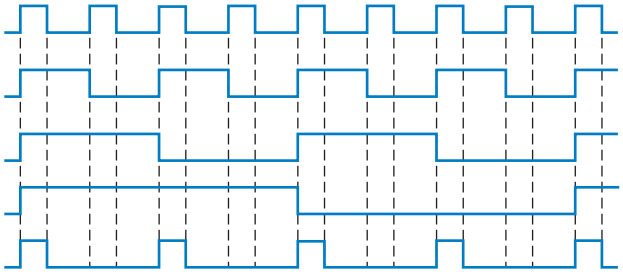
SEÇÃO 3–3 A Porta OR

7. Determine a saída de uma porta OR de 2 entradas quando as formas de onda de entrada forem as da Figura 3–77 e desenhe um diagrama de temporização.

8. Repita o Problema 5 para uma porta OR de 3 entradas.

9. Repita o problema 6 para uma porta OR de 4 entradas.

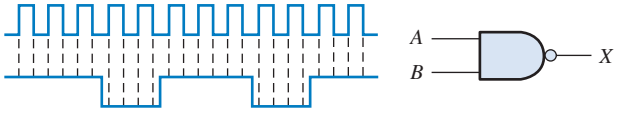
10. Para as cinco formas de onda de entrada vistas na Figura 3–80, determine a saída de uma porta AND de 5 entradas e a saída de uma porta OR de 5 entradas. Desenhe o diagrama de temporização.



▶ FIGURA 3–80

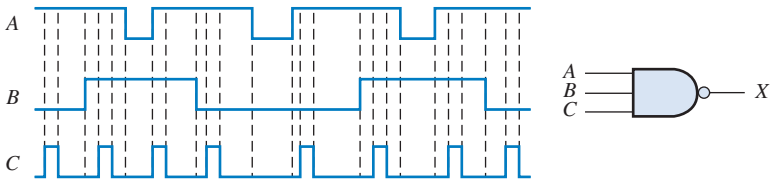
SEÇÃO 3–4 A Porta NAND

11. Para o conjunto de formas de onda de entrada vistas na Figura 3–81, determine a saída para a porta mostrada e desenhe o diagrama de temporização.



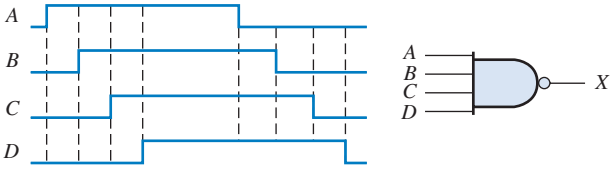
▶ FIGURA 3–81

12. Determine a saída da porta lógica (Figura 3–82) a partir das formas de onda de entrada vistas na mesma figura e desenhe o diagrama de temporização.



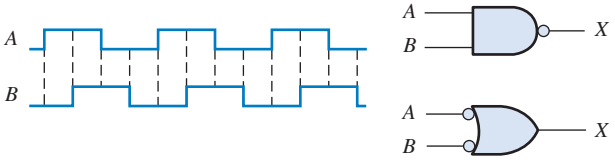
▶ FIGURA 3–82

13. Determine a forma de onda de saída para a porta e as formas de onda mostradas na Figura 3–83.



▶ FIGURA 3–83

14. Conforme aprendemos, os dois símbolos lógicos mostrados na Figura 3–84 representam operações equivalentes. A diferença entre os dois reside no ponto de vista funcional. Para o símbolo NAND, procuramos por dois níveis ALTOs nas entradas para termos um nível BAIXO na saída. Para a OR negativa, buscamos por pelo menos um nível BAIXO nas entradas para termos um nível ALTO na saída. Usando esses dois pontos de vista funcionais, mostre que cada porta produz a mesma saída para as entradas dadas.

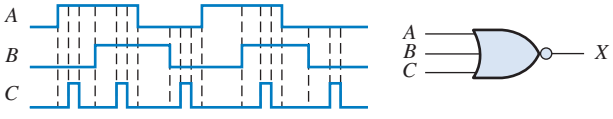


▶ FIGURA 3–84

SEÇÃO 3–5 A Porta NOR

15. Repita o Problema 11 para uma porta NOR de 2 entradas.

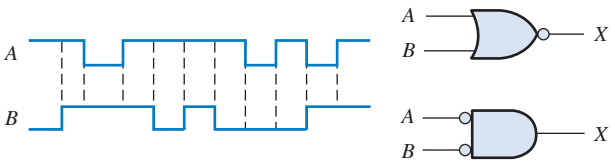
16. Determine a forma de onda de saída na Figura 3–85 e desenhe o diagrama de temporização.



▶ FIGURA 3–85

17. Repita o Problema 13 para uma porta NOR de 4 entradas.

18. Os símbolos da NAND e da OR negativa representam operações equivalentes, porém elas são portas diferentes funcionalmente. Para o símbolo da NOR, buscamos pelo menos um nível ALTO nas entradas para termos um nível BAIXO na saída. Para a AND negativa, buscamos dois níveis BAIXOs nas entradas para termos uma saída de nível ALTO. Usando esses dois pontos de vista funcionais, mostre que as portas vistas na Figura 3–86 produzem a mesma saída para as entradas dadas.



▶ FIGURA 3–86

SEÇÃO 3–6 As Portas OR Exclusivo e NOR Exclusivo

19. Em que uma porta EX-OR difere de uma porta OR na sua operação lógica?

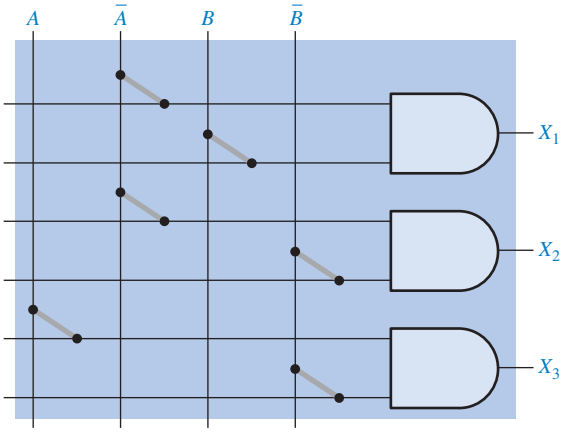
20. Repita o Problema 11 para uma porta EX-OR.

21. Repita o Problema 11 para uma porta EX-NOR.

22. Determine a saída de uma porta EX-NOR para as entradas mostradas na Figura 3–77 e desenhe um diagrama de temporização.

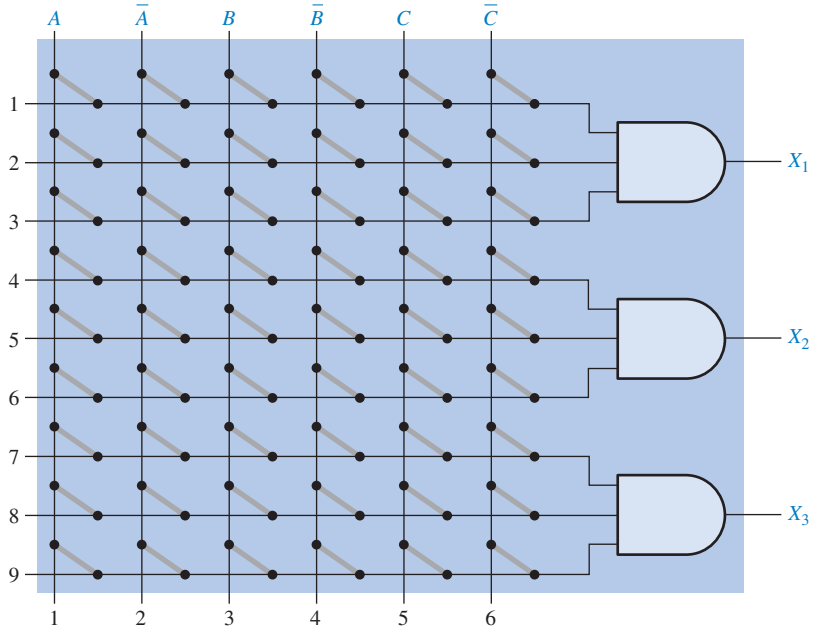
SEÇÃO 3–7 Lógica Programável

23. No arranjo AND programável simples com as conexões programáveis mostradas na Figura 3–87, determine a expressão Booleana de saída.



▶ FIGURA 3–87

24. Determine pela identificação das linhas e colunas quais conexões à fusível têm que ser "queimadas" no arranjo AND programável mostrado na Figura 3–88 para implementar cada um dos seguintes termos-produto: $X_1 = \overline{A}BC$, $X_2 = AB\overline{C}$, $X_3 = \overline{A}B\overline{C}$.



▶ FIGURA 3–88

SEÇÃO 3–8 Lógica de Funções Fixas

25. Comparando-se certos dispositivos lógicos, observa-se que a dissipação de potência para um tipo particular aumenta conforme a freqüência de operação aumenta. Esse dispositivo é TTL ou CMOS?

26. Usando as folhas de dados mostradas nas Figuras 3–65 e 3–66, determine o que se pede:

(a) A dissipação de potência de um dispositivo 74LS00 para uma tensão de alimentação máxima e um ciclo de trabalho de 50%.

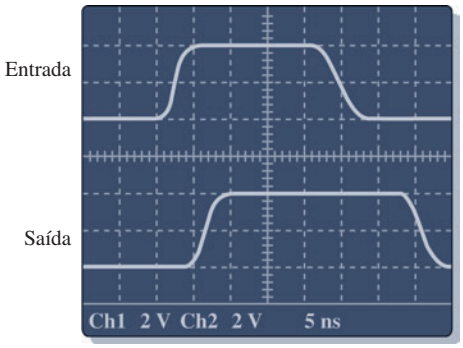
(b) A tensão de saída de nível ALTO mínima para um 74LS00.

(c) O atraso de propagação máximo para um 74LS00.

(d) A tensão de saída de nível BAIXO máxima para um 74HC00A.

(e) O atraso de propagação máximo para um 74HC00A

27. Determine o t_{PLH} e o t_{PHL} a partir da imagem da tela de um osciloscópio vista na Figura 3–89. A informação em baixo na tela indica V/div e sec/div para cada canal.



▶ FIGURA 3–89

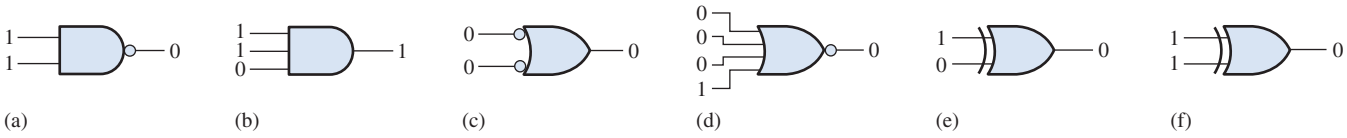
28. A porta *A* tem um $t_{PLH} = t_{PHL} = 6$ns. A porta *B* tem um $t_{PLH} = t_{PHL} = 10$ ns. Qual das portas pode operar numa freqüência maior?

29. Se uma porta lógica opera com uma tensão de alimentação cc de +5 V e drena uma corrente média de 4 mA, qual a dissipação de potência?

30. A variável ICCH representa a corrente de alimentação cc a partir de VCC quando todas as saídas do CI estiverem em nível ALTO. A variável ICCL representa a corrente de alimentação cc quando todas as saídas estão em nível BAIXO. Para um CI 74LS00, determine a dissipação de potência típica quando todas as saídas das quatro portas estiverem em nível ALTO. (Consulte a folha de dados mostrada na Figura 3–65).

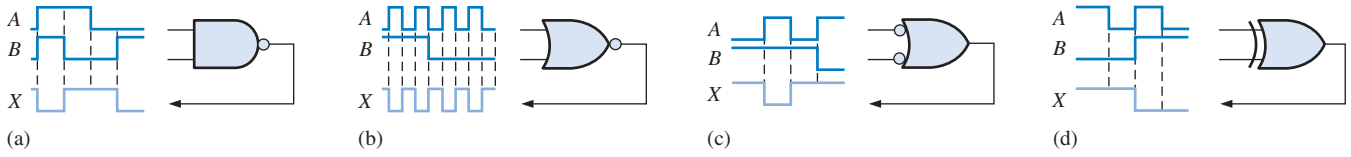
SEÇÃO 3–9 Análise de Defeito

31. Examine as condições indicadas na Figura 3–90 e identifique as portas com defeito.



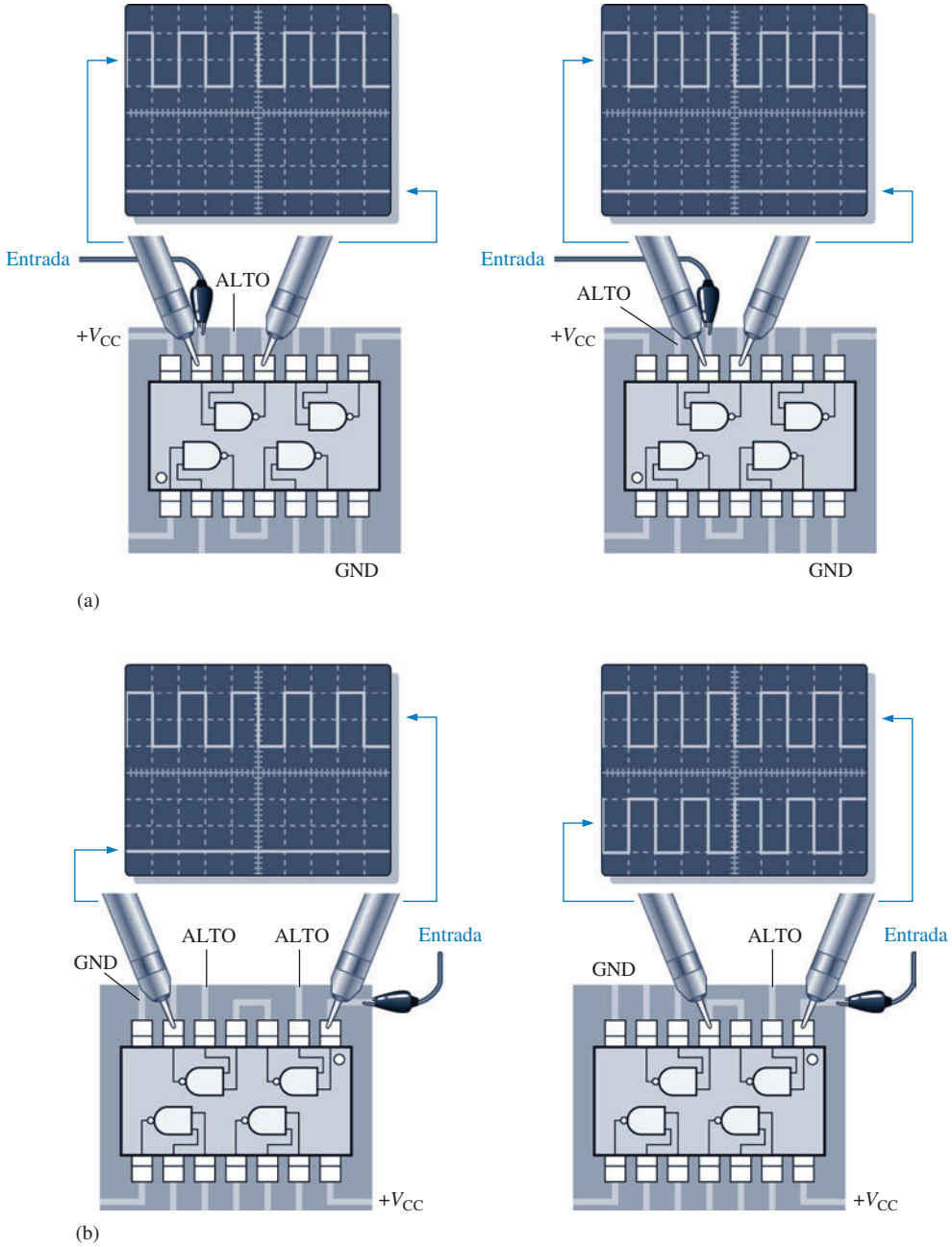
▲ FIGURA 3–90

32. Determine as portas com defeito na Figura 3–91 analisando os diagramas de temporização.



▲ FIGURA 3–91

33. Usando um osciloscópio, um técnico fez as observações indicadas na Figura 3–92. Para cada observação determine o defeito mais provável da porta.



▶ **FIGURA 3–92**

34. O circuito do alarme do cinto de segurança mostrado na Figura 3–16 não está funcionando. O técnico identificou que quando a chave de ignição é acionada e o cinto de segurança não está afivelado, o alarme dispara e não desliga. Qual o problema mais provável? Como você chegou a essa conclusão?

35. Todas as vezes que a chave de ignição é acionada no circuito visto na Figura 3–16, o alarme dispara por trinta segundos, mesmo que o cinto esteja afivelado. Qual é a causa mais provável desse funcionamento incorreto?

36. Qual(is) a(s) falha(s) suspeita(s) se uma porta NAND de 3 entradas permanece em nível ALTO independente dos estados das entradas?

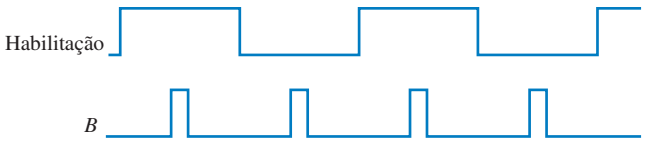
Problemas Especiais de Projeto

37. Sensores são usados para monitorar a pressão e a temperatura de uma solução química armazenada num recipiente fechado. O circuito de cada sensor produz uma tensão de nível ALTO quando o valor máximo é excedido. Um alarme que necessita de uma tensão de nível BAIXO na entrada tem que ser ativado quando a pressão ou a temperatura for excessiva. Projete um circuito para essa aplicação.

38. Em um certo processo produtivo automatizado, os componentes elétricos são automaticamente inseridos numa PCB. Antes que a ferramenta de inserção seja ativada, a PCB tem que ser posicionada corretamente e os componentes a serem inseridos têm que estar na câmara. Cada uma dessas condições é indicada por uma tensão de nível ALTO. A ferramenta de inserção requer uma tensão de nível BAIXO para ser ativada. Projete um circuito para implementar esse processo.

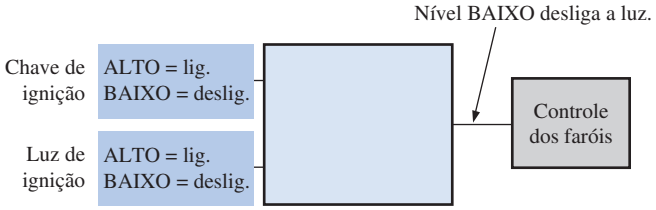
39. Modifique o freqüencímetro mostrado na Figura 3–15 para que opere com um pulso de habilitação ativo em nível BAIXO em vez de nível ALTO durante o intervalo de 1 s.

40. Considere que o sinal de habilitação do circuito mostrado na Figura 3–15 tenha a forma de onda mostrada na Figura 3–93. Considere que a forma de onda *B* esteja disponível. Projete um circuito que produza um pulso de inicialização ativo em nível ALTO para o contador apenas durante o tempo em que o sinal de habilitação esteja em nível BAIXO.



► FIGURA 3–93

41. Projete um circuito para substituir o bloco bege mostrado na Figura 3–94 para que o controle dos faróis de um automóvel seja desligado automaticamente 15 s após a chave de ignição ser desligada, estando ligada a chave que ativa os faróis. Considere que seja necessário um nível BAIXO para desligar os faróis.



► FIGURA 3–94

42. Modifique o circuito lógico para o alarme de intrusão mostrado na Figura 3–24 de forma que dois ambientes adicionais, cada um com duas janelas e uma porta, possa ser protegido.

43. Além da modificação proposta no Problema 42 altere o circuito dos sensores de entrada para que Aberto = BAIXO e Fechado = ALTO.

Prática de Análise de Defeito Usando o Multisim

44. Abra o arquivo P03-44, conecte um conversor lógico do Multisim ao circuito e observe a operação da porta AND. Baseado nas entradas e saída observadas, determine o defeito mais provável na porta.

45. Abra o arquivo P03-45, conecte um conversor lógico do Multisim ao circuito e observe a operação da porta NAND. Baseado nas entradas e saída observadas, determine o defeito mais provável na porta.

46. Abra o arquivo P03-46, conecte um conversor lógico do Multisim ao circuito e observe a operação da porta NOR. Baseado nas entradas e saída observadas, determine o defeito mais provável na porta.

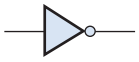
47. Abra o arquivo P03-47, conecte um conversor lógico do Multisim ao circuito e observe a operação da porta EX-OR. Baseado nas entradas e saída observadas, determine o defeito mais provável na porta.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 3–1 O Inversor

1. Quando a entrada do inversor é 1 a saída é 0.
2. **(a)**



(b) Um pulso negativo aparece na saída (de nível ALTO para BAIXO retornando para ALTO).

SEÇÃO 3–2 A porta AND

1. A saída de uma porta AND é nível ALTO apenas quando todas as entradas estiverem em nível ALTO.
2. A saída de uma porta AND é nível BAIXO quando uma ou mais entradas estiverem em nível BAIXO.
3. AND de cinco entradas: $X = 1$ quando $ABCDE = 11111$ e $X = 0$ para todas as outras combinações de $ABCDE$.

SEÇÃO 3–3 A Porta OR

1. A saída de uma porta OR é nível ALTO quando uma ou mais entradas estiverem em nível ALTO.
2. A saída de uma porta OR é nível BAIXO apenas quando todas as entradas estiverem em nível BAIXO.
3. OR de três entradas: $X = 0$ quando $ABC = 000$ e $X = 1$ para todas as outras combinações de ABC.

SEÇÃO 3–4 A Porta NAND

1. A saída de uma porta NAND é nível BAIXO apenas quando todas as entradas estiverem em nível ALTO.
2. A saída de uma porta NAND é nível ALTO quando uma ou mais entradas estiverem em nível BAIXO.
3. NAND: saída ativa em nível BAIXO para todas as entradas em nível ALTO; OR negativa: saída ativa em nível ALTO para uma ou mais entradas em nível BAIXO. Elas têm a mesmas tabelas verdade.
4. $X = \overline{ABC}$

SEÇÃO 3–5 A Porta NOR

1. A saída de uma porta NOR é nível ALTO apenas quando todas as entradas estiverem em nível BAIXO.
2. A saída de uma porta NOR é nível BAIXO quando uma ou mais entradas estiverem em nível ALTO.
3. NOR: saída ativa em nível BAIXO para uma ou mais entradas em nível ALTO; AND negativa: saída ativa em nível ALTO para todas as entradas em nível BAIXO. Elas têm as mesmas tabelas verdade.
4. $X = \overline{A + B + C}$

SEÇÃO 3–6 As Portas OR Exclusivo e NOR Exclusivo

1. A saída de uma porta EX-OR é nível ALTO quando as entradas estão em níveis opostos.
2. A saída de uma EX-NOR é nível ALTO quando as entradas estão no mesmo nível lógico.
3. Aplique os bits nas entradas da porta EX-OR; quando a saída é nível ALTO, os bits são diferentes.

SEÇÃO 3–7 Lógica Programável

1. Fusível, antifusível, EPROM, EEPROM e SRAM.
2. Volátil significa que todos os dados são perdidos quando a alimentação é desligada, sendo que o PLD tem que ser reprogramado; baseado em SRAM.

3. Inserção via texto e inserção gráfica.
4. JTAG é Joint Test Action Group; o padrão 1149.1 da IEEE para programação e teste de interface.

SEÇÃO 3–8 Lógica de Funções Fixas

1. CMOS e TTL
2. **(a)** LS – Schottky de baixa potência **(b)** ALS – LS avançado
   (c) F – TTL rápido **(d)** HC – CMOS de alta velocidade
   (e) AC – CMOS avançado **(f)** HCT – CMOS HC compatível com TTL
   (g) LV – CMOS de baixa tensão
3. **(a)** 74LS04 – Seis inversores **(b)** 74HC00 – Quatro NAND de 2 entradas
   (c) 74LV08 – Quatro AND de 2 entradas **(d)** 74ALS10 – Três NAND de 3 entradas
   (e) 7432 – Quatro OR de 2 entradas **(f)** 74ACT11 – Três AND de 3 entradas
   (g) 74AHC02 – Quatro NOR de 2 entradas
4. Menor potência – CMOS
5. Seis inversores num encapsulamento; quatro portas NAND de 2 entradas num encapsulamento
6. $t_{PLH} = 10$ ns; $t_{PHL} = 8$ ns
7. 18 pJ
8. I_{CCL} – corrente de alimentação cc para saída em estado BAIXO; ICCH – corrente de alimentação cc para saída em estado ALTO
9. V_{IL} – tensão de entrada em nível BAIXO; V_{IH} – tensão de entrada em nível ALTO
10. V_{OL} – tensão de saída em nível BAIXO; V_{OH} – tensão de saída em nível ALTO

SEÇÃO 3–9 Análise de Defeito

1. Circuito aberto e curto-circuito são os defeitos mais comuns.
2. Uma entrada aberta que efetivamente é como se a entrada fosse nível ALTO.
3. Amplitude e período.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

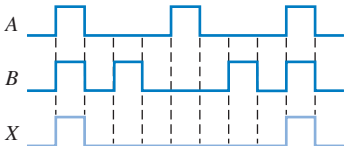
3–1. O diagrama de temporização não é afetado.

3–2. Veja a Tabela 3–13.

► **TABELA 3–13**

ENTRADAS *ABCD*	SAÍDA *X*	ENTRADAS *ABCD*	SAÍDA *X*
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	0	1110	0
0111	0	1111	1

3–3. Veja a Figura 3–95.

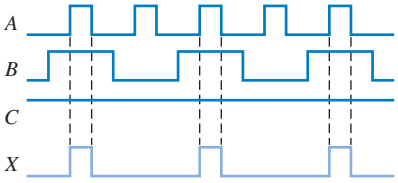


▶ **FIGURA 3–95**

3–4. A forma de onda de saída é a mesma que na entrada *A*.

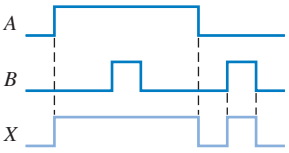
3–5. Veja a Figura 3–96.

3–6. Veja a Figura 3–97.



C = ALTO

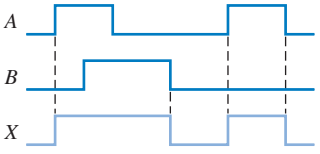
▲ **FIGURA 3–96**



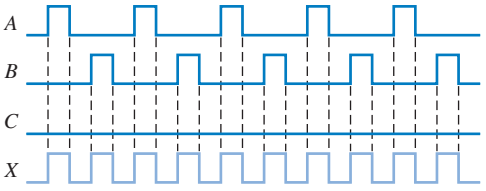
▲ **FIGURA 3–97**

3–7. Veja a Figura 3–98.

3–8. Veja a Figura 3–99.



▲ **FIGURA 3–98**

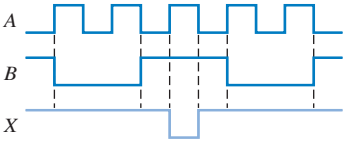


C = BAIXO

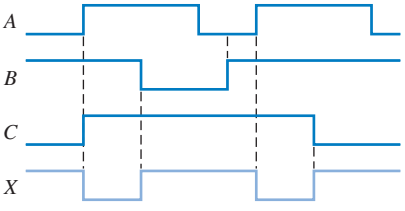
▲ **FIGURA 3–99**

3–9. Veja a Figura 3–100.

3–10. Veja a Figura 3–101.



▲ **FIGURA 3–100**

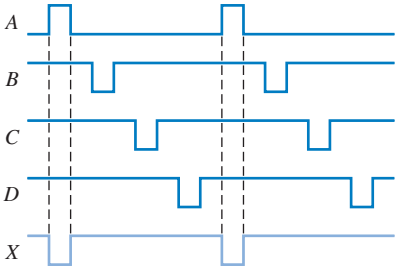


▲ **FIGURA 3–101**

3–11. Use uma porta NAND de 3 entradas.

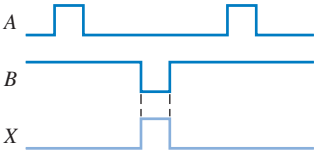
3–12. Use uma porta NAND de 4 entradas operando como uma porta OR negativa.

3–13. Veja a Figura 3–102.

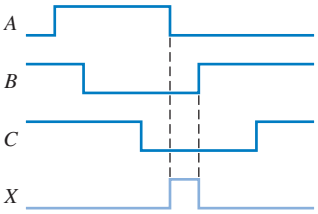


▶ **FIGURA 3–102**

3–14. Veja a Figura 3–103.



▲ **FIGURA 3–103**



▲ **FIGURA 3–104**

3–15. Veja a Figura 3–104.

3–16. Use uma porta NOR de 2 entradas.

3–17. Uma porta NAND de 3 entradas.

3–18. A saída é sempre nível BAIXO. O diagrama de temporização é uma linha reta.

3–19. A porta EX-OR não detecta falhas simultâneas se os dois circuitos produzirem as mesmas saídas.

3–20. As saídas não são afetadas.

3–21. 6 colunas, 9 linhas e 3 portas AND com três entradas cada porta.

3–22. A porta com t_{PLH} e t_{PHL} de 4 ns pode operar com uma freqüência maior.

3–23. 10 mW

3–24. A saída da porta ou o pino 13 (entrada) está aberta internamente.

3–25. O display mostrará uma leitura errada porque o contador continua contando até a inicialização (resete).

3–26. O pulso de habilitação é muito curto ou o contador é resetado cedo demais.

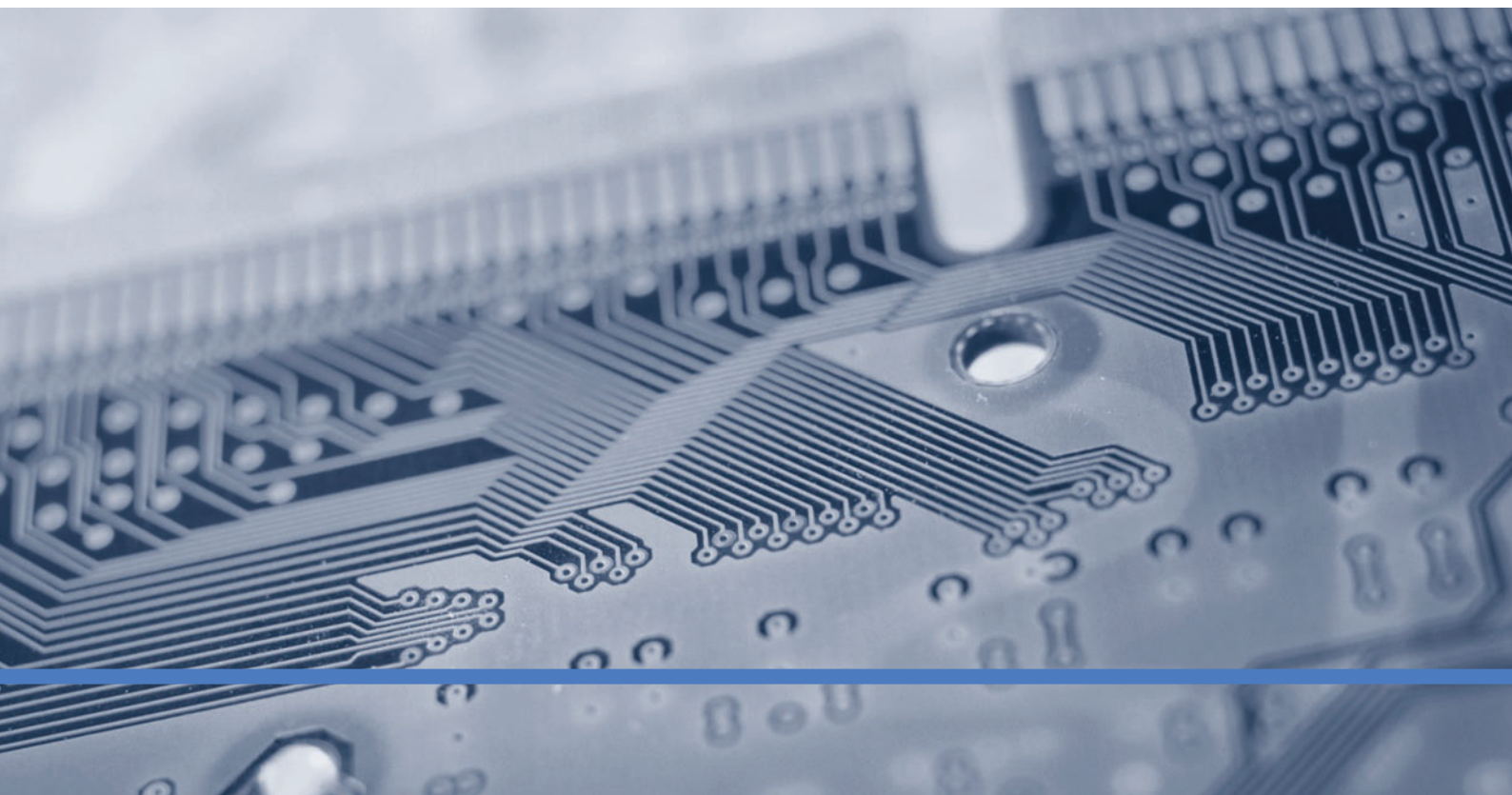
AUTOTESTE

1. (d) **2.** (d) **3.** (a) **4.** (e) **5.** (c) **6.** (a) **7.** (d) **8.** (b) **9.** (d)
10. (b) **11.** (d) **12.** (c) **13.** (b) **14.** (a) **15.** (d) **16.** (c) **17.** (c)

4

Álgebra Booleana e Simplificação Lógica

TÓPICOS DO CAPÍTULO



OBJETIVOS DO CAPÍTULO

- Aplicar as leis e regras básicas da álgebra Booleana
- Aplicar os teoremas de DeMorgan em expressões Booleanas
- Descrever circuitos de portas lógicas com expressões Booleanas
- Calcular expressões Booleanas
- Simplificar expressões usando as leis e regras da álgebra Booleana
- Converter qualquer expressão Booleana numa soma-de-produtos
- Converter qualquer expressão Booleana num produto-de-somas
- Usar um mapa de Karnaugh para simplificar expressões Booleanas
- Usar um mapa de Karnaugh para simplificar funções de tabela-verdade
- Utilizar condições "don't care" para simplificar funções lógicas
- Escrever um programa VHDL para uma lógica simples
- Aplicar a álgebra Booleana, o método do mapa de Karnaugh e VHDL numa aplicação de sistema

TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- Variável
- Complemento
- Termo-soma
- Termo-produto
- soma-de-produtos
- Produto-de-somas
- Mapa de Karnaugh
- Minimização
- "Don't care"
- VHDL

INTRODUÇÃO

Em 1854, Georg Boole publicou um trabalho intitulado "Uma Investigação das leis do Pensamento, sobre as quais são fundadas as teorias Matemáticas de Lógica e Probabilidades" (*Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic AND Probabilities*). Foi nessa publicação que uma "álgebra lógica", conhecida hoje em dia como álgebra Booleana, foi formulada. A álgebra Booleana é uma forma conveniente e sistemática de expressar e analisar a operação de circuitos lógicos. Claude Shannon foi o primeiro a aplicar o trabalho de Boole na análise e projeto de circuitos lógicos. Em 1938, Shannon escreveu uma tese no MIT intitulada *A Symbolic Analysis of Relay and Switching Circuits*.

Este capítulo aborda as leis, regras e teoremas da álgebra Booleana e suas aplicações em circuitos digitais. Estudaremos como definir um dado circuito com uma expressão Booleana e então avaliar a sua operação. Estudaremos também como simplificar circuitos lógicos usando os métodos da álgebra Booleana e mapas de Karnaugh.

A linguagem de descrição de hardware VHDL para a programação de dispositivos lógicos é introduzida.

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico Aplicações em Sistemas Digitais ilustra conceitos ensinados neste capítulo. O circuito lógico de um display de 7 segmentos usado no sistema de contagem e controle de comprimidos a partir do Capítulo 1 é uma boa maneira de ilustrar a aplicação da álgebra Booleana e do mapa de Karnaugh para obter a implementação mais simples possível no projeto de circuitos lógicos. Portanto, nessa aplicação em sistemas digitais, o foco é no decodificador de BCD para 7 segmentos que aciona os dois sistemas de displays mostrados na Figura 1–58.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

4-1 OPERAÇÕES E EXPRESSÕES BOOLEANAS

A álgebra Booleana é a matemática dos sistemas digitais. Um conhecimento básico da álgebra Booleana é indispensável para o estudo e análise de circuitos lógicos. No capítulo anterior, as operações Booleanas em termos de suas relações com as portas NOT, AND, OR, NAND e NOR foram introduzidas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir variável ■ Definir literal ■ Identificar um termo-soma ■ Calcular um termo-soma ■ Identificar um termo-produto ■ Calcular um termo-produto ■ Explicar a adição Booleana ■ Explicar a multiplicação Booleana

NOTA: COMPUTAÇÃO

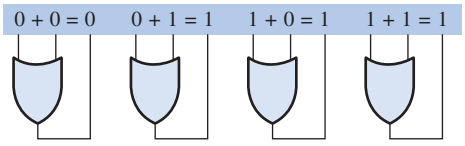
Em um microprocessador, a unidade lógica e aritmética (ALU – *arithmetic logic unit*) realiza operações lógicas Booleanas e aritméticas sobre dados digitais conforme determinado pelas instruções do programa. As operações lógicas são equivalentes às operações das portas básicas as quais estamos familiarizados, porém realizadas com um mínimo de 8 bits de cada vez. AND, OR, NOT e EX-OR são exemplos de instruções lógicas Booleanas, as quais são denominadas de mnemônicos. Um programa em linguagem assembly usa mnemônicos para especificar uma operação. Um outro programa denominado de *assembler* traduz os mnemônicos num código binário que pode ser entendido pelo microprocessador.

A porta OR é um somador Booleano.

Os termos *variável*, *complemento* e *literal* são usados em álgebra Booleana. Uma **variável** é um símbolo (geralmente uma letra maiúscula em itálico) usado para representar uma grandeza lógica. Qualquer variável simples pode ter um valor 1 ou 0. O **complemento** é o inverso de uma variável e é indicado por uma barra sobre a variável. Por exemplo, o complemento da variável A é $\overline{A}$. Se $A = 1$, então $\overline{A} = 0$. Se $A = 0$, então $\overline{A} = 1$. O complemento de uma variável A é lido como "A negado" ou "A barrado". Algumas vezes é usado um outro símbolo, em vez de uma barra, para indicar o complemento de uma variável; por exemplo, B' indica o complemento de B. Neste livro, é usado apenas a barra sobre a variável. Uma **literal** é a variável ou o complemento de uma variável.

Adição Booleana

Lembre-se, do Capítulo 3, de que a **adição Booleana** é equivalente à operação OR e as regras básicas são ilustradas com suas relações com a porta OR da seguinte forma:



Na álgebra Booleana, um **termo-soma** é uma soma de literais. Em circuitos lógicos, um termo-soma é produzido por uma operação OR sem o envolvimento de operações AND. Alguns exemplos de termos-soma são $A + B$, $A + \overline{B}$, $A + B + \overline{C}$ e $\overline{A} + B + C + \overline{D}$.

Um termo-soma será igual a 1 quando uma ou mais das literais no termo for 1. Um termo-soma será igual a 0 somente se cada uma das literais for 0.

EXEMPLO 4–1

Determine os valores de A, B, C e D que tornam o termo-soma $A + \overline{B} + C + \overline{D}$ igual a 0.

Solução Para o termo-soma ser 0, cada uma das literais tem que ser 0. Portanto, $A = 0$ e $B = 1$, de forma que, $\overline{B} = 0$, $C = 0$ e $D = 1$, de forma que $\overline{D} = 0$.

$$A + \overline{B} + C + \overline{D} = 0 + \overline{1} + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

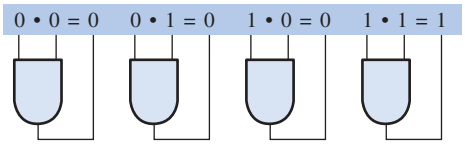
Problema relacionado* Determine os valores de $\overline{A}$ e B que tornam o termo-soma igual a 0.

* As respostas estão no final do capítulo.

Multiplicação Booleana

Lembre-se, também do Capítulo 3, de que a **multiplicação Booleana** é equivalente à operação AND e as regras básicas são ilustradas com as relações com a porta AND a seguir:

A porta AND é um multiplicador Booleano.



Na álgebra Booleana, um **termo-produto** é o produto de literais. Em circuitos lógicos, um termo-produto é produzido por uma operação AND sem o envolvimento de operações OR. Alguns exemplos de termos-produto são AB, $A\overline{B}$, ABC e $A\overline{B}C\overline{D}$.

Um termo-produto é igual a 1 apenas se cada uma das literais no termo for 1. Um termo-produto é igual a 0 quando uma ou mais das literais for 0.

EXEMPLO 4–2

Determine os valores e A, B, C e D que torna o termo-produto $A\overline{B}C\overline{D}$ igual a 1.

Solução Para o termo-produto ser 1, cada uma das literais no termo tem que ser 1. Portanto, $A = 1$, $B = 0$ de forma que $\overline{B} = 1$, $C = 1$ e $D = 0$ de forma que $\overline{D} = 1$.

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Problema relacionado Determine os valores de A e B que tornam o termo-produto $\overline{A}\,\overline{B}$ igual a 1.

SEÇÃO 4–1 REVISÃO

As respostas estão no final do capítulo.

1. Se $A = 0$, qual é o valor de $\overline{A}$?
2. Determine o valor de A, B e C que tornam o termo-soma $\overline{A} + \overline{B} + C$ igual a 0.
3. Determine o valor de A, B e C que tornam o termo-produto $A\overline{B}C$ igual a 1.

4-2 LEIS E REGRAS DA ÁLGEBRA BOOLEANA

Assim como em outras áreas da matemática, existem certas regras bem-desenvolvidas e leis que têm que ser seguidas para aplicar adequadamente a álgebra Booleana. As mais importantes são apresentadas nesta seção.

Ao final do estudo desta seção você deverá ser capaz de:

■ Aplicar as leis comutativas da adição e multiplicação ■ Aplicar as leis associativas da adição e multiplicação ■ Aplicar a lei distributiva ■ Aplicar as doze regras básicas da álgebra Booleana

Leis da Álgebra Booleana

As leis básicas da álgebra Booleana – as **leis comutativas** para a adição e multiplicação, as **leis associativas** para a adição e multiplicação e a **lei distributiva** – são as mesmas que para a álgebra

comum. Cada uma das leis está ilustrada com duas ou três variáveis, porém o número de variáveis não é limitado para essas leis.

Lei Comutativa A *lei comutativa da adição* para duas variáveis é escrita da seguinte forma:

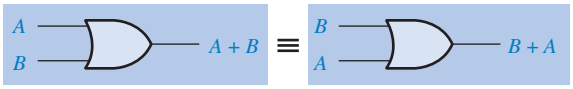
Equação 4–1

$$A + B = B + A$$

Essa lei diz que a ordem das variáveis na qual a função OR é aplicada não faz diferença. Lembre-se que, na álgebra Booleana aplicada a circuitos lógicos, a adição e a operação OR são as mesmas. A Figura 4–1 ilustra a lei comutativa aplicada a uma porta OR e mostra que não importa em qual entrada cada variável é aplicada. (O símbolo ≡ significa "equivalente a").

▶ FIGURA 4–1

Aplicação da lei comutativa da adição.



A *lei comutativa da multiplicação* para duas variáveis é a seguinte:

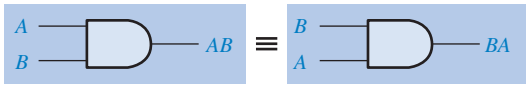
Equação 4–2

$$AB = BA$$

Essa lei diz que a ordem das variáveis na qual a operação AND é aplicada não faz diferença. A Figura 4–2 ilustra essa lei aplicada a uma porta AND.

▶ FIGURA 4–2

Aplicação da lei comutativa da multiplicação.



Lei Associativa A lei associativa da adição escrita para três variáveis é mostrada a seguir:

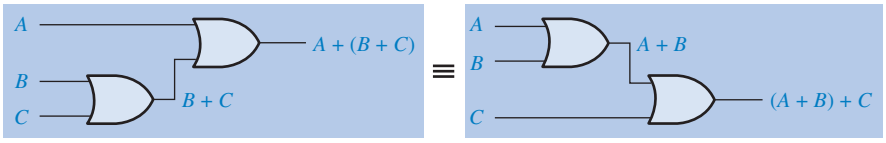
Equação 4–3

$$A + (B + C) = (A + B) + C$$

Essa lei diz que quando é aplicada uma operação OR em mais de duas variáveis, o resultado é o mesmo independente da forma de agrupar as variáveis. A Figura 4–3 ilustra essa lei aplicada em portas OR de 2 entradas.

▶ FIGURA 4–3

Aplicação da lei associativa da adição. Abra o arquivo F04-03 para verificar.



A *lei associativa da multiplicação* escrita para três variáveis é mostrada a seguir:

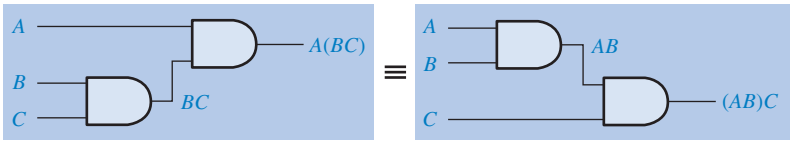
Equação 4–4

$$A(BC) = (AB)C$$

Essa lei diz que a ordem em que as variáveis são agrupadas não faz diferença quando é aplicada uma operação AND em mais de duas variáveis. A Figura 4–4 ilustra essa lei aplicada a portas AND de 2 entradas.

▶ FIGURA 4–4

Aplicação da lei associativa da multiplicação. Abra o arquivo E04-04 para verificar.

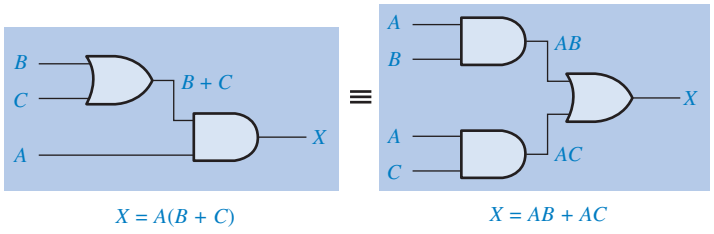


Lei Distributiva A lei distributiva escrita para três variáveis é mostrada a seguir:

Equação 4–5

$$A(B + C) = AB + AC$$

Essa lei diz que a operação AND de uma única variável com o resultado de uma operação OR aplicada em duas ou mais variáveis é equivalente a uma operação OR entre os resultados das operações AND entre uma única variável e cada uma das duas ou mais variáveis. A lei distributiva também expressa o processo de *fatoração* no qual a variável comum A é fatorada em termos-produto, por exemplo, $AB + AC = A(B + C)$. A Figura 4–5 ilustra a lei distributiva em termos de implementação com portas.



◀ **FIGURA 4–5**

Aplicação da lei distributiva. Abra o arquivo F04-05 para verificar.

Regras da Álgebra Booleana

A Tabela 4–1 apresenta uma lista de 12 regras básicas úteis na manipulação e simplificação de expressões Booleanas. As regras de 1 a 9 serão analisadas em termos de suas aplicações em portas lógicas. As regras de 10 a 12 serão obtidas em termos de regras mais simples e das leis discutidas anteriormente.

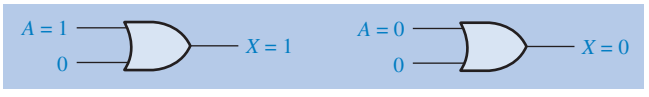
◀ **TABELA 4–1**

Regras básicas da álgebra Booleana

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

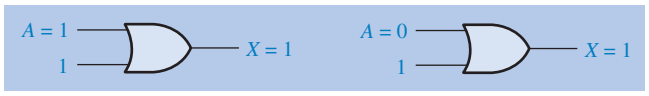
A, B ou C podem representar uma única variável ou uma combinação de variáveis.

Regra 1. $A + 0 = A$ A operação OR de uma variável com 0 é sempre igual a variável. Se a variável de entrada A for 1, a variável X de saída será 1, que é igual a A. Se A for 0, a saída será 0, que também é igual a A. Essa regra é ilustrada na Figura 4–6, na qual a entrada inferior da porta está fixa em 0.



◀ **FIGURA 4–6**

Regra 2. $A + 1 = 1$ A operação OR da variável com 1 é igual a 1. Um 1 numa entrada de uma porta OR produz um 1 na saída, independente do valor da variável na outra entrada. Essa regra é ilustrada na Figura 4–7, na qual a entrada inferior da porta está fixa em 1.



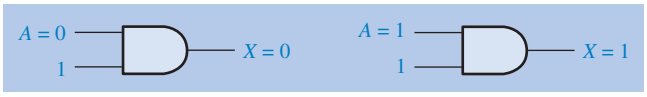
◀ **FIGURA 4–7**

Regra 3. $A \cdot 0 = 0$ A operação AND da variável com 0 sempre é igual a 0. Todas as vezes que uma entrada de uma porta AND for 0, a saída será 0, independente do valor da variável na outra entrada. Essa regra está ilustrada na Figura 4–8, na qual a entrada inferior está fixa em 0.

$X = A \cdot 0 = 0$

▶ FIGURA 4–8

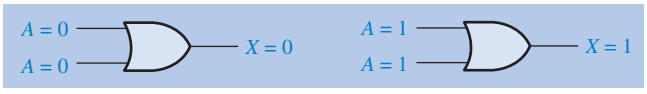
Regra 4. $A \cdot 1 = A$ A operação AND da variável com 1 é sempre igual a variável. Se A for 0 a saída da porta AND será 0. Se A for 1, a saída da porta AND será 1 porque ambas as entradas agora são 1s. Essa regra é mostrada na Figura 4–9, onde a entrada inferior está fixa em 1.



$X = A \cdot 1 = A$

▶ FIGURA 4–9

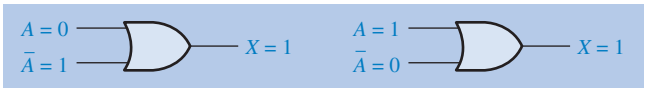
Regra 5. $A + A = A$ A operação OR da variável com ela mesma é sempre igual a variável. Se A for 0, então $0 + 0 = 0$; e se A for 1, então $1 + 1 = 1$. Isso é mostrado na Figura 4–10, onde as duas entradas são a mesma variável.



$X = A + A = A$

▶ FIGURA 4–10

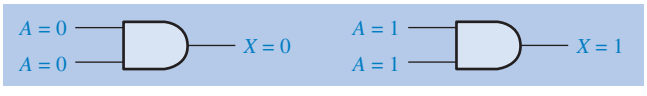
Regra 6. $A + \overline{A} = 1$ A operação OR da variável com o seu complemento é sempre igual a 1. Se A for 0, então $0 + \overline{0} = 0 + 1 = 1$. Se A for 1, então $1 + \overline{1} = 1 + 0 = 1$. Veja a Figura 4–11, onde uma entrada é o complemento da outra.



$X = A + \overline{A} = 1$

▶ FIGURA 4–11

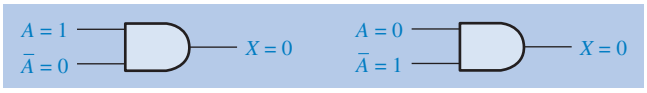
Regra 7. $A \cdot A = A$ A operação AND de uma variável com ela mesma é sempre igual a variável. Se $A = 0$, então $0 \cdot 0 = 0$; e se $A = 1$, então $1 \cdot 1 = 1$. A Figura 4–12 ilustra essa regra.



$X = A \cdot A = A$

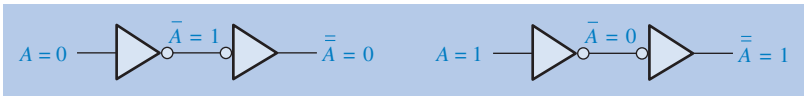
▶ FIGURA 4–12

Regra 8. $A \cdot \overline{A} = 0$ A operação AND de uma variável e o seu complemento é sempre igual a 0. Nesse caso, ou $\overline{A}$ ou sempre será 0; e quando um 0 é aplicado na entrada de uma porta AND, a saída também será 0. A Figura 4–13 ilustra essa regra.



◀ **FIGURA 4–13**

Regra 9. $\overline{\overline{A}} = A$ O complemento duplo de uma variável é sempre igual a variável. Se complementarmos (invertermos) a variável A uma vez, obtemos $\overline{A}$. Então se complementarmos (invertemos) $\overline{A}$, obtemos A, que é a variável original. Essa regra é mostrada na Figura 4–14 usando inversores.



$\overline{\overline{A}} = A$

◀ **FIGURA 4–14**

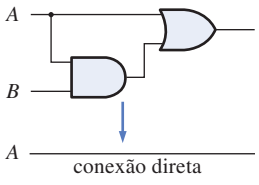
Regra 10. $A + AB = A$ Essa regra pode ser provada aplicando a lei distributiva, Regra 2, e a Regra 4 como a seguir:

$A + B = A(1 + B)$	Fatorando (lei distribuitiva)
$= A \cdot 1$	Regra 2: $(1 + B) = 1$
$= A$	Regra 4: $A \cdot 1 = A$

A prova é mostrada na Tabela 4–2, onde temos a tabela-verdade e a conseqüente simplificação do circuito lógico.

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

igual



◀ **TABELA 4–2**

Regra 10: $A + AB = A$. Abra o arquivo T04-02 para verificar

Regra 11. $A + \overline{A}B = A + B$ Essa regra pode ser provada da seguinte forma:

$A + \overline{A}B = (A + AB) + \overline{A}B$	Regra 10: $A = A + AB$
$= (AA + AB) + \overline{A}B$	Regra 7: $A = AA$
$= AA + AB + A\overline{A} + \overline{A}B$	Regra 8: adicionando $A\overline{A} = 0$
$= (A + \overline{A})(A + B)$	Fatorando
$= 1 \cdot (A + B)$	Regra 6: $A + \overline{A} = 1$
$= A + B$	Regra 4: simplifica o 1

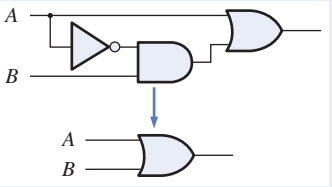
A prova é mostrada na Tabela 4–3, onde temos a tabela-verdade e a conseqüente simplificação do circuito lógico.

▶ TABELA 4–3

Regra 11: $A + \overline{A}B = A + B$. Abra o arquivo T04-03 para verificar

A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

igual



Regra 12. $(A + B)(A + C) = A + BC$ Essa regra pode ser provada da seguinte forma:

$$
\begin{aligned}
(A + B)(A + C) &= AA + AC + AB + BC && \text{Lei distributiva}\\
&= A + AC + AB + BC && \text{Regra 7: } AA = A\\
&= A(1 + C) + AB + BC && \text{Fatorando (lei distributiva)}\\
&= A \cdot 1 + AB + BC && \text{Regra 2: } 1 + C = 1\\
&= A(1 + B) + BC && \text{Fatorando (lei distributiva)}\\
&= A \cdot 1 + BC && \text{Regra 2: } 1 + B = 1\\
&= A + BC && \text{Regra 4: } A \cdot 1 = A
\end{aligned}
$$

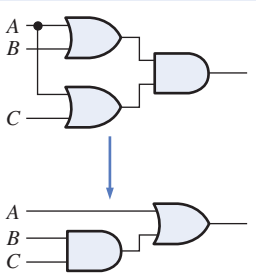
A prova é mostrada na Tabela 4–4, onde temos a tabela-verdade e a conseqüente simplificação do circuito lógico.

▼ TABELA 4–4

Regra 12: $(A + B)(A + C) = A + BC$. Abra o arquivo T04-04 para verificar

A	B	C	$A + B$	$A + C$	$(A + B)(A + C)$	BC	$A + BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

igual



SEÇÃO 4–2 REVISÃO

1. Aplique a lei associativa da adição na expressão $A + (B + C + D)$.
2. Aplique a lei distributiva na expressão $A(B + C + D)$.

4-3 TEOREMAS DE DEMORGAN

DeMorgan, um matemático que conheceu Boole, propôs dois teoremas que representam uma parte importante da álgebra Booleana. Em termos práticos, os teoremas de DeMorgan provêm uma verificação da equivalência entre as portas NAND e OR negativa e a equivalência entre as portas NOR e AND negativa, discutidas no Capítulo 3.

Ao final do estudo desta seção você deverá ser capaz de:

■ Citar os teoremas de DeMorgan ■ Relacionar os teoremas com a equivalência entre as portas NAND e OR negativa e a equivalência entre as portas NOR e AND negativa ■ Aplicar os teoremas de DeMorgan na simplificação de expressões Booleanas

Um dos teoremas de DeMorgan é:

O complemento de um produto de variáveis é igual a soma dos complementos das variáveis.

Dizendo de outra forma,

O complemento de duas ou mais variáveis submetidas a uma operação AND é equivalente a uma operação OR entre os complementos das variáveis individuais.

A fórmula que expressa esse teorema para duas variáveis é:

$$\overline{XY} = \overline{X} + \overline{Y}$$ **Equação 4–6**

O segundo teorema de DeMorgan é expresso da seguinte forma:

O complemento de uma soma de variáveis é igual ao produto do complemento das variáveis.

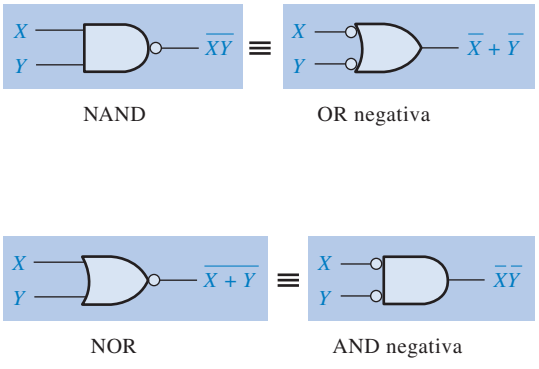
Dizendo de outra forma,

O complemento de duas ou mais variáveis submetidas a uma operação OR é equivalente a uma operação AND entre os complementos das variáveis individuais.

A fórmula para a expressão desse teorema para duas variáveis é:

$$\overline{X + Y} = \overline{X}\,\overline{Y}$$ **Equação 4–7**

A Figura 4–15 mostra as equivalências de portas e as tabelas-verdade para as Equações 4–6 e 4–7.



Entradas		Saída	
X	Y	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Entradas		Saída	
X	Y	$\overline{X+Y}$	$\overline{XY}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

◀ FIGURA 4–15

Equivalências entre portas e as correspondentes tabelas-verdade que ilustram os teoremas de DeMorgan. Observe a igualdade entre as duas colunas de saída em cada tabela. Isso mostra que as portas equivalentes realizam a mesma função lógica.

Conforme já foi dito, os teoremas de DeMorgan também se aplicam a expressões nas quais existem mais que duas variáveis. Os exemplos a seguir ilustram a aplicação dos teoremas de DeMorgan em expressões de 3 e 4 variáveis.

EXEMPLO 4–3

Aplique os teoremas de DeMorgan nas expressões: $\overline{XYZ}$ e $\overline{X + Y + Z}$.

Solução

$$\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado Aplique o teorema de DeMorgan na expressão: $\overline{\overline{X} + \overline{Y} + \overline{Z}}$.

EXEMPLO 4–4

Aplique os teoremas de DeMorgan nas expressões: $\overline{WXYZ}$ e $\overline{W + X + Y + Z}$.

Solução

$$\overline{WXYZ} = \overline{W} + \overline{X} + \overline{Y} + \overline{Z}$$

$$\overline{W + X + Y + Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Problema relacionado Aplique o teorema de DeMorgan na expressão: $\overline{\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}}$.

Cada variável nos teoremas de DeMorgan, conforme dito nas Equações 4–6 e 4–7, também pode representar uma combinação de outras variáveis. Por exemplo, X pode ser igual ao termo $AB + C$ e Y pode ser igual ao termo $A + BC$. Assim, se podemos aplicar o teorema de DeMorgan para duas variáveis conforme foi dito para $\overline{XY} = \overline{X} + \overline{Y}$ na expressão $\overline{(AB + C)(A + BC)}$, obtemos o seguinte resultado:

$$\overline{(AB + C)(A + BC)} = (\overline{AB + C}) + (\overline{A + BC})$$

Observe que no resultado anterior temos dois termos, $\overline{AB + C}$ e $\overline{A + BC}$, para cada um dos quais podemos aplicar novamente o teorema de DeMorgan $\overline{X + Y} = \overline{X}\,\overline{Y}$ individualmente, conforme a seguir:

$$(\overline{AB + C}) + (\overline{A + BC}) = (\overline{AB})\overline{C} + \overline{A}(\overline{BC})$$

Observe que ainda temos dois termos na expressão nos quais o teorema de DeMorgan pode ser aplicado novamente. Esses termos são $\overline{AB}$ e $\overline{BC}$. Uma última aplicação do teorema de DeMorgan resulta no seguinte:

$$(\overline{AB})\overline{C} + \overline{A}(\overline{BC}) = (\overline{A} + \overline{B})\overline{C} + \overline{A}(\overline{B} + \overline{C})$$

Embora esse resultado possa ser simplificado pelo uso das leis e regras Booleanas, os teoremas de DeMorgan não podem mais ser usados.

Aplicando os Teoremas de DeMorgan

Os procedimentos a seguir ilustram a aplicação dos teoremas de DeMorgan e da álgebra Booleana na seguinte expressão:

$$\overline{\overline{A + B\overline{C}} + D(\overline{E + \overline{F}})}$$

Passo 1. Identifique os termos nos quais podemos aplicar os teoremas de DeMorgan e pense que cada termo é uma única variável. Faça $\overline{A + B\overline{C}} = X$ e $D(\overline{E + \overline{F}}) = Y$.

Passo 2. Como $\overline{X + Y} = \overline{X}\,\overline{Y}$,

$$\overline{(A + B\overline{C}) + (D(E + \overline{F}))} = (\overline{A + B\overline{C}})(\overline{D(E + \overline{F})})$$

Passo 3. Use a Regra 9 $(\overline{\overline{A}} = A)$ para cancelar a dupla barra sobre o termo da esquerda (isso não é parte do teorema de DeMorgan).

$$(\overline{A + B\overline{C}})(\overline{D(E + \overline{F})}) = (A + B\overline{C})(\overline{D(E + \overline{F})})$$

Passo 4. Aplique o teorema de DeMorgan ao segundo termo.

$$(A + B\overline{C})(\overline{D(E + \overline{F})}) = (A + B\overline{C})(\overline{D} + (\overline{E + \overline{F}}))$$

Passo 5. Use a Regra 9 $(\overline{\overline{A}} = A)$ para cancelar a dupla barra sobre a parte $E + \overline{F}$ do termo.

$$(A + B\overline{C})(\overline{D} + \overline{E + \overline{F}}) = (A + B\overline{C})(\overline{D} + E + \overline{F})$$

Os próximos três exemplos ilustram ainda mais como usar os teoremas de DeMorgan.

EXEMPLO 4–5

Aplique os teoremas de DeMorgan em cada uma das seguintes expressões:

(a) $\overline{(A + B + C)D}$ **(b)** $\overline{ABC + DEF}$ **(c)** $\overline{A\overline{B} + \overline{C}D + EF}$

Solução **(a)** Faça $A + B + C = X$ e $D = Y$. A expressão $\overline{(A + B + C)D}$ é da forma $\overline{XY} = \overline{X} + \overline{Y}$ e pode ser reescrita como:

$$\overline{(A + B + C)D} = \overline{A + B + C} + \overline{D}$$

Em seguida aplique o teorema de DeMorgan no termo $\overline{A + B + C}$.

(b) Faça $ABC = X$ e $DEF = Y$. A expressão $\overline{ABC + DEF}$ está na forma $\overline{X + Y} = \overline{X}\,\overline{Y}$ e pode ser reescrita da seguinte forma:

$$\overline{ABC + DEF} = (\overline{ABC})(\overline{DEF})$$

Em seguida, aplique o teorema de DeMorgan em cada um dos seguintes termos: $\overline{ABC}$ e $\overline{DEF}$.

$$(\overline{ABC})(\overline{DEF}) = (\overline{A} + \overline{B} + \overline{C})(\overline{D} + \overline{E} + \overline{F})$$

(c) Faça $A\overline{B} = X$, $\overline{C}D = Y$ e $EF = Z$. A expressão $\overline{A\overline{B} + \overline{C}D + EF}$ está na forma $\overline{X + Y + Z} = \overline{X}\,\overline{Y}\,\overline{Z}$ e pode ser reescrita como:

$$\overline{A\overline{B} + \overline{C}D + EF} = (\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF})$$

Em seguida, aplique o teorema de DeMorgan em cada um dos seguintes termos: $\overline{A\overline{B}}$, $\overline{\overline{C}D}$ e $\overline{EF}$.

$$(\overline{A\overline{B}})(\overline{\overline{C}D})(\overline{EF}) = (\overline{A} + B)(C + \overline{D})(\overline{E} + \overline{F})$$

Problema relacionado Aplique os teoremas de DeMorgan na expressão: $\overline{\overline{ABC} + D + E}$.

EXEMPLO 4–6

Aplique os teoremas de DeMorgan em cada uma das expressões a seguir:

(a) $\overline{(\overline{A + B}) + \overline{C}}$ **(b)** $\overline{(\overline{A} + B) + CD}$ **(c)** $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}}$

Solução

(a) $\overline{(\overline{A + B}) + \overline{C}} = (\overline{\overline{A + B}})\overline{\overline{C}} = (A + B)C$

(b) $\overline{(\overline{A} + B) + CD} = (\overline{\overline{A} + B})\overline{CD} = (\overline{\overline{A}}\,\overline{B})(\overline{C} + \overline{D}) = A\overline{B}(\overline{C} + \overline{D})$

(c) $\overline{(A + B)\overline{C}\,\overline{D} + E + \overline{F}} = \overline{((A + B)\overline{C}\,\overline{D})(\overline{E + \overline{F}})} = (\overline{A}\,\overline{B} + C + D)\overline{E}F$

Problema relacionado Aplique os teoremas na expressão $\overline{\overline{A}B(C + \overline{D}) + E}$.

EXEMPLO 4–7

A expressão Booleana para uma porta EX-OR é $A\overline{B} + \overline{A}B$. Tendo essa expressão como ponto de partida, use o teorema de DeMorgan e quaisquer outras regras ou leis aplicáveis para desenvolver uma expressão para a porta EX-NOR.

Solução Comece pela complementação da expressão para a EX-OR, aplicando em seguida o teorema de DeMorgan como mostrado a seguir:

$$\overline{A\overline{B} + \overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A} + \overline{\overline{B}})(\overline{\overline{A}} + \overline{B}) = (\overline{A} + B)(A + \overline{B})$$

Em seguida, aplique a lei distributiva e a Regra 8 ($A \cdot \overline{A} = 0$).

$$(\overline{A} + B)(A + \overline{B}) = \overline{A}A + \overline{A}\,\overline{B} + AB + B\overline{B} = \overline{A}\,\overline{B} + AB$$

A expressão final para a EX-NOR é $\overline{A}\,\overline{B} + AB$. Observe que essa expressão é igual a 1 todas as vezes que as duas variáveis forem 0 ou 1.

Problema relacionado Começando pela expressão da porta NAND de 4 entradas, use os teoremas de DeMorgan para desenvolver uma expressão para uma porta OR negativa de 4 entradas.

SEÇÃO 4–3 REVISÃO

1. Aplique os teoremas e DeMorgan às seguintes expressões:

(a) $\overline{ABC} + (\overline{D + E})$ **(b)** $\overline{(A + B)C}$ **(c)** $\overline{A + B + C} + \overline{\overline{D}E}$

4-4 ANÁLISE BOOLEANA DE CIRCUITOS LÓGICOS

A álgebra Booleana provê uma forma concisa de expressar a operação de um circuito lógico constituído de uma combinação de portas lógicas de forma que a saída possa ser determinada por várias combinações de valores de entrada.

Ao final do estudo desta seção você deverá ser capaz de:

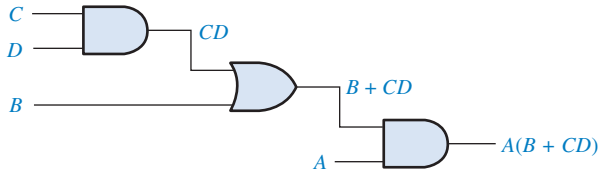
■ Determinar a expressão Booleana para uma combinação de portas ■ Avaliar a operação lógica de um circuito a partir da expressão Booleana ■ Construir uma tabela-verdade

Expressão Booleana para um Circuito Lógico

Um circuito lógico pode ser descrito por uma equação Booleana.

Para obter a expressão Booleana para um dado circuito lógico, comece pelas entradas mais à esquerda e, percorrendo o circuito até a saída final, escreva a expressão para cada porta lógica. Para o exemplo de circuito mostrado na Figura 4–16, a expressão Booleana é determinada da seguinte forma:

1. A expressão para a porta AND mais à esquerda com entradas C e D é CD.
2. A saída da porta AND mais à esquerda é uma das entradas da porta OR e B é a outra entrada. Portanto, a expressão para a porta OR é $B + CD$.
3. A saída da porta OR é uma das entradas da porta AND mais à direita e A é a outra entrada. Portanto, a expressão para essa porta AND é $A(B + CD)$, que é a expressão final de saída para o circuito completo.



◀ **FIGURA 4–16**
Um circuito lógico mostrando o desenvolvimento da expressão Booleana de saída.

Construindo uma Tabela-verdade para um Circuito Lógico

Um circuito lógico pode ser descrito por uma tabela-verdade.

Uma vez que a expressão Booleana para um dado circuito lógico foi determinada, uma tabela-verdade que mostra a saída para todos os valores possíveis das variáveis de entrada pode ser desenvolvida. O procedimento requer que avaliemos a expressão Booleana para todas as combinações possíveis dos valores das variáveis de entrada. No caso do circuito visto na Figura 4–16, existem quatro variáveis de entrada (A, B, C e D) sendo possível portanto dezesseis ($2^4 = 16$) combinações de valores.

Calculando a Expressão Para o cálculo da expressão $A(B + CD)$, determine primeiro os valores das variáveis que tornam a expressão igual a 1, usando as regras para a adição e multiplicação Booleanas. Nesse caso, a expressão é igual a 1 apenas se $A = 1$ e $B + CD = 1$ porque

$$A(B + CD) = 1 \cdot 1 = 1$$

Agora determine quando o termo $B + CD$ é igual a 1. O termo $B + CD = 1$ se $B = 1$ ou $CD = 1$ ou se B e CD forem ambos iguais a 1 porque

$$B + CD = 1 + 0 = 1$$
$$B + CD = 0 + 1 = 1$$
$$B + CD = 1 + 1 = 1$$

O termo $CD = 1$ apenas se $C = 1$ e $D = 1$.

Resumindo, a expressão $A\ (B + CD) = 1$ quando $A = 1$ e $B = 1$ independente dos valores de C e D ou quando $A = 1$, $C = 1$ e $D = 1$ independente do valor de B. A expressão $A\ (B + CD) = 0$ para todas as outras combinações de valores das variáveis.

Colocando os Resultados no Formato de Tabela-Verdade O primeiro passo é fazer uma lista das dezesseis combinações de 1s e 0s das variáveis de entrada numa seqüência binária conforme mostra a Tabela 4–5. Em seguida, coloque um 1 na coluna de saída para cada combinação das variáveis de entrada que foi determinada na avaliação. Finalmente, coloque um 0 na coluna de saída para todas as outras combinações das variáveis de entrada. Esses resultados são mostrados na tabela-verdade na Tabela 4–5.

TABELA 4–5

Tabela-verdade para o circuito lógico mostrado na Figura 4–16

ENTRADAS				SAÍDA
A	*B*	*C*	*D*	*A(B + CD)*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

SEÇÃO 4–4 REVISÃO

1. Substitua as portas AND por portas OR e a porta OR por porta AND no circuito visto na Figura 4–16 e determine a expressão Booleana para a saída.
2. Construa uma tabela-verdade para o circuito da Questão 1.

4-5 SIMPLIFICAÇÃO USANDO A ÁLGEBRA BOOLEANA

Ao aplicarmos a álgebra Booleana, muitas vezes temos que reduzir uma determinada expressão para a sua forma mais simples ou transformá-la em um formato mais conveniente a fim de implementar a expressão mais eficientemente. A abordagem feita nesta seção usa as leis básicas, regras e teoremas da álgebra Booleana para manipular e simplificar uma expressão. Esse método depende do conhecimento completo da álgebra Booleana e uma considerável prática na sua aplicação, além de habilidade e inteligência.

Ao final do estudo desta seção você deverá ser capaz de:

- Aplicar as leis, regras e teoremas da álgebra Booleana para simplificar expressões em geral

Uma expressão Booleana simplificada usa a menor quantidade de portas possível para implementar uma dada expressão. Os Exemplos 4–8 a 4–11 ilustram a simplificação Booleana.

EXEMPLO 4–8

Usando técnicas da álgebra Booleana, simplifique a expressão

$$AB + A(B + C) + B(B + C)$$

Solução Os passos apresentados a seguir não representam necessariamente a única abordagem.

Passo 1 Aplique a lei distributiva ao segundo e terceiro termos da expressão, conforme mostrado a seguir:

$$AB + AB + AC + BB + BC$$

Passo 2 Aplique a Regra 7 ($BB = B$) no quarto termo.

$$AB + AB + AC + B + BC$$

Passo 3 Aplique a Regra 5 ($AB + AB = AB$) nos primeiros dois termos.

$$AB + AC + B + BC$$

Passo 4 Aplique a Regra 10 ($B + BC = B$) nos últimos dois termos.

$$AB + AC + B$$

Passo 5 Aplique a Regra 10 ($AB + B = B$) ao primeiro e terceiro termos.

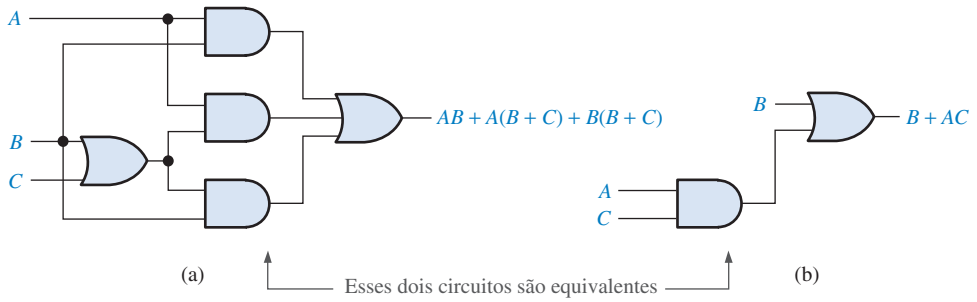
$$B + AC$$

Nesse ponto a expressão está com a máxima simplificação possível. Uma vez adquirido prática na aplicação da álgebra Booleana, podemos combinar diversos passos individuais.

Problema relacionado Simplifique a expressão Booleana $A\overline{B} + A(\overline{B + C}) + B(\overline{B + C})$.

Simplificação significa menos portas lógicas para a mesma função.

A Figura 4–17 mostra que o processo de simplificação dado no Exemplo 4–8 reduziu significativamente o número de portas lógicas necessárias para implementar a expressão. A parte (a) dessa figura mostra que cinco portas são necessárias para implementar a expressão na sua forma original; entretanto, apenas duas portas são necessárias para a expressão simplificada, mostrada na parte (b). É importante perceber que esses dois circuitos de portas são equivalentes. Ou seja, para qualquer combinação de níveis nas entradas A, B e C temos a mesma saída para qualquer um dos dois circuitos.



▲ **FIGURA 4–17**

Circuitos de portas para o Exemplo 4–8. Abra o arquivo F04-17 para verificar a equivalência.

EXEMPLO 4–9

Simplifique a seguinte expressão Booleana:

$$[A\overline{B}(C + BD) + \overline{A}\,\overline{B}]C$$

Observe que os colchetes e parênteses significam o mesmo: o termo interno é multiplicado (operação AND) com os termos externos.

Solução

Passo 1 Aplique a lei distributiva aos termos dentro dos colchetes.

$$(A\overline{B}C + A\overline{B}BD + \overline{A}\,\overline{B})C$$

Passo 2 Aplique a Regra 8 $(\overline{B}B = 0)$ ao segundo termo dentro dos parênteses.

$$(A\overline{B}C + A \cdot 0 \cdot D + \overline{A}\,\overline{B})C$$

Passo 3 Aplique a Regra 3 $(A \cdot 0 \cdot D = 0)$ ao segundo termo dentro dos parênteses.

$$(A\overline{B}C + 0 + \overline{A}\,\overline{B})C$$

Passo 4 Aplique a Regra 1 (simplifique o 0) dentro dos parênteses.

$$(A\overline{B}C + \overline{A}\,\overline{B})C$$

Passo 5 Aplique a lei distributiva.

$$A\overline{B}CC + \overline{A}\,\overline{B}C$$

Passo 6 Aplique a Regra 7 $(CC = C)$ ao primeiro termo.

$$A\overline{B}C + \overline{A}\,\overline{B}C$$

Passo 7 Fatore $\overline{B}C$.

$$\overline{B}C(A + \overline{A})$$

Passo 8 Aplique a Regra 6 $(A + \overline{A} = 1)$.

$$\overline{B}C \cdot 1$$

Passo 9 Aplique a Regra 4 (simplifique o 1).

$$\overline{B}C$$

Problema relacionado Simplifique a seguinte expressão Booleana $[AB(C + \overline{BD}) + \overline{AB}]CD$.

EXEMPLO 4–10

Simplifique a seguinte expressão Booleana:

$$\overline{A}BC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

Solução

Passo 1 Fatore BC com o primeiro e o último termos.

$$BC(\overline{A} + A) + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C$$

Passo 2 Aplique a Regra 6 $(\overline{A} + A + 1)$ ao termo entre parênteses e fatore $\overline{AB}$ a partir do segundo e último termos.

$$BC \cdot 1 + A\overline{B}(\overline{C} + C) + \overline{A}\,\overline{B}\,\overline{C}$$

Passo 3 Aplique a Regra 4 (simplifique o 1) ao primeiro termo e a Regra 6 $(\overline{C} + C = 1)$ ao termo entre parênteses.

$$BC + A\overline{B} \cdot 1 + \overline{A}\,\overline{B}\,\overline{C}$$

Passo 4 Aplique a Regra 4 (simplifique o 1) no segundo termo.

$$BC + A\overline{B} + \overline{A}\,\overline{B}\,\overline{C}$$

Passo 5 Fatore $\overline{B}$ a partir do segundo e terceiro termos.

$$BC + \overline{B}(A + \overline{A}\,\overline{C})$$

Passo 6 Aplique a Regra 11 $(A + \overline{A}\,\overline{C} = A + \overline{C})$ no termo entre parênteses.

$$BC + \overline{B}(A + \overline{C})$$

Passo 7 Use as leis distributiva e comutativa para obter a seguinte expressão:

$$BC + A\overline{B} + \overline{B}\,\overline{C}$$

Problema relacionado Simplifique a seguinte expressão Booleana: $AB\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C}$.

EXEMPLO 4–11

Simplifique a seguinte expressão Booleana:

$$\overline{AB + AC} + \overline{A}\,\overline{B}C$$

Solução **Passo 1** Aplique o teorema de DeMorgan no primeiro termo.

$$(\overline{AB})(\overline{AC}) + \overline{A}\,\overline{B}C$$

Passo 2 Aplique o teorema de DeMorgan em cada termo entre parênteses.

$$(\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A}\,\overline{B}C$$

Passo 3 Aplique a lei distributiva ao dois termos entre parênteses.

$$\overline{A}\,\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{B}C$$

Passo 4 Aplique a Regra 7 $(\overline{A}\,\overline{A} = \overline{A})$ no primeiro termo, e aplique a Regra 10 $[\overline{A}\,\overline{B} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}(1 + C) = \overline{A}\,\overline{B}]$ no terceiro e último termos.

$$\overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

Passo 5 Aplique a Regra 10 $[\overline{A} + \overline{A}\,\overline{C} = \overline{A}(1 + \overline{C}) = \overline{A}]$ no primeiro e segundo termos.

$$\overline{A} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$

Passo 6 Aplique a Regra 10 $[\overline{A} + \overline{A}\,\overline{B} = \overline{A}(1 + \overline{B}) = \overline{A}]$ no primeiro e segundo termos.

$$\overline{A} + \overline{B}\,\overline{C}$$

Problema relacionado Simplifique a seguinte expressão Booleana: $\overline{AB} + \overline{AC} + \overline{A}\,\overline{B}\,\overline{C}$.

SEÇÃO 4–5 REVISÃO

1. Simplifique, quando possível, as seguintes expressões Booleanas:

(a) $A + AB + A\overline{B}C$ **(b)** $(\overline{A} + B)C + ABC$ **(c)** $A\overline{B}C(BD + CDE) + A\overline{C}$

2. Implemente cada expressão originalmente apresentada na Questão 1 usando as portas lógicas apropriadas. Em seguida, implemente a expressão simplificada e compare o número de portas.

4-6 FORMAS PADRONIZADAS DE EXPRESSÕES BOOLEANAS

Todas as expressões Booleanas, independente das suas formas, podem ser convertidas em qualquer uma das duas formas padrão: a forma de soma-de-produtos e a forma de produto-de-somas. A padronização faz a avaliação, simplificação e implementação de expressões Booleanas de forma muito mais sistemática e fácil.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar uma expressão de soma-de-produtos ■ Determinar o domínio de uma expressão Booleana ■ Converter qualquer expressão de soma-de-produtos para uma forma padrão ■ Avaliar uma expressão padrão de soma-de-produtos em termos dos valores binários ■ Identificar uma expressão de produto-de-somas ■ Converter qualquer expressão de produto-de-somas para uma forma padrão ■ Avaliar uma expressão padrão de produto-de-somas em termos dos valores binários ■ Converter expressões de um padrão para outro

A Forma de soma-de-produtos

Uma expressão de soma-de-produtos pode ser implementada com uma OR e duas ou mais ANDs.

Um termo-produto foi definido na Seção 4–1 como um termo que consiste em produto (multiplicação Booleana) de literais (variáveis ou seus complementos). Quando dois ou mais termos-produto são somados por uma adição Booleana, a expressão resultante é uma **soma-de-produtos**. Alguns exemplos são mostrados a seguir:

$$AB + ABC$$
$$ABC + CDE + \overline{B}C\overline{D}$$
$$\overline{A}B + \overline{A}B\overline{C} + AC$$

Uma expressão na forma de soma-de-produtos também pode conter um termo de uma única variável, como em $A + \overline{A}\,\overline{B}C + BC\overline{D}$. Consulte os exemplos de simplificação na última seção, e o leitor verá que cada uma das expressões finais é um termo de produto único ou uma soma-de-produtos. Numa expressão na forma de soma-de-produtos, uma barra não se estende por mais que uma variável; entretanto, mais de uma variável num termo pode ter uma barra sobre ela. Por exemplo, uma expressão na forma de soma-de-produtos pode ter um termo $\overline{A}\,\overline{B}\,\overline{C}$ porém não um termo $\overline{ABC}$.

Domínio de uma Expressão Booleana O **domínio** de uma expressão Booleana geral é o conjunto das variáveis contidas na expressão na forma complementada ou não complementada. Por exemplo, o domínio da expressão $\overline{A}B + A\overline{B}C$ é o conjunto das variáveis A, B, C, e o domínio da expressão $AB\overline{C} + C\overline{D}E + \overline{B}C\overline{D}$ é o conjunto das variáveis A, B, C, D e E.

Implementação AND/OR de uma Expressão de soma-de-produtos A implementação de uma expressão de soma-de-produtos requer simplesmente uma operação OR das saídas de duas ou mais portas AND. Um termo-produto é produzido por uma operação AND e a soma (adição) de dois ou mais termos-produto é produzida por uma operação OR. Portanto, uma expressão de soma-de-produtos pode ser implementada por uma lógica AND/OR na qual as saídas (em quantidade igual ao número de termos-produto na expressão) de portas AND são conectadas às entradas de uma porta OR, conforme mostra a Figura 4–18 para a expressão $AB + BCD + AC$. A saída X da porta OR é igual a expressão de soma-de-produtos.

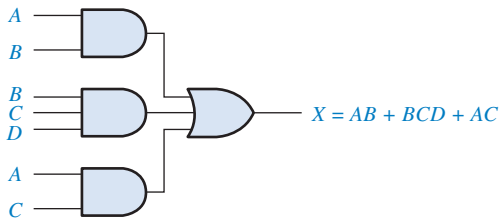


FIGURA 4–18
Implementação da expressão de soma-de-produtos $AB + BCD + AC$.

Implementação NAND/NAND de uma Expressão de soma-de-produtos As portas NAND podem ser usadas na implementação de uma expressão de soma-de-produtos. Usando apenas portas NAND, uma função AND/OR pode ser realizada, conforme ilustra a Figura 4–19. O primeiro nível de portas NAND alimenta a porta NAND que funciona como uma porta OR negativa. As inversões da porta NAND e da OR negativa se cancelam e o resultado é efetivamente um circuito AND/OR.

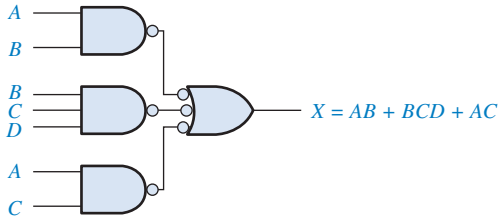


FIGURA 4–19
Essa implementação NAND/NAND é equivalente a AND/OR da Figura 4–18.

Conversão de uma Expressão Geral para a Forma de soma-de-produtos

Qualquer expressão lógica pode ser mudada para o formato de soma-de-produtos aplicando técnicas da álgebra Booleana. Por exemplo, a expressão $A(B + CD)$ pode ser convertida para o formato de soma-de-produtos aplicando a lei distributiva:

$$A(B + CD) = AB + ACD$$

EXEMPLO 4–12

Converta cada uma das seguintes expressões Booleanas para o formato de soma-de-produtos:
(a) $AB + B(CD + EF)$ **(b)** $(A + B)(B + C + D)$ **(c)** $\overline{(\overline{A + B})} + C$

Solução

(a) $AB + B(CD + EF) = AB + BCD + BEF$

(b) $(A + B)(B + C + D) = AB + AC + AD + BB + BC + BD$

(c) $\overline{(\overline{A + B})} + C = (\overline{\overline{A + B}})\overline{C} = (A + B)\overline{C} = A\overline{C} + B\overline{C}$

Problema relacionado Converta $\overline{A}B\overline{C} + (A + \overline{B})(B + \overline{C} + A\overline{B})$ para a forma de soma-de-produtos.

A Forma Padrão de soma-de-produtos

Até agora, lidamos com expressões de soma-de-produtos nas quais alguns dos termos-produto não continham todas as variáveis no domínio da expressão. Por exemplo, a expressão $\overline{A}B\overline{C} + A\overline{B}D + \overline{A}B\overline{C}D$ tem um domínio constituído pelas variáveis A, B, C e D. Entretanto, observe que o conjunto completo de variáveis do domínio não é representado nos primeiros dois termos da expressão; ou seja, D ou $\overline{D}$ não aparece no primeiro termo e C ou $\overline{C}$ não aparece no segundo termo.

Uma *expressão* de soma-de-produtos *padrão* é uma expressão na qual *todas* as variáveis do domínio aparecem em cada um dos termos-produto na expressão. Por exemplo, $A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + AB\overline{C}\,\overline{D}$ é uma expressão de soma-de-produtos padrão. Expressões de soma-de-produtos padrão são importantes na construção de tabelas-verdade, abordado na Seção 4–7, e no uso do método de simplificação com mapa de Karnaugh, o qual é abordado na Seção 4–8. Qualquer expressão de soma-de-produtos não padrão (mencionada simplesmente como soma-de-produtos) pode ser convertida na forma padrão usando a álgebra Booleana.

Conversão de Termos-Produto para Soma-de-Produtos Padrão Cada termo-produto numa expressão de soma-de-produtos que não contém todas as variáveis do domínio pode ser expandida para a forma padrão de modo a incluir todas as variáveis do domínio e seus complementos. Conforme expresso nos passos a seguir, uma expressa de soma-de-produtos não padrão é convertida na forma padrão usando a regra 6 $(A + \overline{A} = 1)$ da álgebra Booleana a partir da Tabela 4–1: uma variável somada ao seu complemento é igual a 1.

Passo 1. Multiplique cada termo-produto não padrão por um termo constituído de uma soma de uma variável que não aparece no termo com o seu complemento. O resultado é dois termos-produto. Conforme sabemos, podemos multiplicar qualquer coisa por 1 sem alterar o seu valor.

Passo 2. Repita o passo 1 até que todos os termos-produto resultantes contenham todas as variáveis do domínio na forma complementada ou não-complementada. Na conversão de um termo-produto para a forma padrão, o número de termos-produto é duplicado para cada variável que não aparece, conforme mostra o Exemplo 4–13.

EXEMPLO 4–13

Converta a seguinte expressão Booleana para a forma de soma-de-produtos padrão:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D$$

Solução O domínio dessa expressão de soma-de-produtos é A, B, C, D. Trabalhe com um termo de cada vez. O primeiro termo, $A\overline{B}C$, não tem a variável D ou $\overline{D}$, então multiplique o primeiro termo por $D + \overline{D}$, conforme mostrado a seguir:

$$A\overline{B}C = A\overline{B}C(D + \overline{D}) = A\overline{B}CD + A\overline{B}C\overline{D}$$

Nesse caso, dois termos-produto padrão aparecem como resultado.

No segundo termo, $\overline{A}\,\overline{B}$, não aparece a variável C ou $\overline{C}$ e D ou $\overline{D}$. Assim, multiplique primeiro o segundo termo por $C + \overline{C}$, conforme mostrado a seguir:

$$\overline{A}\,\overline{B} = \overline{A}\,\overline{B}(C + \overline{C}) = \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

Nos dois termos resultantes não aparece a variável D ou $\overline{D}$, assim, multiplique os dois termos por $D + \overline{D}$, conforme mostrado a seguir:

$$\overline{A}\,\overline{B} = \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} = \overline{A}\,\overline{B}C(D + \overline{D}) + \overline{A}\,\overline{B}\,\overline{C}(D + \overline{D})$$
$$= \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Nesse caso, o resultado são quatro termos-produto padrão.

O terceiro termo, $AB\overline{C}D$, já está na forma padrão. O formato completo padrão da soma dos produtos da expressão original é:

$$A\overline{B}C + \overline{A}\,\overline{B} + AB\overline{C}D = A\overline{B}CD + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}D$$

Problema relacionado Converta $W\overline{X}Y + \overline{X}Y\overline{Z} + WX\overline{Y}$ para a forma padrão de soma-de-produtos.

Representação Binária de um Termo-produto Padrão Um termo-produto padrão é igual a 1 para apenas uma combinação de valores das variáveis. Por exemplo, o termo-produto $A\overline{B}C\overline{D}$ é igual a 1 quando $A = 1$, $B = 0$, $C = 1$ e $D = 0$, conforme mostrado abaixo, e é 0 para todas as outras combinações de valores das variáveis.

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Nesse caso, o termo-produto tem um valor binário de 1010 (decimal dez).

Lembre-se, o termo-produto é implementado com uma porta AND cuja saída é 1 apenas se cada uma de suas entradas for 1. Inversores são usados para produzir os complementos das variáveis conforme necessário.

Uma expressão na forma de soma-de-produtos é igual a 1 apenas se um ou mais dos termos-produto na expressão for igual a 1.

EXEMPLO 4–14

Determine os valores binários para os quais a expressão de soma-de-produtos padrão a seguir é igual a 1:

$$ABCD + A\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Solução O termo $ABCD$ é igual a 1 quando $A = 1$, $B = 1$, $C = 1$ e $D = 1$.

$$ABCD = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

O termo $A\overline{B}\,\overline{C}D$ é igual a 1 quando $A = 1$, $B = 0$, $C = 0$ e $D = 1$.

$$A\overline{B}\,\overline{C}D = 1 \cdot \overline{0} \cdot \overline{0} \cdot 1 = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

O termo $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ é igual a 1 quando $A = 0$, $B = 0$, $C = 0$ e $D = 0$.

$$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = \overline{0} \cdot \overline{0} \cdot \overline{0} \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

A expressão de soma-de-produtos é igual a 1 quando qualquer um ou os três termos-produto for 1.

Problema relacionado Determine os valores binários para os quais a seguinte expressão de soma-de-produtos é igual a 1:

$$\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + \overline{X}\overline{Y}\overline{Z} + XYZ$$

Essa expressão é uma soma-de-produtos padrão?

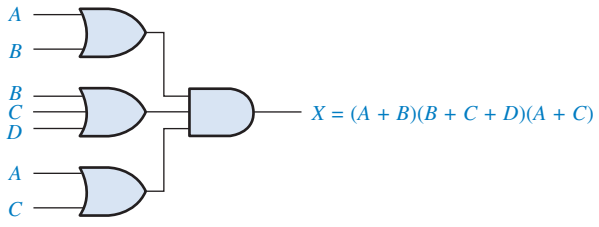
A Forma de Produto-de-Somas

Um termo-soma foi definido na Seção 4–1 como um termo que consiste de uma soma (adição Booleana) de literais (as variáveis ou seus complementos). Quando dois ou mais termos-soma são multiplicados, a expressão resultante é um **produto-de-somas**. Alguns exemplos são mostrados a seguir:

$$(\overline{A} + B)(A + \overline{B} + C)$$
$$(\overline{A} + \overline{B} + \overline{C})(C + \overline{D} + E)(\overline{B} + C + D)$$
$$(A + B)(A + \overline{B} + C)(\overline{A} + C)$$

Uma expressão de produto-de-somas pode conter um termo com uma única variável, conforme ocorre em $\overline{A}(A + \overline{B} + C)(\overline{B} + \overline{C} + D)$. Numa expressão de produto-de-somas, uma única barra sobreposta não pode se estender por mais que uma variável; entretanto, mais que uma variável no termo pode conter uma barra sobreposta. Por exemplo, uma expressão de produto-de-somas pode ter um termo $\overline{A} + \overline{B} + \overline{C}$ mas não um termo $\overline{A + B + C}$.

Implementação de uma Expressão de Produto-de-Somas A implementação de uma expressão de produto-de-somas requer simplesmente uma operação AND entre as saídas de duas ou mais portas OR. Um termo-soma é produzido por uma operação OR, sendo que o produto de dois ou mais termos-soma é produzido por uma operação AND. Portanto, uma expressão de produto-de-somas pode ser implementada por uma lógica na qual as saídas (numa quantidade igual ao número de termos-soma na expressão) das portas OR são conectadas às entradas de uma porta AND, conforme mostra a Figura 4–20 para a expressão $(A + B)(B + C + D)(A + C)$. A saída X da porta AND é igual a expressão de produto-de-somas.



◀ **FIGURA 4–20**
Implementação da expressão de produto-de-somas $(A + B)(B + C + D)(A + C)$.

A Forma Padrão de Produto-de-Somas

Até agora, temos lidado com expressões nas quais alguns dos termos-soma não continham todas as variáveis do domínio da expressão. Por exemplo, a expressão

$$(A + \overline{B} + C)(A + B + \overline{D})(A + \overline{B} + \overline{C} + D)$$

tem um domínio constituído pelas variáveis A, B, C e D. Observe que o conjunto completo das variáveis no domínio não é representada nos dois primeiros termos da expressão; ou seja, D ou $\overline{D}$ não aparece no primeiro termo e C ou $\overline{C}$ não aparece no segundo termo.

Uma expressão de produto-de-somas padrão é uma expressão na qual *todas* as variáveis do domínio aparecem em cada termo-soma na expressão. Por exemplo,

$$(\overline{A} + \overline{B} + \overline{C} + \overline{D})(A + \overline{B} + C + D)(A + B + \overline{C} + D)$$

é uma expressão de produto-de-somas. Qualquer expressão de produto-de-somas não padrão (mencionada simplesmente de produto-de-somas) pode ser convertida na forma padrão usando a álgebra Booleana.

Conversão de um Termo-Soma para um Produto-de-Somas Padrão Cada termo-soma numa expressão de produto-de-somas que não contém todas as variáveis do domínio pode ser expandido para a forma padrão de modo a incluir todas as variáveis do domínio e os seus complementos. Conforme expresso nos passos a seguir, uma expressão de produto-de-somas não padrão é convertida para a forma padrão usando a Regra 8 $(A \cdot \overline{A} = 0)$ da álgebra Booleana a partir da Tabela 4–1: a variável multiplicada pelo seu complemento é igual a 0.

Passo 1. Acrescente a cada termo-produto não padrão um termo constituído do produto da variável que não aparece pelo complemento dela. Isso resulta em dois termos-soma. Como sabemos, podemos somar 0 com qualquer coisa sem alterar o seu valor.

Passo 2. Aplique a Regra 12 a partir da Tabela 4–1: $A + BC = (A + B)(A + C)$

Passo 3. Repita o passo 1 até que todos os termos-soma resultantes contenham todas as variáveis do domínio na forma complementada ou não complementada.

EXEMPLO 4–15

Converta a seguinte expressão Booleana para a forma de produto-de-somas padrão:

$$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$$

Solução O domínio dessa expressão de produto-de-somas é A, B, C e D. Trabalhe com um termo de cada vez. No primeiro termo, $A + \overline{B} + C$, a variável D ou $\overline{D}$ não aparece, assim, acrescentamos $D\overline{D}$ e aplicamos a Regra 12 como mostrado a seguir:

$$A + \overline{B} + C = A + \overline{B} + C + D\overline{D} = (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})$$

No segundo termo, $\overline{B} + C + \overline{D}$, a variável A ou $\overline{A}$ não aparece, assim, acrescentamos $A\overline{A}$ e aplicamos a Regra 12 como mostrado a seguir:

$$\overline{B} + C + \overline{D} = \overline{B} + C + \overline{D} + A\overline{A} = (A + \overline{B} + C + \overline{D})(\overline{A} + \overline{B} + C + \overline{D})$$

O terceiro termo, $A + \overline{B} + \overline{C} + D$, já está na forma padrão. A forma do produto-de-somas padrão a partir da expressão original é:

$$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D) =$$
$$(A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D})(\overline{A} + \overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$$

Problema relacionado Converta $A + \overline{B} + C + \overline{D}$ para a forma de produto-de-somas.

Representação Binária de um Termo-Soma Padrão Um termo-soma padrão é igual a 0 para apenas uma combinação de valores das variáveis. Por exemplo, o termo-soma $A + \overline{B} + C + \overline{D}$ é 0 quando $A = 0$, $B = 1$, $C = 0$ e $D = 1$, conforme mostrado abaixo, e é 1 para todas as outras combinações de valores das variáveis.

$$A + \overline{B} + C + \overline{D} = 0 + \overline{1} + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

Nesse caso, o termo-soma tem um valor binário e 0101 (decimal 5). Lembre-se, um termo-soma é implementado com uma porta OR cuja saída é 0 apenas se cada uma de suas entradas for 0. Inversores são usados para produzir os complementos das variáveis conforme necessário.

Uma expressão na forma de produto-de-somas é igual a 0 apenas se um ou mais termos-soma na expressão for igual a 0.

EXEMPLO 4–16

Determine os valores binários das variáveis para os quais as seguintes expressões produto-de-somas sejam iguais a zero.

$$(A + B + C + D)(A + \overline{B} + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$$

Solução O termo $A + B + C + D$ é igual a 0 quando $A = 0$, $B = 0$, $C = 0$ e $D = 0$.

$$A + B + C + D = 0 + 0 + 0 + 0 = 0$$

O termo $A + \overline{B} + \overline{C} + D$ é igual a zero quando $A = 0$, $B = 1$, $C = 1$ e $D = 0$.

$$A + \overline{B} + \overline{C} + D = 0 + \overline{1} + \overline{1} + 0 = 0 + 0 + 0 + 0 = 0$$

O termo $\overline{A} + \overline{B} + \overline{C} + \overline{D}$ é igual a 0 quando $A = 1$, $B = 1$, $C = 1$ e $D = 1$.

$$\overline{A} + \overline{B} + \overline{C} + \overline{D} = \overline{1} + \overline{1} + \overline{1} + \overline{1} = 0 + 0 + 0 + 0 = 0$$

A expressão de produto-de-somas é igual a 0 quando qualquer dos três termos-soma for igual a 0.

Problema relacionado Determine os valores binários para os quais a seguinte expressão de produto-de-somas é igual a 0:

$$(X + \overline{Y} + Z)(\overline{X} + Y + Z)(X + Y + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z})(X + \overline{Y} + \overline{Z})$$

Essa expressão é um produto-de-somas padrão?

Conversão de uma Soma-de-Produtos Padrão para um Produto-de-Somas Padrão

Os valores binários dos termos-produto numa dada expressão de soma-de-produtos não estão presentes na expressão equivalente de produto-de-somas padrão. Além disso, os valores binários que não são representados na expressão de soma-de-produtos estão presentes na expressão equivalente de produto-de-somas. Portanto, para converter de soma-de-produtos padrão para produto-de-somas padrão, os passos a seguir são realizados:

Passo 1. Avalie cada termo-produto na expressão de soma-de-produtos. Ou seja, determine os números binários que representam os termos-produto.

Passo 2. Determine todos os números binários não incluídos na avaliação no Passo 1.

Passo 3. Escreva o termo-soma equivalente para cada número binário a partir do passo 2 e os expresse na forma de produto-de-somas.

EXEMPLO 4–17

Converta a seguinte expressão de soma-de-produtos para uma expressão equivalente de produto-de-somas:

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

Solução A avaliação é a seguinte:

$$000 + 010 + 011 + 101 + 111$$

Como existem três variáveis de domínio nessa expressão, existe um total de oito (2^3) combinações possíveis. A expressão de soma-de-produtos contém cinco dessas combinações, assim, o produto-de-somas tem que conter os outros três os quais são 001, 100 e 110.

Lembre-se, esses são os valores binários que tornam o termo-soma 0. A expressão de produto-de-somas equivalente é

$$(A + B + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + C)$$

Problema relacionado Verifique que as expressões de soma-de-produtos e produto-de-somas nesse exemplo são equivalentes substituindo os valores binários em cada uma.

SEÇÃO 4–6 REVISÃO

1. Identifique cada uma das seguintes expressões como soma-de-produtos, soma-de-produtos padrão, produto-de-somas e produto-de-somas padrão.
   (a) $AB + \overline{A}BD + \overline{A}C\overline{D}$ **(b)** $(A + \overline{B} + C)(A + B + \overline{C})$
   (c) $\overline{A}BC + AB\overline{C}$ **(d)** $A(A + \overline{C})(A + B)$
2. Converta cada expressão de soma-de-produtos na Questão 1 para a forma padrão.
3. Converta cada expressão de produto-de-somas na Questão 1 para a forma padrão.

4-7 EXPRESSÕES BOOLEANAS E TABELAS-VERDADE

Todas as expressões Booleanas padrão podem ser facilmente convertidas no formato de uma tabela-verdade usando valores binários para cada termo na expressão. A tabela-verdade é uma forma comum de apresentação, num formato conciso, da operação lógica de um circuito. Além disso, expressões de soma-de-produtos padrão ou produto-de-somas podem ser determinadas a partir de uma tabela-verdade. Encontramos tabelas-verdade em folhas de dados e outras literaturas relacionadas à operação de circuitos digitais.

Ao final do estudo desta seção você deverá ser capaz de:

■ Converter uma expressão de soma-de-produtos padrão no formato de tabela-verdade ■ Converter uma expressão de produto-de-somas padrão no formato de tabela-verdade ■ Obter uma expressão padrão a partir de uma tabela-verdade ■ Interpretar adequadamente os dados de uma tabela-verdade.

Conversão de Expressões de soma-de-produtos para o Formato de Tabela-Verdade

Lembre-se, da Seção 4–6, de que uma expressão de soma-de-produtos é igual a 1 apenas se pelo menos um dos termos-produto for igual a 1. Uma tabela-verdade é simplesmente uma lista de combinações possíveis dos valores das variáveis de entrada e os correspondentes valores de saída (1 ou 0). Para uma expressão com um domínio de duas variáveis, existem quatro combinações diferentes para as variáveis ($2^2 = 4$). Para uma expressão com um domínio de três variáveis, existem oito diferentes combinações de variáveis ($2^3 = 8$). Para uma expressão com um domínio de quatro variáveis, existem dezesseis combinações diferentes para as variáveis ($2^4 = 16$), e assim por diante.

O primeiro passo na construção de uma tabela-verdade é fazer uma lista de todas as combinações possíveis dos valores binários das variáveis na expressão. Em seguida, converta a expressão de soma-de-produtos para a forma padrão caso ela não esteja nesse formato. Finalmente, coloque um 1 na coluna de saída (X) para cada valor binário que torna a expressão de soma-de-produtos padrão um 1 e coloque um 0 para todos os valores binários restantes. Esse procedimento é ilustrado no Exemplo 4–18.

EXEMPLO 4–18

Desenvolva uma tabela-verdade para a expressão de soma-de-produtos $\overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + ABC$.

Solução Existem três variáveis no domínio, assim existem oito combinações possíveis de valores binários das variáveis conforme listado nas três colunas à esquerda da Tabela 4–6. Os valores binários que tornam os termos-produto nas expressões iguais a 1 são $\overline{A}\,\overline{B}C$: 001; $A\overline{B}\,\overline{C}$: 100; e ABC: 111. Para cada um desses valores binários, coloque um 1 na coluna de saída como mostrado na tabela. Para cada uma das combinações binárias restantes, coloque um 0 na coluna de saída.

▶ **TABELA 4–6**

ENTRADAS			SAÍDA	
A	*B*	*C*	*X*	TERMO PRODUTO
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

Problema relacionado Crie uma tabela-verdade para a expressão de soma-de-produtos padrão $\overline{A}B\overline{C} + A\overline{B}C$.

Conversão de Expressões de Produto-de-Somas para o Formato de Tabela-verdade

Lembre-se de que uma expressão de produto-de-somas é igual a 0 apenas se pelo menos um dos termos-soma for igual a 0. Para construir uma tabela-verdade a partir de uma expressão de produto-de-somas, faça uma lista de todas as combinações possíveis de valores binários das variáveis da mesma forma que foi feito para a expressão de soma-de-produtos. Em seguida, converta a expressão de produto-de-somas para a forma padrão, caso ela ainda não esteja nesta forma. Finalmente, coloque um 0 na coluna de saída (X) para cada valor binário que torna a expressão um 0 e coloque um 1 para todos os outros valores binários restantes. Esse procedimento está ilustrado no Exemplo 4–19.

EXEMPLO 4–19

Determine a tabela-verdade para a seguinte expressão de produto-de-somas:
$(A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$

Solução Existem três variáveis no domínio e as oito possibilidades de valores binários estão listadas nas três colunas à esquerda da Tabela 4–7. Os valores binários que tornam os termos-soma na expressão iguais a 0 são $A + B + C$: 000; $A + \overline{B} + C$: 010; $A + \overline{B} + \overline{C}$: 011; $\overline{A} + B + \overline{C}$: 101 e $\overline{A} + \overline{B} + C$: 110. Para cada um desses valores binários, coloque um 0 na coluna de saída conforme mostra a tabela. Para cada uma das combinações binárias restantes, coloque um 1 na coluna de saída.

▶ **TABELA 4–7**

ENTRADAS A	B	C	SAÍDA X	TERMO-SOMA
0	0	0	0	$(A + B + C)$
0	0	1	1	
0	1	0	0	$(A + \overline{B} + C)$
0	1	1	0	$(A + \overline{B} + \overline{C})$
1	0	0	1	
1	0	1	0	$(\overline{A} + B + \overline{C})$
1	1	0	0	$(\overline{A} + \overline{B} + C)$
1	1	1	1	

Observe que a tabela-verdade nesse exemplo é a mesma que para o Exemplo 4–18. Isso significa que a expressão de soma-de-produtos no exemplo anterior e a expressão de produto-de-somas nesse exemplo são equivalentes.

Problema relacionado Desenvolva a tabela-verdade para a seguinte expressão de produto-de-somas padrão:

$$(A + \overline{B} + C)(A + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})$$

Determinação de Expressões Padrão a partir de uma Tabela-Verdade

Para determinar a expressão de soma-de-produtos padrão representada por uma tabela-verdade, faça uma lista dos valores binários das variáveis de entrada para os quais a saída seja 1. Converta cada valor binário para o termo-produto correspondente substituindo cada 1 pela variável correspondente e cada 0 pelo complemento da variável correspondente. Por exemplo, o valor binário 1010 é convertido para o termo-produto mostrado a seguir:

$$1010 \longrightarrow A\overline{B}C\overline{D}$$

Se substituirmos os valores, veremos que o termo-produto é 1:

$$A\overline{B}C\overline{D} = 1 \cdot \overline{0} \cdot 1 \cdot \overline{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Para determinar a expressão de produto-de-somas representado pela tabela-verdade, liste os valores binários para os quais a saída é 0. Converta cada valor binário para o correspondente termo-soma substituindo cada 1 pelo complemento da variável correspondente e cada 0 pela variável correspondente. Por exemplo, o valor binário 1001 é convertido num termo-soma como mostrado a seguir:

$$1001 \longrightarrow \overline{A} + B + C + \overline{D}$$

Se substituirmos os valores, veremos que o termo-soma é 0.

$$\overline{A} + B + C + \overline{D} = \overline{1} + 0 + 0 + \overline{1} = 0 + 0 + 0 + 0 = 0$$

EXEMPLO 4–20

A partir da tabela-verdade na Tabela 4–8, determine a expressão de soma-de-produtos padrão e a expressão equivalente de produto-de-somas padrão.

TABELA 4–8

ENTRADAS			SAÍDA
A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solução Existem quatro 1s na coluna de saída e os valores binários correspondentes são 011, 110 e 111. Converta esses valores binários para termos-produto como mostrado a seguir:

$$011 \longrightarrow \overline{A}BC$$
$$100 \longrightarrow A\overline{B}\,\overline{C}$$
$$110 \longrightarrow AB\overline{C}$$
$$111 \longrightarrow ABC$$

A expressão de soma-de-produtos padrão resultante para a saída *X* é

$$X = \overline{A}BC + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$

Para a expressão de produto-de-somas, a saída é 0 para os valores binários 000, 001, 010 e 101. Converta esses valores binários para termos-soma como mostrado a seguir:

$$000 \quad A + B + C$$
$$001 \quad A + B + \overline{C}$$
$$010 \quad A + \overline{B} + C$$
$$101 \quad \overline{A} + B + \overline{C}$$

A expressão de produto-de-somas padrão resultante para a saída *X* é

$$X = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + \overline{C})$$

Problema relacionado Através da substituição dos valores binários, mostre que as expressões de soma-de-produtos e de produto-de-somas obtidas nesse exemplo são equivalentes; ou seja, para qualquer valor binário ambas devem ser 1 ou 0.

SEÇÃO 4–7 REVISÃO

1. Se uma certa expressão Booleana tem um domínio de cinco variáveis, quantos valores binários terão a tabela-verdade?
2. Numa certa tabela-verdade, a saída é 1 para o valor binário 0110. Converta esse valor binário para o termo-produto correspondente usando as variáveis *W*, *X*, *Y* e *Z*.
3. Numa certa tabela-verdade, a saída é 0 para o valor binário 1100. Converta esse valor binário para o termo-soma correspondente usando as variáveis *W*, *X*, *Y* e *Z*.

4-8 O MAPA DE KARNAUGH

Um mapa de Karnaugh provê um método sistemático para simplificação de expressões Booleanas e, se usado adequadamente, produz a expressão de soma-de-produtos ou de produto-de-somas mais simples possível, conhecida como expressão mínima. Conforme já vimos, a efetividade da simplificação algébrica depende da nossa familiaridade com todas as leis, regras e teoremas da álgebra Booleana e da habilidade de cada um em aplicá-las. Por outro lado, o mapa de Karnaugh provê um método tipo "livro de receitas" para simplificação.

Ao final do estudo desta seção você deverá ser capaz de:

■ Construir um mapa de Karnaugh para três ou quatro variáveis ■ Determinar o valor binário de cada célula num mapa de Karnaugh ■ Determinar o termo-produto padrão representado por cada célula num mapa de Karnaugh ■ Explicar a adjacência de células e identificar células adjacentes

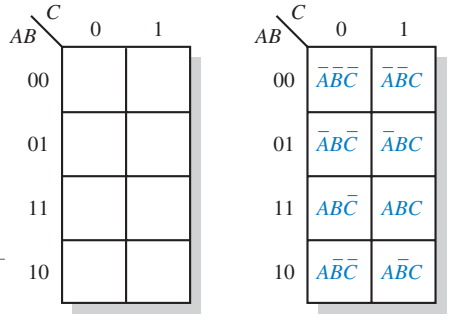
A finalidade do mapa de Karnaugh é simplificar uma expressão Booleana.

Um **mapa de Karnaugh** é similar a uma tabela-verdade porque todos os valores possíveis das variáveis de entrada e a saída resultante para cada valor estão presentes no mapa. Em vez de estar organizado em colunas e linhas como uma tabela-verdade, o mapa de Karnaugh é um arranjo de **células** no qual cada célula representa um valor binário das variáveis de entrada. As células são arranjadas de forma que a simplificação de uma dada expressão é obtida simplesmente fazendo um agrupamento adequado de células. Os mapas de Karnaugh podem ser usados para expressões com duas, três, quatro e cinco variáveis, porém discutiremos apenas as situações de 3 e 4 variáveis para ilustrar os princípios. A Seção 4–11 apresenta o caso de 5 variáveis usando um mapa de Karnaugh de 32 células. Um outro método, que está além do escopo desse livro, denominado de método Quine-McClusky pode ser usado para um número maior de variáveis.

O número de células num mapa de Karnaugh é igual ao número total de combinações possíveis das variáveis de entrada que é igual ao número de linhas na tabela-verdade. Para o caso de três variáveis, o número de células é $2^3 = 8$. Para quatro variáveis, o número de células é $2^4 = 16$.

O Mapa de Karnaugh de 3 Variáveis

O mapa de Karnaugh de 3 variáveis é um arranjo de oito células, conforme mostra a Figura 4–21(a). Nesse caso, A, B e C são usadas como variáveis embora outras letras poderiam ser usadas. Os valores binários de A e B estão ao longo do lado esquerdo (observe a seqüência) e os valores de C estão na parte superior. O valor de uma dada célula corresponde aos valores binários de A e B à esquerda na mesma linha combinados com o valor de C na parte superior na mesma coluna. Por exemplo, a célula no canto superior esquerdo tem um valor binário de 000 e a célula no canto inferior direito tem um valor binário de 101. A Figura 4–21(b) mostra os termos-produto padrão que são representados por cada célula do mapa de Karnaugh.



▶ **FIGURA 4–21**

Um mapa de Karnaugh de 3 variáveis mostrando os termos-produto.

O Mapa de Karnaugh de 4 Variáveis

O mapa de Karnaugh de 4 variáveis é um arranjo de dezesseis células, conforme mostra a Figura 4–22(a). Os valores binários de A e B estão ao longo do lado esquerdo e os valores de C e D estão

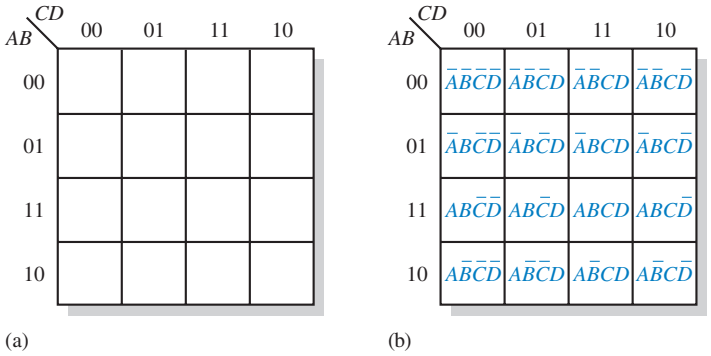


FIGURA 4–22
Um mapa de Karnaugh de 4 variáveis.

na parte superior. O valor de uma dada célula corresponde aos valores binários de *A* e *B* à esquerda na mesma linha combinados com os valores binários de *C* e *D* na parte superior na mesma coluna. Por exemplo, a célula no canto superior direito tem um valor binário de 0010 e a célula no canto inferior direito tem um valor binário de 1010. A Figura 4–22(b) mostra os termos-produto padrão que são representados por cada célula no mapa de Karnaugh de 4 variáveis.

Célula Adjacente

Células que diferem em apenas uma variável são adjacentes.

Células com valores que diferem em mais de uma variável não são adjacentes.

As células num mapa de Karnaugh são arranjadas de forma que exista apenas uma mudança simples de variável entre células adjacentes. A **adjacência** é definida por uma mudança simples de variável. Num mapa de 3 variáveis a célula 010 é adjacente à célula 000, à célula 011 e à célula 110. A célula 010 não é adjacente à célula 001, nem à célula 111, nem à célula 100 ou à célula 101.

Fisicamente, cada célula é adjacente a células que estão imediatamente próximas a ela por qualquer um dos seus quatro lados. Uma célula não é adjacente às células que tocam diagonalmente qualquer um dos vértices. Além disso, as células na linha superior são adjacentes às células correspondentes na linha inferior e as células na coluna mais à esquerda são adjacentes à células correspondentes na coluna mais à direita. Isso é denominado de adjacência cilíndrica porque podemos pensar no mapa enrolado de cima para baixo formando um cilindro ou da esquerda para a direita formando um cilindro. A Figura 4–23 ilustra a adjacência de células com um mapa de 4 variáveis, embora as mesmas regras se aplicam a mapas de Karnaugh com qualquer número de células.

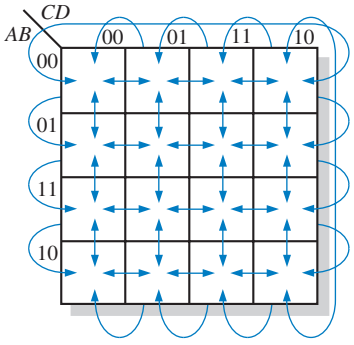


FIGURA 4–23
Células adjacentes num mapa de Karnaugh são aquelas que diferem uma da outra em apenas uma variável. As setas indicam as células adjacentes.

SEÇÃO 4–8 REVISÃO

1. Num mapa de Karnaugh de 3 variáveis, qual é o valor binário para cada célula nas seguintes localizações:
   (a) canto superior esquerdo **(b)** canto inferior direito
   (c) canto inferior esquerdo **(d)** canto superior direito
2. Qual é o termo-produto padrão para cada célula na Questão 1 para as variáveis *X*, *Y* e *Z*?
3. Repita a Questão 1 para um mapa de 4 variáveis.
4. Repita a Questão 2 para um mapa de 4 variáveis usando as variáveis *W*, *X*, *Y* e *Z*.

4-9 MINIMIZAÇÃO DE SOMA-DE-PRODUTOS USANDO O MAPA DE KARNAUGH

Conforme dito na seção anterior, o mapa de Karnaugh é usado para simplificação de expressões Booleanas para a forma mínima. Uma expressão de soma-de-produtos minimizada contém a menor quantidade possível de termos com a menor quantidade possível de variáveis por termo. Geralmente uma expressão de soma-de-produtos mínima pode ser implementada com menos portas lógicas que uma expressão padrão.

Ao final do estudo desta seção você deverá ser capaz de:

■ Inserir no mapa uma expressão de soma-de-produtos padrão ■ Agrupar os 1s no mapa em grupos de tamanho máximo ■ Determinar o termo-produto mínimo para cada grupo no mapa ■ Combinar os termos-produto mínimo para formar uma expressão de soma-de-produtos mínima ■ Converter uma tabela verdade num mapa de Karnaugh para simplificação da expressão representada ■ Usar condições "don't care" no mapa de Karnaugh

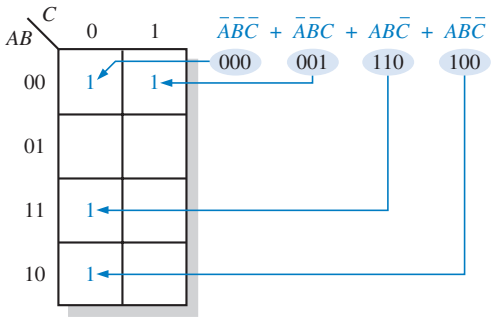
Mapeando uma Expressão Padrão de soma-de-produtos

Para uma expressão na forma de soma-de-produtos padrão, um 1 é colocado no mapa de Karnaugh para cada termo-produto na expressão. Cada 1 é colocado na célula correspondente ao valor de um termo-produto. Por exemplo, para o termo-produto $A\overline{B}C$, um 1 é colocado na célula 101 num mapa de 3 variáveis.

Quando uma expressão de soma-de-produtos é completamente inserida no mapa, existirá um número de 1s no mapa de Karnaugh igual ao número de termos-produto na expressão de soma-de-produtos padrão. As células que não possuem um 1 são as células para as quais a expressão é 0. Geralmente, quando trabalhamos com expressões de soma-de-produtos, os 0s são deixados fora do mapa. Os passos a seguir e a ilustração mostrada na Figura 4–24 apresentam o processo de inserção da expressão no mapa.

Passo 1 Determine o valor binário de cada termo-produto na expressão de soma-de-produtos. Após adquirir alguma prática, podemos geralmente fazer a avaliação dos termos mentalmente.

Passo 2 À medida que cada termo-produto é avaliado, coloque um 1 no mapa de Karnaugh na célula que tem o mesmo valor que o termo-produto.



▶ **FIGURA 4–24**
Exemplo de inserção de uma expressão de soma-de-produtos no mapa.

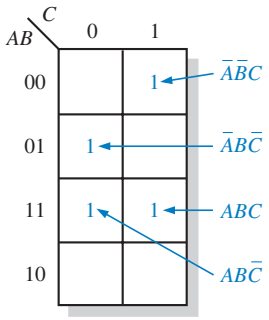
EXEMPLO 4–21

Coloque no mapa de Karnaugh a seguinte expressão de soma-de-produtos:

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$

Solução Avalie a expressão conforme mostrado a seguir. Coloque um 1 no mapa de Karnaugh de 3 variáveis, visto na Figura 4–25, para cada termo-produto padrão da expressão.

$$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$$
$$001 \quad 010 \quad 110 \quad 111$$



▶ FIGURA 4–25

Problema relacionado Coloque num mapa de Karnaugh a expressão de soma-de-produtos $\overline{A}BC + A\overline{B}C + A\overline{B}\,\overline{C}$.

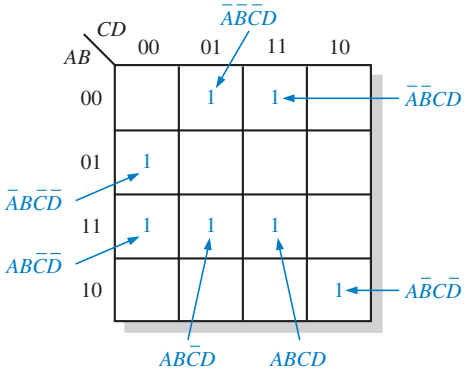
EXEMPLO 4–22

Coloque a seguinte expressão de soma-de-produtos num mapa de Karnaugh:

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$$

Solução Avalie a expressão conforme mostrado abaixo. Coloque um 1 no mapa de Karnaugh de 4 variáveis, visto na Figura 4–26, para cada termo-produto padrão da expressão.

$$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$$
$$0011 \quad 0100 \quad 1101 \quad 1111 \quad 1100 \quad 0001 \quad 1010$$



▶ FIGURA 4–26

Problema relacionado Coloque num mapa de Karnaugh a seguinte expressão de soma-de-produtos padrão:

$$\overline{A}BC\overline{D} + ABC\overline{D} + AB\overline{C}\,\overline{D} + ABCD$$

Mapeando uma Expressão Não Padrão de Soma-de-Produtos

Uma expressão Booleana tem que estar primeiro na forma padrão antes de usarmos o mapa de Karnaugh. Se uma expressão não estiver na forma padrão, então ela deve ser convertida para a forma padrão usando o procedimento abordado na Seção 4–6 ou através de expansão numérica. Como uma expressão deve ser avaliada antes de colocar no mapa, a expansão numérica é provavelmente a forma mais eficiente.

Expansão Numérica de um Termo-Produto Lembre que um termo-produto não padrão tem uma ou mais variáveis que não aparecem. Por exemplo, considere que um dos termos-produto nu-

ma certa expressão de soma-de-produtos de 3 variáveis seja $A\overline{B}$. Esse termo pode ser expandido numericamente para a forma padrão conforme explicado a seguir. Primeiro, escreva o valor binário das duas variáveis e acrescente um 0 para a variável que não aparece $\overline{C}$: 100. Em seguida, escreva o valor binário das duas variáveis e acrescente um 1 para a variável C: 100 que não aparece. Os dois números binários resultantes são os valores dos termos da soma-de-produtos padrão $A\overline{B}\,\overline{C}$ e $A\overline{B}C$.

Como um outro exemplo, considere que um dos termos-produto na expressão de 3 variáveis seja B (lembre que uma única variável conta como um termo-produto numa expressão de soma-de-produtos). Esse termo pode ser expandido numericamente para a forma padrão como mostrado a seguir. Escreva o valor binário da variável: então acrescente todos os valores possíveis para as variáveis A e C que não aparecem como mostrado a seguir:

B
010
011
110
111

Os quatro números binários resultantes são os valores dos termos $\overline{A}\overline{B}\overline{C}$, $\overline{A}BC$, $AB\overline{C}$ e ABC da soma-de-produtos padrão.

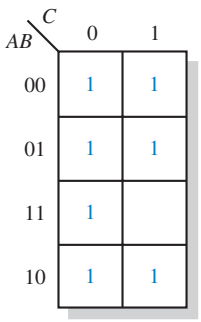
EXEMPLO 4–23

Insira no mapa de Karnaugh a seguinte expressão de soma-de-produtos: $\overline{A} + A\overline{B} + AB\overline{C}$.

Solução A expressão de soma-de-produtos não está obviamente na forma padrão porque cada termo-produto não possui as três variáveis. No primeiro termo não aparecem duas variáveis e no segundo termo não aparece uma variável. Já o terceiro termo está na forma padrão. Primeiro faça a expansão numérica dos termos como mostrado a seguir:

$\overline{A}$	$A\overline{B}$	$AB\overline{C}$
000	100	110
001	101	
010		
011		

Preencha o mapa com os valores binários resultantes colocando um 1 na célula apropriada do mapa de Karnaugh de 3 variáveis mostrado na Figura 4–27.



► FIGURA 4–27

Problema relacionado Insira no mapa de Karnaugh a expressão de soma-de-produtos $BC + \overline{A}\,\overline{C}$.

EXEMPLO 4–24

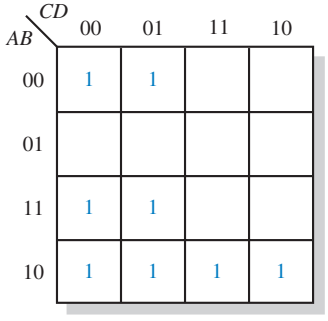
Insira no mapa de Karnaugh a seguinte expressão de soma-de-produtos:

$$\overline{B}\,\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$$

Solução A expressão de soma-de-produtos não está obviamente na forma padrão porque cada termo-produto não tem as quatro variáveis. No primeiro e segundo termos não aparecem duas variáveis e os termos restantes já estão na forma padrão. Primeiro faça a expansão numéricas dos termos incluindo todas as combinações das variáveis que não aparecem como mostrado a seguir:

$\overline{B}\,\overline{C}$	$A\overline{B}$	+	$AB\overline{C}$	+ $A\overline{B}C\overline{D}$	+ $\overline{A}\,\overline{B}\,\overline{C}D$	+ $A\overline{B}CD$
0000	1000		1100	1010	0001	1011
0001	1001		1101			
1000	1010					
1001	1011					

Preencha o mapa com os valores binários resultantes colocando um 1 na célula apropriada do mapa de Karnaugh de 4 variáveis mostrado na Figura 4–28. Observe que alguns dos valores na expressão expandida são redundantes.



▶ **FIGURA 4–28**

Problema relacionado Insira no mapa de Karnaugh a expressão $A + \overline{C}D + AC\overline{D} + \overline{A}BC\overline{D}$.

Simplificação via Mapa de Karnaugh de Expressões de soma-de-produtos

O processo que resulta numa expressão que contém o menor número de termos possível com o menor número de variáveis possível é denominado de **minimização**. Após a expressão de soma-de-produtos ser inserida no mapa, uma expressão de soma-de-produtos mínima é obtida agrupando os 1s e determinando a expressão de soma-de-produtos mínima a partir do mapa.

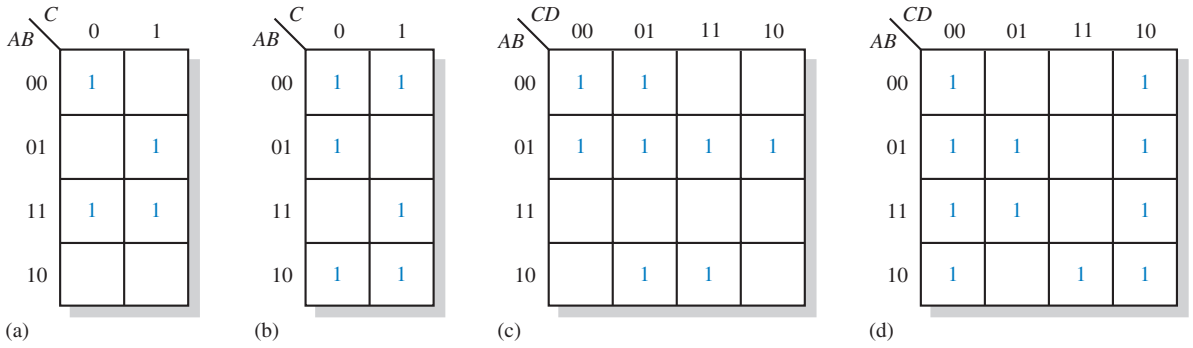
Agrupando os 1s Podemos fazer grupos de 1s no mapa de Karnaugh de acordo com as regras apresentadas em seguida, enlaçando aquelas células adjacentes que contêm 1s. A meta é maximizar o tamanho dos grupos e minimizar o número de grupos.

1. Um grupo tem que conter 1, 2, 4, 8 ou 16 células, cujos números são potências inteiras de 2. No caso de um mapa de 3 variáveis, $2^3 = 8$ células é o grupo máximo.
2. Cada célula num grupo tem que ser adjacente a uma ou mais células do mesmo grupo, porém todas as células não têm que ser adjacentes uma da outra.
3. Sempre inclua o maior número de 1s num grupo de acordo com a regra 1.

4. Cada 1 no mapa tem que ser incluído em pelo menos um grupo. Os 1s que já fazem parte de um grupo podem ser incluídos num outro grupo enquanto os grupos sobrepostos incluem 1s não comuns.

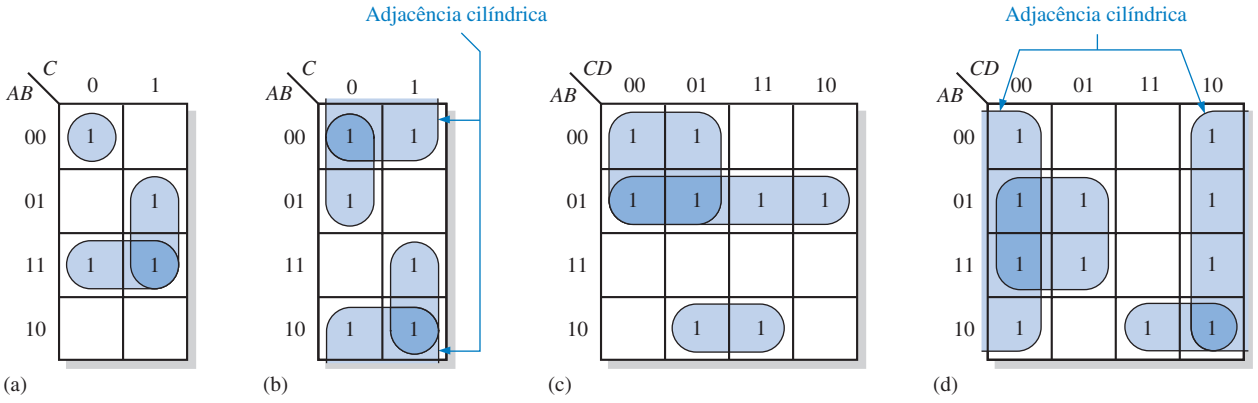
EXEMPLO 4–25

Agrupe os 1s em cada um dos mapas de Karnaugh mostrados na Figura 4–29.



▲ FIGURA 4–29

Solução Os agrupamentos são mostrados na Figura 4–30. Em alguns casos, existem mais de uma forma de agrupar os 1s para formar agrupamentos máximos.



▲ FIGURA 4–30

Problema relacionado Determine, caso existam, outras formas de agrupar os 1s nos mapas da Figura 4–30 para obter um número mínimo de grupos máximos.

Determinação da Expressão de soma-de-produtos Mínima a partir do Mapa Quando todos os 1s que representam termos-produto padrão estão adequadamente inseridos no mapa e agrupados, começa o processo de determinação da expressão de soma-de-produtos mínima resultante. As regras a seguir são aplicadas para determinar os termos-produto mínimos e a expressão de soma-de-produtos mínima:

1. Agrupe as células que têm 1s. Cada grupo de células que contém 1s cria um termo-produto composto de todas as variáveis que ocorrem num formato apenas (não complementada ou

complementada) dentro do grupo. Variáveis que ocorrem tanto de forma complementada quanto não complementada dentro do grupo são eliminadas. Essas são denominadas de *variáveis contraditórias*.

2. Determine o termo-produto mínimo para cada grupo.

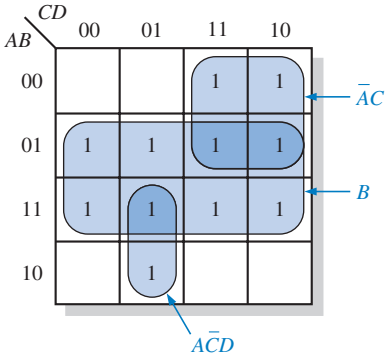
a. Para um mapa de 3 variáveis:
- **(1)** Um grupo de 1 célula resulta num termo-produto de 3 variáveis
- **(2)** Um grupo de 2 células resulta num termo-produto de 2 variáveis
- **(3)** Um grupo de 4 células resulta num termo-produto de 1 variáveis
- **(4)** Um grupo de 8 células resulta num valor 1 para a expressão

b. Para um mapa de 4 variáveis:
- **(1)** Um grupo de 1 célula resulta num termo-produto de 4 variáveis
- **(2)** Um grupo de 2 células resulta num termo-produto de 3 variáveis
- **(3)** Um grupo de 4 células resulta num termo-produto de 2 variáveis
- **(4)** Um grupo de 8 células resulta num termo de 1 variável
- **(5)** Um grupo de 16 células resulta numa expressão de valor 1

3. Quando se obtém todos os termos-produto mínimos a partir do mapa de Karnaugh, eles são somados para formar a expressão de soma-de-produtos mínima.

EXEMPLO 4–26

Determine os termos-produto para o mapa de Karnaugh visto na Figura 4–31 e escreva a expressão de soma-de-produtos mínima.



► FIGURA 4–31

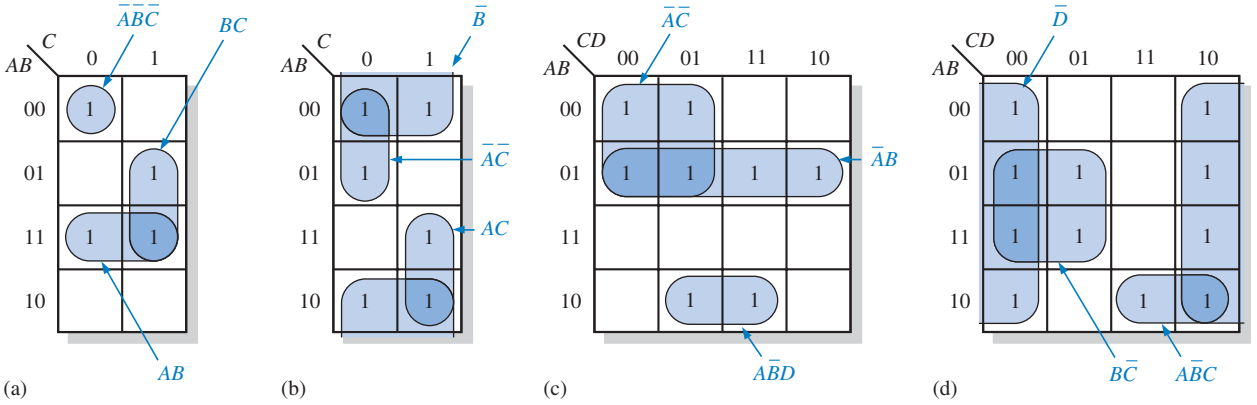
Solução Elimine as variáveis de um grupo que estão na forma complementada e não complementada. Na Figura 4–31, o termo-produto para o grupo de 8 células é B porque as células dentro desse grupo contêm tanto A quanto $\overline{A}$, tanto C quanto $\overline{C}$ e tanto D quanto $\overline{D}$, as quais são eliminadas. O grupo de 4 células contém, B, $\overline{B}$, D e $\overline{D}$, sobrando $\overline{A}$ e C, as quais formam o termo-produto $\overline{A}C$. O grupo de 2 células contém B e $\overline{B}$, sobrando as variáveis A, $\overline{C}$ e D, as quais formam o termo-produto $A\overline{C}D$. Observe como a sobreposição de grupos é usada para maximizar o tamanho dos grupos. A expressão de soma-de-produtos mínimas resultante é a soma desses termos-produto:

$$B + \overline{A}C + A\overline{C}D$$

Problema relacionado Para o mapa de Karnaugh mostrado na Figura 4–31, acrescente um 1 na célula do canto inferior direito (1010) e determine a expressão de soma-de-produtos mínima resultante.

EXEMPLO 4–27

Determine os termos-produto para cada um dos mapas de Karnaugh mostrados na Figura 4–32 e escreva a expressão de soma-de-produtos mínima.



▲ FIGURA 4–32

Solução O termo-produto mínimo resultante para cada grupo é mostrado na Figura 4–32. As expressões de soma-de-produtos mínima para cada mapa de Karnaugh da figura são:

(a) $AB + BC + \overline{A}\,\overline{B}\,\overline{C}$ **(b)** $\overline{B} + \overline{A}\,\overline{C} + AC$

(c) $\overline{A}B + \overline{A}\,\overline{C} + A\overline{B}D$ **(d)** $\overline{D} + A\overline{B}C + B\overline{C}$

Problema relacionado Para o mapa de Karnaugh visto na Figura 4–32(d), acrescente um 1 na célula 0111 e determine a expressão de soma-de-produtos mínima resultante.

EXEMPLO 4–28

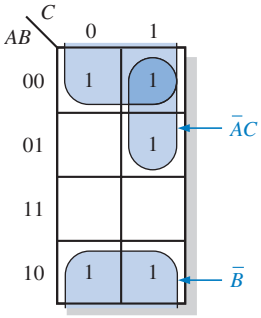
Use um mapa de Karnaugh para minimizar a seguinte expressão de soma-de-produtos padrão:

$$A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}$$

Solução Os valores binários da expressão são:

$$101 + 011 + 011 + 000 + 100$$

Preencha o mapa com a expressão de soma-de-produtos padrão e agrupe as células conforme mostra a Figura 4–33.



▶ FIGURA 4–33

Observe o grupo de 4 células (mapa enrolado como um cilindro) que incluem as linhas superior e inferior de 1s. O 1 que sobra é absorvido num grupo de sobreposição de duas células. O grupo de quatro 1s produz um termo de uma variável, $\overline{B}$. Isso é determinado observando que, dentro do grupo, $\overline{B}$ é a única variável que não muda de uma célula para outra. O grupo de dois 1s produz um termo de 2 variáveis, $\overline{A}C$ Isso é determinado observando que dentro do grupo, $\overline{A}$ e C não mudam de uma célula para a seguinte. O termo-produto para cada grupo é mostrado. A expressão de soma-de-produtos mínima resultante é:

$$\overline{B} + \overline{A}C$$

Tenha em mente que essa expressão mínima é equivalente à expressão padrão original.

Problema relacionado Use um mapa de Karnaugh para minimizar a seguinte expressão de soma-de-produtos:

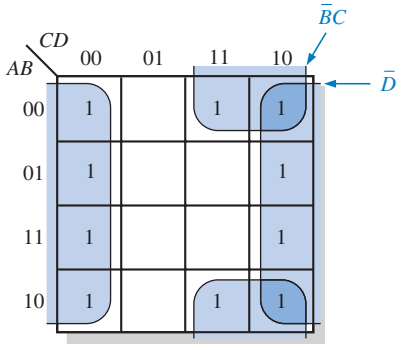
$$X\overline{Y}Z + XY\overline{Z} + \overline{X}YZ + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ$$

EXEMPLO 4–29

Use um mapa de Karnaugh para minimizar a seguinte expressão de soma-de-produtos:

$$\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

Solução O primeiro termo $\overline{B}\,\overline{C}\,\overline{D}$ tem que ser expandido para $A\overline{B}\,\overline{C}\,\overline{D}$ e $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ para se obter a expressão de soma-de-produtos padrão, a qual é então inserida no mapa; e as células são agrupadas conforme mostra a Figura 4–34.



► **FIGURA 4–34**

Observe que os dois grupos apresentam adjacências quando se enrola o mapa para formar um cilindro. O grupo de oito é formado porque as células das colunas externas são adjacentes. O grupo de quatro é formado de forma a enlaçar os dois 1s restantes porque as células das linhas superior e inferior são adjacentes. O termo-produto para cada grupo é mostrado. A expressão de soma-de-produtos mínima resultante é:

$$\overline{D} + \overline{B}C$$

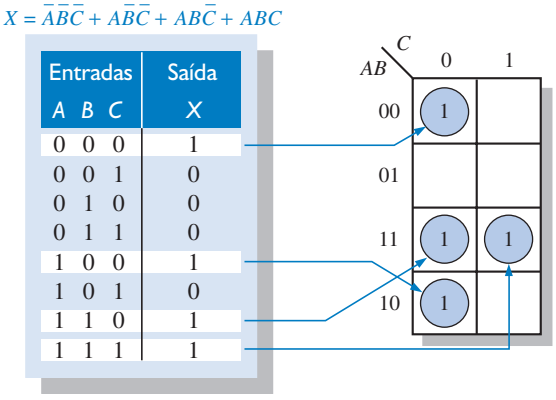
Tenha em mente que essa expressão mínima é equivalente à expressão padrão original.

Problema relacionado Use um mapa de Karnaugh para simplificar a expressão de soma-de-produtos a seguir:

$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

Preenchendo o Mapa Diretamente a partir da Tabela-Verdade

Vimos como inserir uma expressão Booleana no mapa; agora vamos aprender como passar diretamente da tabela-verdade para o mapa de Karnaugh. Lembre-se que uma tabela-verdade fornece a expressão Booleana de saída para todas as combinações de variáveis de entrada possíveis. Um exemplo de uma expressão Booleana e a sua representação em tabela-verdade é mostrado na Figura 4–35. Observe na tabela-verdade que a saída X é 1 para quatro diferentes combinações das variáveis de entrada. Os 1s na coluna de saída da tabela-verdade são transferidos diretamente para o mapa de Karnaugh nas células correspondentes aos valores das combinações das variáveis de entrada associadas, conforme mostra a Figura 4–35. Nessa figura podemos ver que a expressão Booleana, a tabela-verdade e o mapa de Karnaugh são simplesmente formas diferentes de representar uma função lógica.



▶ **FIGURA 4–35**

Exemplo de transferência direta de uma tabela-verdade para um mapa de Karnaugh.

Condições "Don't Care"

Algumas vezes surge uma situação na qual uma combinação das variáveis de entrada não é permitida. Por exemplo, lembre-se que no código BCD abordado no Capítulo 2, existem seis combinações inválidas: 1010, 1011, 1100, 1101, 1110 e 1111. Como esses estados não permitidos nunca ocorrerão numa aplicação envolvendo código BCD, eles podem ser tratados como termos **"don't care"** (não importam) em relação aos seus efeitos na saída. Ou seja, para esses termos "don't care" podemos associar um 1 ou um 0 à saída; na realidade não importa já que eles nunca irão ocorrer.

Os termos "don't care" podem ser usados para se obter vantagens no uso do mapa de Karnaugh. A Figura 4–36 mostra que para cada termo "don't care", um X é colocado na célula. Quando se faz o agrupamento de 1s, os Xs podem ser tratados como 1s para tornar os grupos maiores ou como 0s se eles não representam vantagens. Quanto maior o tamanho de um grupo, mais simplificado será o termo resultante.

A tabela-verdade vista na Figura 4–36(a) descreve uma função lógica que tem uma saída 1 apenas quando o código BCD presente nas entradas for relativo ao 7, 8 ou 9. Se os estados "don't care" forem usados como 1s, a expressão resultante para a função é $A + BCD$, conforme indicado na parte (b) da figura. Se os estados "don't care" não forem usados como 1s, a expressão resultante é $A\bar{B}\bar{C} + \bar{A}BCD$; assim podemos perceber a vantagem de usar termos "don't care" para obter uma expressão mais simples.

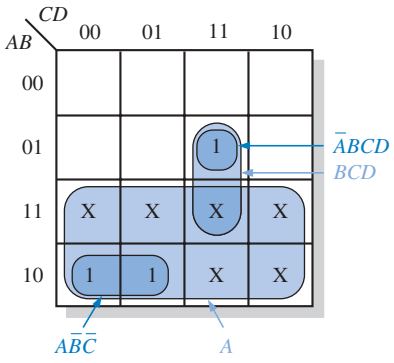
SEÇÃO 4–9 REVISÃO

1. Desenhe os mapas de Karnaugh para três e quatro variáveis.
2. Agrupe os 1s e escreva a expressão de soma-de-produtos simplificada para o mapa de Karnaugh dado na Figura 4–25.
3. Escreva a expressão de soma-de-produtos padrão original para cada um dos mapas de Karnaugh vistos na Figura 4–32.

Entradas				Saída
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(a) Tabela-verdade

Don't cares



(b) Sem "don't cares" $Y = ABC + \bar{A}B\bar{C}D$
Com "don't cares" $Y = A + BCD$

◀ **FIGURA 4–36**
Exemplo de uso de condições "don't care" para simplificar uma expressão.

4-10 MINIMIZAÇÃO DE PRODUTO-DE-SOMAS USANDO O MAPA DE KARNAUGH

Na última seção, estudamos a minimização de uma expressão de soma-de-produtos usando o mapa de Karnaugh. Nessa seção, passamos o foco para as expressões de produto-de-somas. As abordagens são as mesmas exceto que com expressões de produto-de-somas, os 0s representam os termos-soma padrão que são colocados no mapa de Karnaugh em vez de 1s.

Ao final do estudo desta seção você deverá ser capaz de:

■ Fazer o mapa de Karnaugh para uma expressão de produto-de-somas ■ Combinar os 0s no mapa formando grupos de tamanho máximo ■ Determinar o termo-soma mínimo para cada grupo no mapa ■ Combinar os termos-soma mínimos para formar a expressão de produto-de-somas mínima ■ Usar o mapa de Karnaugh para converter entre produto-de-somas e soma-de-produtos

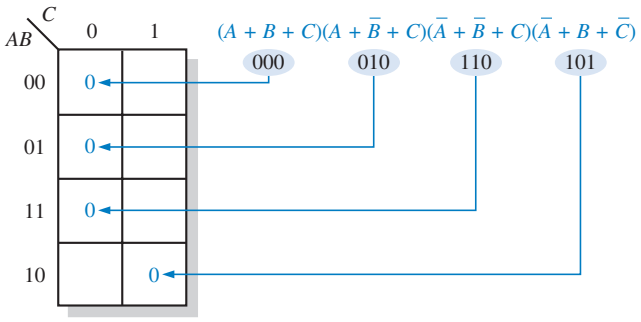
Inserindo no Mapa uma Expressão de Produto-de-Somas Padrão

Para uma expressão de produto-de-somas na forma padrão, um 0 é colocado no mapa de Karnaugh para cada termo-soma na expressão. Cada 0 é colocado na célula que corresponde ao valor de um termo-soma. Por exemplo, para o termo-soma $A + \overline{B} + C$, um 0 é colocado na célula 010 num mapa de 3 variáveis.

Quando uma expressão de produto-de-somas é totalmente inserida no mapa, existirá um número de 0s no mapa de Karnaugh igual ao número de termos-soma na expressão de produto-de-somas padrão. As células que não têm um 0 são as células para as quais a expressão é 1. Geralmente, quando trabalhamos com expressões de produto-de-somas, os 1s são deixados de fora. Os passos a seguir e a ilustração dada na Figura 4–37 mostram o processo de preenchimento do mapa.

Passo 1 Determine o valor binário de cada termo-soma na expressão de produto-de-somas padrão. Esse é um valor binário que torna o termo igual a 0.

Passo 2 À medida que cada termo-soma é avaliado, coloque um 0 no mapa de Karnaugh na célula correspondente.



▶ **FIGURA 4–37**

Exemplo de inserção de uma expressão de produto-de-somas no mapa.

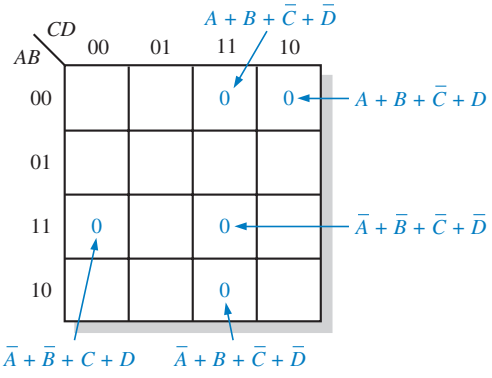
EXEMPLO 4–30

Preencha o mapa de Karnaugh com a seguinte expressão de produto-de-somas padrão:

$$(\overline{A} + \overline{B} + C + D)(\overline{A} + B + \overline{C} + \overline{D})(A + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})(A + B + \overline{C} + \overline{D})$$

Solução Avalie a expressão conforme mostrado a seguir e coloque um 0 no mapa de Karnaugh de 4 variáveis, mostrado na Figura 4–38, para cada termo-soma padrão da expressão.

$$(\overline{A} + \overline{B} + C + D)(\overline{A} + B + \overline{C} + \overline{D})(A + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})(A + B + \overline{C} + \overline{D})$$
$$1100 \qquad 1011 \qquad 0010 \qquad 1111 \qquad 0011$$



▶ **FIGURA 4–38**

Problema relacionado Preencha o mapa de Karnaugh para a seguinte expressão de produto-de-somas padrão:

$$(A + \overline{B} + \overline{C} + D)(A + B + C + \overline{D})(A + B + C + D)(\overline{A} + B + \overline{C} + D)$$

Simplificação por Mapa de Karnaugh de Expressões de Produto-de-Somas

O processo para minimização de uma expressão de produto-de-somas é basicamente o mesmo que para uma expressão de soma-de-produtos exceto que agrupamos os 0s para produzir termos-soma mínimos em vez de agruparmos os 1s e produzirmos termos-produto mínimos. As regras para o agrupamento de 0s são as mesmas que para o agrupamento de 1s que aprendemos na Seção 4–9.

EXEMPLO 4–31

Use um mapa de Karnaugh para minimizar a seguinte expressão de produto-de-somas:

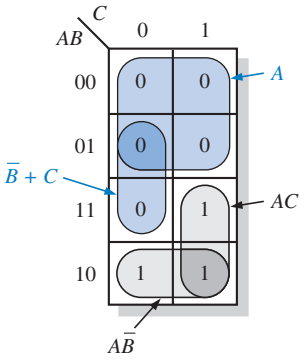
$$(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)$$

Além disso, obtenha a expressão de soma-de-produtos equivalente.

Solução As combinações de valores binários da expressão são:

$$(0+0+0)(0+0+1)(0+1+0)(0+1+1)(1+1+0)$$

Preencha o mapa com a expressão de soma-de-produtos padrão e agrupe as células como mostrado na Figura 4–39.



▶ **FIGURA 4–39**

Observe como o 0 na célula 110 está incluído no grupo de 2 células utilizando um 0 do grupo de 4 células. O termo-soma para cada grupo laranja é mostrado na figura e a expressão de produto-de-somas resultante é:

$$A(\overline{B} + C)$$

Tenha em mente que a expressão de produto-de-somas mínima é equivalente à expressão de produto-de-somas padrão original.

Os agrupamentos de 1s conforme mostrado pelas áreas cinzas resultam numa expressão de soma-de-produtos que é equivalente à expressão obtida pelo agrupamento de 0s.

$$AC + A\overline{B} = A(\overline{B} + C)$$

Problema relacionado Use um mapa de Karnaugh para simplificar a seguinte expressão de produto-de-somas padrão:

$$(X + \overline{Y} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)(\overline{X} + Y + Z)$$

EXEMPLO 4–32

Use um mapa de Karnaugh para minimizar a seguinte expressão de produto-de-somas:

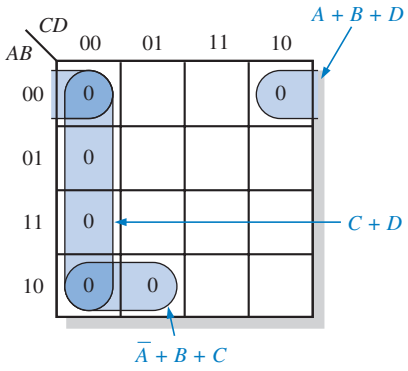
$$(B + C + D)(A + B + \overline{C} + D)(\overline{A} + B + C + \overline{D})(A + \overline{B} + C + D)(\overline{A} + \overline{B} + C + D)$$

Solução O primeiro termo tem que ser expandido para $\overline{A} + B + C + D$ e $A + B + C + D$ para se obter uma expressão de produto-de-somas padrão, a qual é então inserida no mapa; e as

células são agrupadas como mostra a Figura 4–40. O termo-soma para cada grupo é mostrado e a expressão de produto-de-somas mínima resultante é:

$$(C + D)\,(A + B + D)\,(\overline{A} + B + C)$$

Tenha em mente que essa expressão de produto-de-somas mínima é equivalente à expressão de produto-de-somas padrão original.



► FIGURA 4–40

Problema relacionado Use um mapa de Karnaugh para simplificar a seguinte expressão de produto-de-somas:

$$(W + \overline{X} + Y + \overline{Z})(W + X + Y + Z)(W + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)$$

Conversão entre Produto-de-Somas e Soma-de-Produtos Usando o Mapa de Karnaugh

Quando uma expressão de produto-de-somas é inserida no mapa, ela pode ser facilmente convertida para a forma de soma-de-produtos equivalente diretamente a partir do mapa de Karnaugh. Além disso, dada uma expressão de soma-de-produtos já inserida no mapa, uma expressão de produto-de-somas equivalente pode ser obtida diretamente a partir do mapa. Essa é uma boa forma de comparar as duas formas mínimas de uma expressão para determinar se uma delas pode ser implementada com menos portas que a outra.

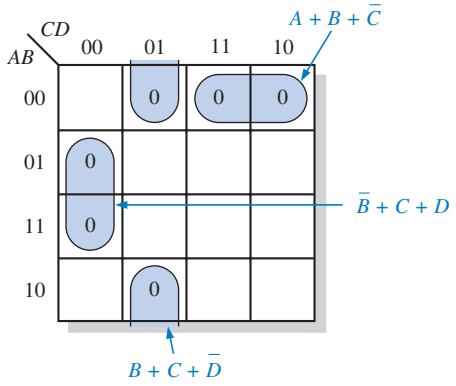
Para uma expressão de produto-de-somas, todas as células que não contêm 0s contêm 1s, a partir dos quais a expressão de soma-de-produtos é obtida. Igualmente, para uma expressão de soma-de-produtos, todas as células que não contêm 1s contêm 0s, a partir dos quais a expressão de produto-de-somas é obtida. O Exemplo 4–33 ilustra essa conversão.

EXEMPLO 4–33

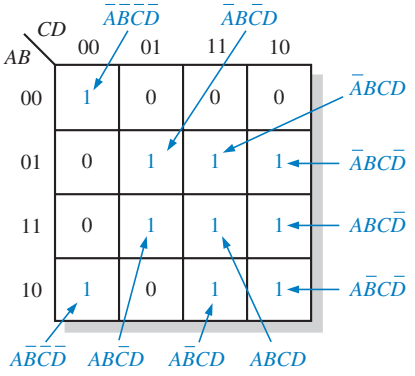
Usando um mapa de Karnaugh, converta a seguinte expressão de produto-de-somas padrão numa expressão de produto-de-somas mínima, numa expressão de soma-de-produtos padrão e numa expressão de soma-de-produtos mínima.

$$(\overline{A} + \overline{B} + C + D)(A + \overline{B} + C + D)(A + B + C + \overline{D})$$
$$(A + B + \overline{C} + \overline{D})(\overline{A} + B + C + \overline{D})(A + B + \overline{C} + D)$$

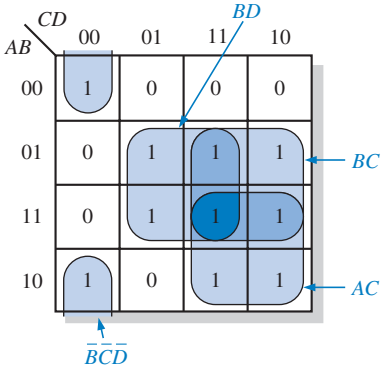
Solução Os 0s para a expressão de produto-de-somas padrão são inseridos no mapa e agrupados para obter a expressão de produto-de-somas mínima na Figura 4–41(a). Na Figura 4–41(b), 1s são inseridos nas células que não contêm 0s. A partir de cada célula que contém um 1, um termo-produto padrão é obtido conforme indicado. Esses termos-produto formam a expressão de soma-de-produtos padrão. Na Figura 4–41(c), os 1s são agrupados e a expressão de soma-de-produtos mínima é obtida.



(a) Produto-de-somas mínimo:
$(A + B + C)(\overline{B} + \overline{C} + D)(B + C + \overline{D})$



(b) Soma-de-produtos padrão:
$\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} + AB\overline{C}\overline{D} + A\overline{B}C\overline{D} + AB\overline{C}D + A\overline{B}CD + ABCD$



(c) Soma-de-produtos mínima: $AC + BC + BD + \overline{B}\overline{C}\overline{D}$

▲ **FIGURA 4–41**

Problema relacionado Use um mapa de Karnaugh para converter a seguinte expressão para a forma de soma-de-produtos mínima:

$$(W + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})(\overline{W} + \overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + \overline{Z})$$

SEÇÃO 4–40 REVISÃO

1. Qual é a diferença no preenchimento de um mapa de Karnaugh com uma expressão de produto-de-somas e uma expressão de soma-de-produtos?
2. Qual o termo-soma padrão expresso com as variáveis *A*, *B*, *C* e *D* para se ter um 0 na célula 1011 do mapa de Karnaugh?
3. Qual é o termo-produto padrão expresso com as variáveis *A*, *B*, *C* e *D* para se ter um 1 na célula 0010 no mapa de Karnaugh?

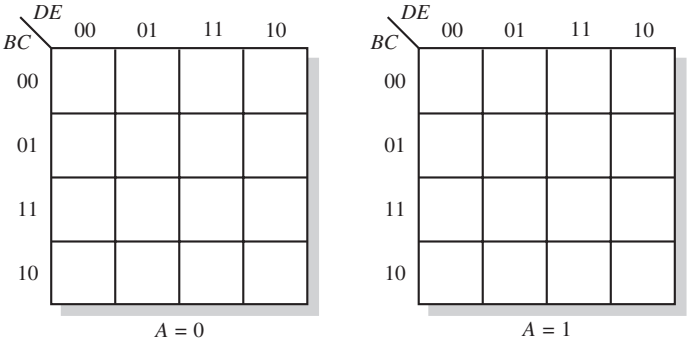
4-11 MAPAS DE KARNAUGH DE CINCO VARIÁVEIS

As funções Booleanas com cinco variáveis podem ser simplificadas usando um mapa de Karnaugh de 32 células. Na realidade são usados dois mapas de 4 variáveis (16 células cada) para construir um mapa de 5 variáveis. Já sabemos como é a adjacência entre células num mapa de 4 variáveis e como formar grupos de células contendo 1s para simplificar expressões de soma-de-produtos. Tudo o que precisamos saber para cinco variáveis é a adjacência das células entre os dois mapas de 4 variáveis e como agrupar os 1s adjacentes.

Ao final do estudo desta seção você deverá ser capaz de:

■ Determinar a adjacência entre células num mapa de 5 variáveis ■ Formar agrupamentos máximos de células num mapa de 5 variáveis ■ Minimizar expressões Booleanas de 5 variáveis usando o mapa de Karnaugh

Um mapa de Karnaugh para cinco variáveis (*ABCDE*) pode ser construído usando dois mapas de 4 variáveis com os quais já estamos familiarizados. Cada mapa contém 16 células com todas as combinações das variáveis *A, B, C, D* e *E*. Um mapa é para $A = 0$ e o outro é para $A = 1$, conforme mostra a Figura 4–42.

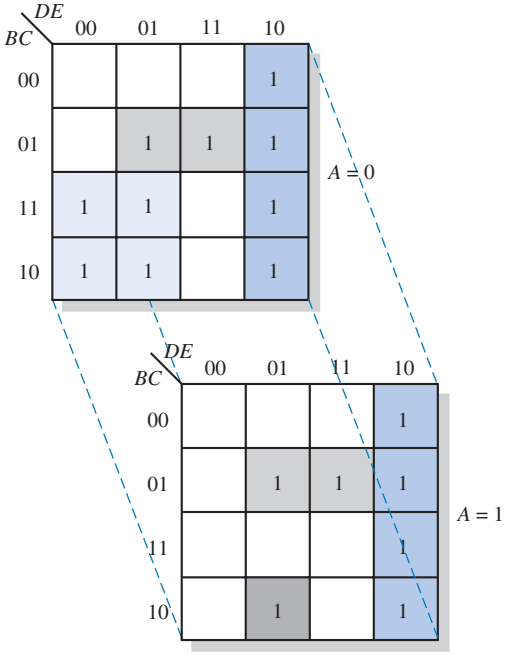


► **FIGURA 4–42**
Um mapa de Karnaugh para 5 variáveis.

Células Adjacentes

Já sabemos como determinar células adjacentes num mapa de 4 variáveis. A melhor forma de visualizar células adjacentes entre os dois mapas de 16 células é imaginar que o mapa para $A = 0$ é posto sobre o mapa para $A = 1$. Cada célula no primeiro mapa é adjacente à célula diretamente abaixo no segundo mapa.

Para ilustrar, um exemplo com quatro grupos é mostrado na Figura 4–43 com os mapas num arranjo tridimensional. Os 1s nas células em amarelo formam um grupo de 8 bits (quatro no ma-



► **FIGURA 4–43**
Ilustração do agrupamento de 1s em células adjacentes num mapa de 5 variáveis.

pa para $A = 0$ combinadas com as quatro no mapa para $A = 1$). Os 1s nas células em laranja formam um grupo de 4 bits. Os 1s nas células em laranja claro formam um grupo de 4 apenas no mapa para $A = 0$. O 1 na célula em cinza escuro no mapa para $A = 1$ é agrupado com o 1 na célula adjacente em laranja claro no mapa para $A = 0$ formando um grupo de 2 bits.

Determinação da Expressão Booleana A expressão Booleana de soma-de-produtos original que é inserida no mapa de Karnaugh da Figura 4–43 contém dezessete termos de 5 variáveis porque existem dezessete 1s no mapa. Como sabemos, apenas as variáveis que não mudam de complementada para não complementada ou vice-versa dentro de um grupo permanecem na expressão para aquele grupo. A expressão simplificada obtida do mapa é desenvolvida como mostrado a seguir:

- O termo para o grupo em laranja é $D\overline{E}$.
- O termo para o grupo em cinza claro é $\overline{B}CE$.
- O termo para o grupo em laranja claro é $\overline{A}B\overline{D}$.
- O termo para o grupo formado pela célula em cinza escuro com a célula em laranja claro é $B\overline{C}\,\overline{D}E$.

A combinação desses termos numa expressão de soma-de-produtos resulta em:

$$X = D\overline{E} + \overline{B}CE + \overline{A}B\overline{D} + B\overline{C}\,\overline{D}E$$

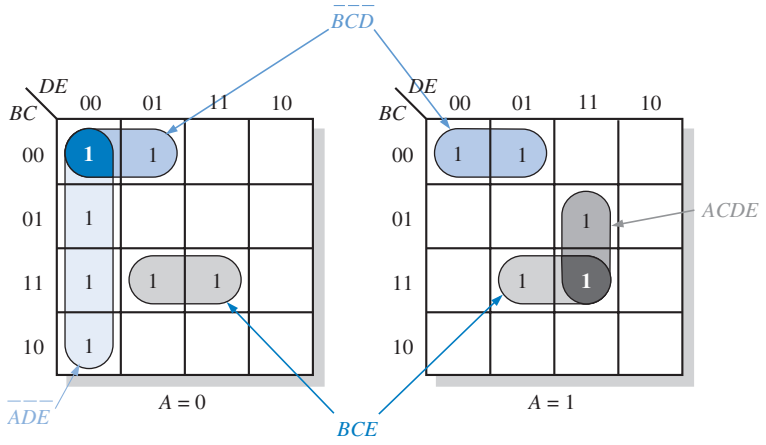
EXEMPLO 4–34

Use um mapa de Karnaugh para minimizar a seguinte expressão de soma-de-produtos padrão de 5 variáveis:

$$X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}BC\overline{D}\,\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}\,\overline{C}D\overline{E} + \overline{A}BCD\overline{E} + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}\,\overline{C}D\overline{E} + ABC\overline{D}\,\overline{E} + ABCDE + A\overline{B}CDE$$

Solução Insira no mapa a expressão de soma-de-produtos. A Figura 4–44 mostra os agrupamentos e os termos correspondentes. Combinando os termos temos a seguinte expressão de soma-de-produtos minimizada:

$$X + \overline{A}D\overline{E} + \overline{B}\,\overline{C}D + BCE + ACDE$$



▶ FIGURA 4–44

Problema relacionado Minimize a seguinte expressão:

$$Y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}BC\overline{D}\,\overline{E} + \overline{A}B\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + A\overline{B}C\overline{D}\,\overline{E} + AB\overline{C}\,\overline{D}\,\overline{E} + AB\overline{C}\,\overline{D}E + \overline{A}\,\overline{B}\,\overline{C}D\overline{E} + \overline{A}\,\overline{B}CD\overline{E} + \overline{A}B\overline{C}D\overline{E} + A\overline{B}\,\overline{C}D\overline{E} + A\overline{B}CD\overline{E} + AB\overline{C}D\overline{E}$$

SEÇÃO 4–11 REVISÃO

1. Por que um mapa de Karnaugh de 5 variáveis requer 32 células?
2. Qual é a expressão representada por um mapa de Karnaugh de 5 variáveis no qual cada célula contém um 1?

4-12 VHDL (Opcional)

Essa seção opcional provê uma breve introdução em VHDL, sendo que não estudaremos a estrutura completa e a sintaxe da linguagem. Para informações e instruções mais detalhadas consulte a nota de rodapé nessa seção. Linguagens de descrição de hardware (HDLs – *hardware description languages*) são ferramentas para inserção de projetos de dispositivos lógicos. Embora o VHDL proporcione múltiplas formas de descrever um circuito lógico, discutiremos aqui apenas os mais simples e diretos exemplos de inserção via texto.

Ao final do estudo desta seção você deverá ser capaz de:

- Enunciar os elementos essenciais do VHDL
- Escrever um programa simples em VHDL

O V em VHDL* quer dizer circuito integrado de altíssima velocidade (VHSIC – *V*ery *H*igh *S*peed *I*ntegrated *C*ircuit) e o HDL, é claro, significa linguagem de descrição de hardware. Conforme mencionado, **VHDL** é uma linguagem padrão adotada pelo Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE – *Institute of Electrical and Electronics Engineers*) e é designado como o padrão 1076-1993 do IEEE. VHDL é uma linguagem complexa e compreensiva sendo que o uso de todo o seu potencial envolve um grande esforço e experiência.

O VHDL provê três abordagens básicas para a descrição de um circuito digital usando software: *procedimental*, *fluxo de dados* e *estrutural*. Restringiremos-nos a discutir a abordagem de fluxo de dados na qual escrevemos declarações do tipo Booleanas para descrever um circuito lógico. Tenha em mente que VHDL, bem como as outras HDLs, é uma ferramenta para implementação de projetos digitais sendo portanto um meio para se atingir um fim e não propriamente um fim.

É relativamente fácil escrever programas para descrever circuitos lógicos simples em VHDL. Os operadores lógicos são as seguintes palavras reservadas em VHDL: **and, or, not, nand, nor, xor** e **xnor**. Os dois elementos essenciais em qualquer programa VHDL são a entidade e a arquitetura, sendo que eles têm que ser usados juntos. A **entidade** descreve uma dada função lógica em termos das suas entradas e saídas externas, que são chamadas de ports. A **arquitetura** descreve a operação interna da função lógica.

Na sua forma mais simples, o elemento entidade consiste de três declarações: A primeira declaração associa um nome à função lógica; a segunda, denominada de declaração port que é recuada, especifica as entradas e saídas; e a terceira é a declaração end. Embora o leitor não tenha certamente escrito um programa VHDL para uma única porta lógica, é uma forma didática começar um simples exemplo tal como uma porta AND. A declaração de entidade VHDL para uma porta AND de 2 entradas é a seguinte:

Vírgulas e ponto-e-vírgulas têm que ser usados apropriadamente em todos os programas VHDL.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;
```

Os termos em negrito e em cor são palavras reservadas em VHDL; os outros termos são identificadores que associamos; e os parênteses, vírgulas e ponto-e-vírgulas fazem parte da sintaxe necessária em VHDL. Como podemos ver *A* e *B* foram especificados como os bits de entrada e *X* foi especificado como o bit de saída. Os identificadores do port *A*, *B* e *X* bem como o nome da entidade AND_Gate2 são definidos pelo usuário e podem ser renomeados. Assim como em todos os HDLs, o uso de vírgulas e ponto-e-vírgulas é importante e tem que ser estritamente seguido (faz parte da sintaxe da linguagem).

* Veja Floyd, Thomas. 2003, *Digital Fundamentals with VHDL*, Prentice Hall; Pellerin, David and Taylor, Douglas, 1997. VHDL Made Easy! Prentice Hall; Bhasker, Jayaram, 1999. *A VHDL Primer*, 3 ed. Prentice Hall.

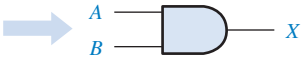
O elemento arquitetura VHDL do programa para a porta AND de 2 entradas descrito pela entidade é:

```
architecture LogicFunction of AND_Gate2 is
begin
  X ⇐ A and B;
end architecture LogicFunction;
```

Novamente, as palavras reservadas VHDL estão em negrito e em cor, e os pontos-e-vírgulas e o operador associado ⇐ representam uma sintaxe necessária. A primeira declaração do elemento arquitetura tem que se referenciar ao nome da entidade.

A entidade e a arquitetura são combinadas num único programa VHDL para descrever uma porta AND, conforme ilustrado na Figura 4–45.

```
entity AND_Gate2 is
  port (A, B: in bit; X: out bit);
end entity AND_Gate2;

architecture LogicFunction of AND_Gate2 is
begin
  X <= A and B;
end architecture LogicFunction;
```

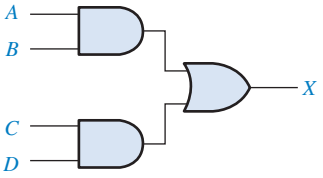


◀ **FIGURA 4–45**
Um programa VHDL para uma porta AND de 2 entradas.

Escrita de Expressões Booleanas em VHDL Como vimos, a expressão para uma porta AND de 2 entradas, $X = AB$, é escrita em VHDL como X ⇐ A **and** B; qualquer expressão Booleana pode ser escrita usando as palavras reservadas em VHDL **not, and, or, nand, nor, xor** e **xnor**. Por exemplo, a expressão Booleana $X = A + B + C$ é escrita em VHDL como X ⇐ A **or** B **or** C;. A expressão Booleana $X = A\overline{B} + \overline{C}D$ pode ser escrita em VHDL como X ⇐ (A **and not** B) **or** (**not** C **and** D);. Como um outro exemplo, a declaração para uma porta NAND de 2 entradas em VHDL pode ser escrita como X ⇐ **not**(A **and** B); ou ainda pode ser escrita como X ⇐ A **nand** B;.

EXEMPLO 4–35

Escreva um programa em VHDL para descrever o circuito lógico mostrado na Figura 4–46.



▶ **FIGURA 4–46**

Solução Esse circuito lógico AND/OR é descrito em álgebra Booleana como:

$$X = AB + CD$$

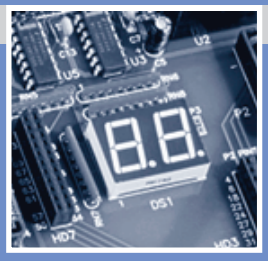
O programa VHDL é mostrado a seguir. O nome da entidade é AND_OR.

```
entity AND_OR is
  port (AA, B, C, D: in bit;X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
  X <= (A and B) or (C and D);
end architecture LogicFunction;
```

Problema relacionado Escreva a declaração VHDL para descrever o circuito lógico desse exemplo se uma porta NOR substituir a porta OR na Figura 4–46.

SEÇÃO 4–12 REVISÃO

1. O que é HDL?
2. Cite dois elementos de projeto essenciais num programa VHDL.
3. O que faz a entidade (entity)?
4. O que faz a arquitetura (architecture)?



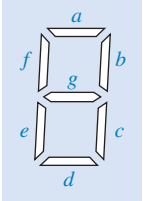
APLICAÇÕES EM SISTEMAS DIGITAIS

Os displays de sete segmentos são usados em vários tipos de produtos. O sistema de controle e contagem de comprimidos que foi descrito no Capítulo 1 tem dois displays de 7 segmentos. Esses displays são usados com circuitos lógicos que decodificam um número codificado em binário (BCD) e ativam os dígitos apropriados nos displays. Nesta seção de aplicações de sistemas digitais damos enfoque no projeto implementado com menor número de portas para ilustrar uma aplicação de expressões Booleanas e mapa de Karnaugh. Como opção, VHDL também é aplicado.

O Display de 7 Segmentos

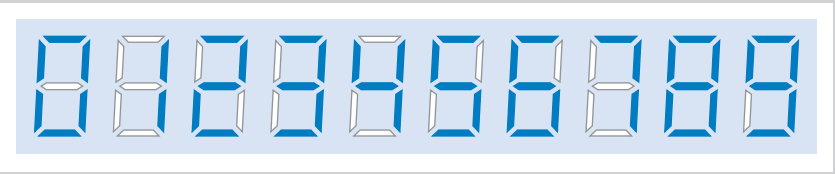
A Figura 4–47 mostra um formato de display comum composto de sete elementos ou segmentos. Ao energizar certas combinações de segmentos, podemos construir cada um dos dez dígitos no display. A Figura 4–48 ilustra esse método de display digital para cada um dos dez dígitos usando um segmento laranja para representar um segmento energizado. Para produzir um 1, os segmentos b e c são energizados. Para produzir um 2 os segmentos a, b, g, e e d são usados; e assim por diante.

Displays de LEDs Um tipo comum de display de 7 segmentos consiste em diodos emissores de luz (LEDs – *light-emitting diodes*) arranjados como mostra a Figura 4–49. Cada segmento é um LED que emite luz quando existe uma corrente passando através dele. Na Figura 4–49(a) o arranjo tipo anodo comum requer que o circuito de acionamento proporcione uma tensão de nível BAIXO para ativar um dado segmento. Quando um nível BAIXO é aplicado na entrada de um segmento, o LED é ligado, pois existe uma corrente através dele. Na Figura 4–49(b) o arranjo tipo catodo comum requer que o acionador proporcione uma tensão de nível ALTO para ativar um segmento. Quando um nível ALTO é aplicado na entrada de um segmento, o LED é ligado, pois existe uma corrente através dele.

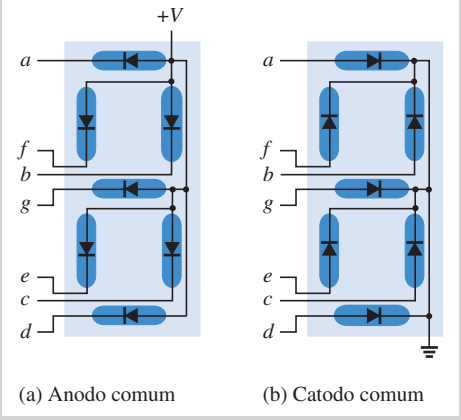


▲ **FIGURA 4–47**
Formato de um display de 7 segmentos mostrando o arranjo dos segmentos.

Displays de LCD Um outro tipo comum de display de 7 segmentos é o display de cristal líquido (LCD – *liquid crystal display*). Os LCDs operam por polarização da luz de forma que um segmento desativado reflete a luz incidente mostrando-se invisível em relação ao fundo (plano posterior que reflete a luz de volta). Um segmento ativo



▲ **FIGURA 4–48**
Visualização dos dígitos decimais com um dispositivo de 7 segmentos.



(a) Anodo comum (b) Catodo comum

◀ **FIGURA 4–49**
Arranjos de um display de LED de 7 segmentos.

não reflete a luz incidente mostrando-se escuro. Os LCDs consomem muito menos potência que os LEDs porém não podem ser vistos no escuro, enquanto que os displays de LEDs podem ser vistos.

Lógica dos Segmentos

Cada segmento é usado para vários dígitos decimais, porém nenhum segmento é usado em todos os dez dígitos. Portanto, cada segmento deve ser ativado pelo seu próprio circuito de decodificação que detecta a ocorrência de qualquer um dos números no qual o segmento é usado. A partir das Figuras 4–47 e 4–48, os segmentos que são necessários serem ativados para cada dígito mostrado são determinados e listados na Tabela 4–9.

Tabela-verdade para a Lógica de Segmentos A lógica de decodificação de segmentos requer quatro entradas BCD e sete saídas, uma para cada segmento no display, conforme indicado no diagrama em bloco da Figura 4–50. A tabela-verdade de múltiplas saídas, mostrada na Tabela 4–10, é na realidade sete tabelas-verdade numa só tabela, sendo que poderiam ser separadas (uma tabela para cada segmento). Um 1 nas colunas de saídas dos segmentos na tabela indica um segmento ativado.

Como o código BCD não inclui os valores binários 1010, 1011, 1100, 1101, 1110 e 1111, essas combinações nunca aparecem nas entradas e, portanto, podem ser tratados como condições "don't care" (X), conforme indicado na tabela-verdade. Para estar de acordo com o que é praticado pela maioria dos fabricantes de CIs, a letra A representa o bit menos significativo e D representa o bit mais significativo nessa aplicação em particular.

Expressões Booleanas para a Lógica de Segmentos A partir da tabela-verdade, uma expressão de soma-de-produtos padrão ou de produto-de-somas padrão pode ser escrita para cada segmento. Por exemplo, a expressão de soma-de-produtos padrão para o segmento a é

$$a = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}B\overline{A}$$

e a expressão de soma-de-produtos padrão para o segmento e é

$$e = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}CB\overline{A} + D\overline{C}\,\overline{B}\,\overline{A}$$

As expressões para os outros segmentos podem ser desenvolvidas de forma similar. Como podemos ver, a expressão para o segmento a tem oito termos-produto e a expressão para o segmento e tem quatro termos-produto representando cada uma das entradas BCD que ativa aquele segmento. Isso significa que a implementação da expressão de soma-de-produtos padrão da lógica do segmento a requer um circuito AND-OR que consiste de oito portas AND de 4 entradas e uma porta OR de 8 entradas. A implementação da lógica do segmento e requer quatro portas AND de 4 entradas e uma porta OR de 4 entradas. Em ambos os casos, são necessários quatro inversores para produzir o complemento de cada variável.

Minimização por Mapa da Karnaugh da Lógica de Segmentos Vamos começar obtendo a expressão de soma-de-produtos mínima para o segmento a. Um mapa de Karnaugh para o segmento a é mostrado na Figura 4–51 e os passos a seguir executados:

Passo 1 Os 1s são inseridos no mapa diretamente a partir da Tabela 4–10.

TABELA 4–9

Segmentos ativos para cada dígito decimal

DÍGITO	SEGMENTOS ATIVADOS
0	*a, b, c, d, e, f*
1	*b, c*
2	*a, b, d, e, g*
3	*a, b, c, d, g*
4	*b, c, f, g*
5	*a, c, d, f, g*
6	*a, c, d, e, f, g*
7	*a, b, c*
8	*a, b, c, d, e, f, g*
9	*a, b, c, d, f, g*

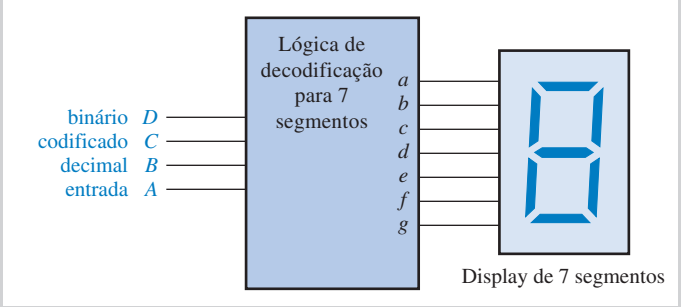


FIGURA 4–50

Diagrama em bloco da lógica de 7 segmentos com o display.

TABELA 4–10

Tabela-verdade para a lógica de 7 segmentos

DÍGITO DECIMAL	ENTRADAS				SAÍDAS DOS SEGMENTOS						
	D	*C*	*B*	*A*	*a*	*b*	*c*	*d*	e	*f*	*g*
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Saída = 1 significa segmento ativado (lig.)

Saída = 0 significa segmento desativado (deslig.)

Saída = X significa "don't care"

Passo 2 Todas as condições "don't care" (*X*) são colocadas no mapa.

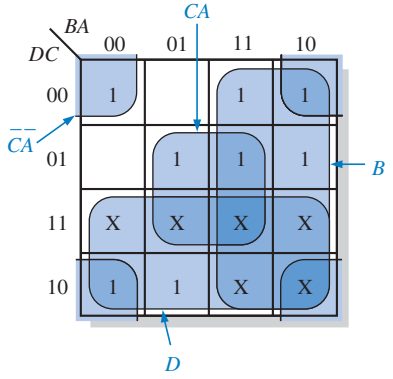
Passo 3 Os 1s são agrupados conforme mostrado. Os "don't cares" e as sobreposições de células são usadas para formar os maiores grupos possíveis.

Passo 4 Escreva o termo-produto mínimo para cada grupo e some os termos para formar a expressão de soma-de-produtos mínima.

Tenha em mente que "don't cares" não têm que ser incluídos num grupo, mas nesse caso todos eles foram usados. Além disso, observe que os 1s nas células situadas nos cantos são agrupados com um "don't care" usando o conceito de células adjacentes (enrolando o mapa tanto na vertical quanto na horizontal).

Expressão padrão de soma-de-produtos

$\overline{D}\overline{C}\overline{B}\overline{A} + \overline{D}\overline{C}\overline{B}A + \overline{D}C\overline{B}\overline{A} + \overline{D}C\overline{B}A + D\overline{C}\overline{B}\overline{A} + D\overline{C}\overline{B}A + DC\overline{B}\overline{A} + DC\overline{B}A$



Expressão mínima de soma-de-produtos: $D + B + CA + \overline{C}\overline{A}$

FIGURA 4–51

Minimização via mapa de Karnaugh da expressão lógica do segmento *a*.

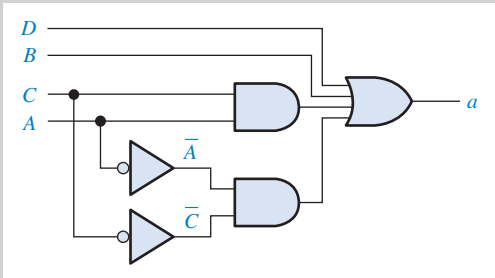


FIGURA 4–52

A implementação lógica mínima para o segmento *a* do display de 7 segmentos.

Implementação Mínima da Lógica de Segmentos

A expressão de soma-de-produtos mínima obtida do mapa de Karnaugh da Figura 4–52 para a lógica do segmento *a* é

$$D + B + CA + \overline{C}\,\overline{A}$$

Essa expressão pode ser implementada com duas portas AND de 2 entradas, uma porta OR de 4 entradas e dois inversores conforme mostra a Figura 4–52. Compare essa expressão com a implementação de soma-de-produtos padrão para a lógica do segmento *a* discutida anteriormente; veremos que o número de portas e inversores foi reduzido de treze para cinco e, como resultado, o número de interconexões foi significativamente reduzido.

A lógica mínima para cada um dos seis segmentos restantes (*b*, *c*, *d*, *e*, *f* e *g*) pode ser a partir de uma abordagem similar.

Implementação VHDL (opcional)

Todas as lógicas de segmentos podem ser descritas por VHDL para implementação num dispositivo lógico programável. A lógica do segmento *a* pode ser descrita pelo seguinte programa VHDL:

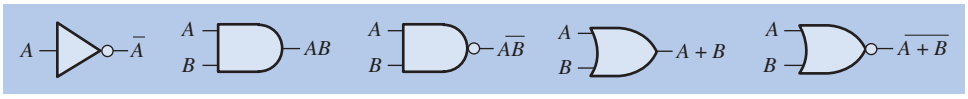
```
entity SEGLOGIC is
    port (A, B, C: in bit; SEGa: out bit);
end entity SEGLOGIC;
architecture LogicFunction of
    SEGLOGIC is
begin
    SEGa <= (A and C) or (not A
            and not C) or B or D;

end architecture LogicFunction;
```

Tarefas Propostas no Sistema

- *Atividade 1*: Determine a lógica mínima para o segmento *b*.
- *Atividade 2*: Determine a lógica mínima para o segmento *c*.
- *Atividade 3*: Determine a lógica mínima para o segmento *d*.
- *Atividade 4*: Determine a lógica mínima para o segmento *e*.
- *Atividade 5*: Determine a lógica mínima para o segmento *f*.
- *Atividade 6*: Determine a lógica mínima para o segmento *g*.
- *Atividade opcional*: Complete o programa VHDL para todos os segmentos incluindo cada descrição lógica de segmento na arquitetura.

RESUMO

- Os símbolos das portas e as expressões Booleanas para as saídas de um inversor e de portas de 2 entradas são mostradas na Figura 4–53.



▲ **FIGURA 4–53**

- Leis comutativas: $A + B = B + A$
  $AB = BA$
- Leis associativas: $A + (B + C) = (A + B) + C$
  $A(BC) = (AB)C$
- Lei distributiva: $A(B + C) = AB + AC$
- Regras Booleanas:

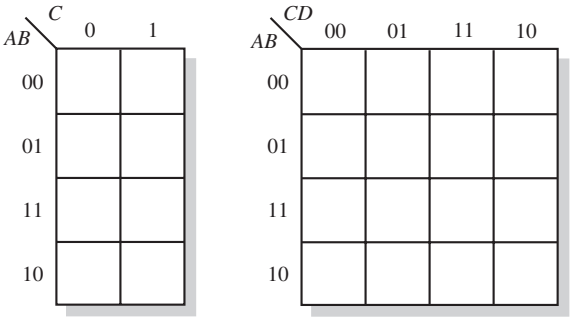
1.	$A + 0 = A$	**7.**	$A \cdot A = A$
2.	$A + 1 = 1$	**8.**	$A \cdot \overline{A} = 0$
3.	$A \cdot 0 = 0$	**9.**	$\overline{\overline{A}} = A$
4.	$A \cdot 1 = A$	**10.**	$A + AB = A$
5.	$A + A = A$	**11.**	$A + \overline{A}B = A + B$
6.	$A + \overline{A} = 1$	**12.**	$(A + B)(A + C) = A + BC$

- Teoremas de DeMorgan:

1. O complemento de um produto é igual à soma dos complementos dos termos do produto, $\overline{XY} = \overline{X} + \overline{Y}$

2. O complemento de uma soma é igual ao produto dos complementos dos termos da soma.
$\overline{X + Y} = \overline{X}\,\overline{Y}$

- Os mapas de Karnaugh de 3 e 4 variáveis são mostrados na Figura 4–54. Um mapa de 5 variáveis é formado a partir de dois mapas de 4 variáveis.



▶ **FIGURA 4–54**

- O elemento básico de projeto em VHDL é o par entidade/arquitetura.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Complemento O inverso ou o oposto de um número. Em álgebra Booleana, a função inversa é expressa com uma barra sobre a variável.

"Don't care" Uma combinação de literais que não podem ocorrer e podem ser usadas como um 1 ou um 0 no mapa de Karnaugh para simplificação.

Mapa de Karnaugh Um arranjo de células que representam as combinações de literais numa expressão Booleana e usado para uma simplificação sistemática dessa expressão.

Minimização O processo que resulta numa expressão Booleana de soma-de-produtos ou num produto-de-somas que contém o menor número de literais por termo.

Produto-de-somas Uma forma de expressão Booleana que é basicamente a operação AND de termo OR.

Soma-de-produtos Uma forma de expressão Booleana que é basicamente a operação OR de termos AND.

Termo-produto O produto Booleano de dois ou mais literais equivalente a uma operação AND.

Termo-soma A soma Booleana de dois ou mais literais equivalentes a uma operação OR.

Variável Um símbolo usado para representar uma grandeza lógica que pode ter um valor 1 ou 0, geralmente designada por uma letra em itálico.

VHDL Uma linguagem de descrição de hardware padrão. Padrão 1076-1993 da IEEE.

AUTOTESTE

As respostas estão no final do capítulo.

1. O complemento de uma variável é sempre
(a) 0 **(b)** 1 **(c)** igual à variável **(d)** o inverso da variável
2. A expressão Booleana $A + \overline{B} + C$ é
(a) um termo-soma **(b)** um termo literal **(c)** um termo-produto **(d)** um termo complementado
3. A expressão Booleana $A\overline{B}C\overline{D}$ é
(a) um termo-soma **(b)** um termo literal **(c)** um termo-produto **(d)** sempre 1

4. O domínio da expressão $A\overline{B}CD + A\overline{B} + \overline{C}D + B$ é
(a) A e D **(b)** apenas B **(c)** A, B, C e D **(d)** nenhuma das alternativas anteriores

5. De acordo com a lei comutativa da adição,
(a) $AB = BA$ **(b)** $A = A + A$
(c) $A + (B + C) = (A + B) + C$ **(d)** $A + B = B + A$

6. De acordo com a lei associativa da multiplicação,
(a) $B = BB$ **(b)** $A(BC) = (AB)C$ **(c)** $A + B = B + A$ **(d)** $B + B(B + 0)$

7. De acordo com a lei distributiva,
(a) $A(B + C) = AB + AC$ **(b)** $A(BC) = ABC$ **(c)** $A(A + 1) = A$ **(d)** $A + AB = A$

8. Qual das seguintes alternativas não é uma regra válida da álgebra Booleana?
(a) $A + 1 = 1$ **(b)** $A = \overline{A}$ **(c)** $AA = A$ **(d)** $A + 0 = A$

9. Qual das seguintes regras diz que se uma entrada de uma porta AND for sempre 1, a saída é igual a outra entrada?
(a) $A + 1 = 1$ **(b)** $A + A = A$ **(c)** $A \cdot A = A$ **(d)** $A \cdot 1 = A$

10. De acordo com os teoremas de DeMorgan, a(s) seguinte(s) igualdade(s) está(ão) correta(s):
(a) $\overline{AB} = \overline{A} + \overline{B}$ **(b)** $\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}$
(c) $\overline{A + B + C} = \overline{A}\,\overline{B}\,\overline{C}$ **(d)** todos os itens estão corretos

11. A expressão Booleana $X = AB + CD$ representa
(a) uma AND entre as saídas de duas portas OR. **(b)** uma porta AND de 4 entradas.
(c) uma OR entre as saídas de duas portas AND. **(d)** uma EX-OR.

12. Um exemplo de uma expressão de soma-de-produtos é
(a) $A + B(C + D)$ **(b)** $\overline{A}B + A\overline{C} + A\overline{B}C$
(c) $(\overline{A} + B + C)(A + \overline{B} + C)$ **(d)** as alternativas (a) e (b) estão corretas.

13. Um exemplo de uma expressão de produto-de-somas é
(a) $A(B + C) + A\overline{C}$ **(b)** $(A + B)(\overline{A} + B + \overline{C})$
(c) $\overline{A} + \overline{B} + BC$ **(d)** os itens (a) e (b) estão corretos

14. Um exemplo de uma expressão de soma-de-produtos padrão é
(a) $\overline{A}B + A\overline{B}C + AB\overline{D}$ **(b)** $A\overline{B}C + A\overline{C}D$
(c) $A\overline{B} + \overline{A}B + AB$ **(d)** $A\overline{B}C\overline{D} + \overline{A}B + \overline{A}$

15. Um mapa de Karnaugh de 3 variáveis tem
(a) oito células **(b)** três células
(c) dezesseis células **(d)** quatro células

16. Em um mapa de Karnaugh de 4 variáveis, um termo-produto de 2 variáveis é produzido por
(a) um grupo de 2 células de 1s **(b)** um grupo e 8 células de 1s
(c) um grupo de 4 células de 1s **(d)** um grupo de 4 células de 0s

17. Em um mapa de Karnaugh, o agrupamento de 0s produz
(a) uma expressão de produto-de-somas. **(b)** uma expressão de soma-de-produtos.
(c) uma condição "don't care". **(d)** uma lógica AND-OR.

18. Um mapa de Karnaugh de 5 variáveis tem
(a) dezesseis células **(b)** trinta e duas células **(c)** sessenta e quatro células

19. Um SPLD que tem um arranjo AND programável e um arranjo OR fixo é um dispositivo
(a) PROM **(b)** PLA **(c)** PAL **(d)** GAL

20. VHDL é um tipo de
(a) lógica programável **(b)** linguagem de descrição de hardware
(c) arranjo programável **(d)** matemática lógica

21. Em VHDL, um port é
(a) um tipo de entidade **(b)** um tipo de arquitetura
(c) uma entrada ou saída **(d)** um tipo de variável

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 4–1 Operações e Expressões Booleanas

1. Usando a notação Booleana, escreva uma expressão que seja 1 sempre que uma ou mais de suas variáveis (*A, B, C* e *D*) sejam 1s.

2. Escreva uma expressão que seja 1 apenas se todas as suas variáveis (*A, B, C, D* e *E*) forem 1s.

3. Escreva uma expressão que seja 1 apenas quando uma ou mais de suas variáveis (*A*, *B* e *C*) forem 0.

4. Avalie as seguintes operações:

(a) $0+0+1$ **(b)** $1+1+1$ **(c)** $1\cdot 0\cdot 0$

(d) $1\cdot 1\cdot 1$ **(e)** $1\cdot 0\cdot 1$ **(f)** $1\cdot 1+0\cdot 1\cdot 1$

5. Determine os valores das variáveis que tornam cada termo-produto 1 e cada termo-soma 0.

(a) AB **(b)** $A\overline{B}C$ **(c)** $A+B$ **(d)** $\overline{A}+B+\overline{C}$

(e) $\overline{A}+\overline{B}+C$ **(f)** $\overline{A}+B$ **(g)** $A\overline{B}\,\overline{C}$

6. Determine o valor de *X* para todos os valores possíveis das variáveis.

(a) $X=(A+B)C+B$ **(b)** $X=(\overline{A+B})C$ **(c)** $X=A\overline{B}C+AB$

(d) $X=(A+B)(\overline{A}+B)$ **(e)** $X=(A+BC)(\overline{B}+\overline{C})$

SEÇÃO 4–2 Leis e Regras da Álgebra Booleana

7. Identifique a lei da álgebra Booleana na qual cada uma das seguintes equações se baseia:

(a) $A\overline{B}+CD+A\overline{C}D+B=B+A\overline{B}+A\overline{C}D+CD$

(b) $AB\overline{C}D+\overline{ABC}=D\overline{C}BA+\overline{CBA}$

(c) $AB(CD+E\overline{F}+GH)=ABCD+ABE\overline{F}+ABGH$

8. Identifique a(s) regra(s) Booleana(s) na(s) qual(is) cada uma da igualdades se baseia:

(a) $\overline{\overline{AB+CD}}+\overline{EF}=AB+CD+\overline{EF}$ **(b)** $A\overline{A}B+AB\overline{C}+AB\overline{B}=AB\overline{C}$

(c) $A(BC+BC)+AC=A(BC)+AC$ **(d)** $AB(C+\overline{C})+AC=AB+AC$

(e) $A\overline{B}+A\overline{B}C=A\overline{B}$ **(f)** $ABC+\overline{AB}+\overline{ABC}D=ABC+\overline{AB}+D$

SEÇÃO 4–3 Teoremas de DeMorgan

9. Aplique os teoremas de DeMorgan em cada expressão a seguir:

(a) $\overline{A+\overline{B}}$ **(b)** $\overline{\overline{A}B}$ **(c)** $\overline{A+B+C}$ **(d)** $\overline{ABC}$

(e) $\overline{A(B+C)}$ **(f)** $\overline{AB}+\overline{CD}$ **(g)** $\overline{AB+CD}$ **(h)** $\overline{(A+\overline{B})(\overline{C}+D)}$

10. Aplique os teoremas de DeMorgan em cada expressão a seguir:

(a) $\overline{A\overline{B}(C+\overline{D})}$ **(b)** $\overline{AB(CD+EF)}$

(c) $\overline{(A+\overline{B}+C+\overline{D})}+\overline{ABC\overline{D}}$ **(d)** $\overline{(\overline{A}+B+C+D)(A\overline{B}\,\overline{CD})}$

(e) $\overline{A\overline{B}(CD+\overline{EF})(\overline{AB}+\overline{CD})}$

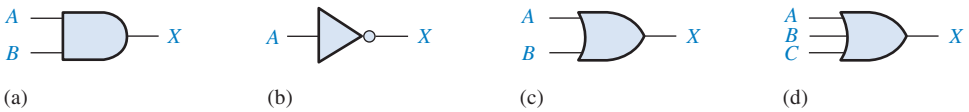
11. Aplique os teoremas de DeMorgan nas seguintes expressões:

(a) $\overline{\overline{(ABC)}(\overline{EFG})}+\overline{(\overline{HIJ})(\overline{KLM})}$ **(b)** $\overline{(A+\overline{B\overline{C}}+CD)}+\overline{\overline{BC}}$

(c) $\overline{\overline{(A+B)}(\overline{C+D})(\overline{E+F})(\overline{G+H})}$

SEÇÃO 4–4 Análise Booleana de Circuitos Lógicos

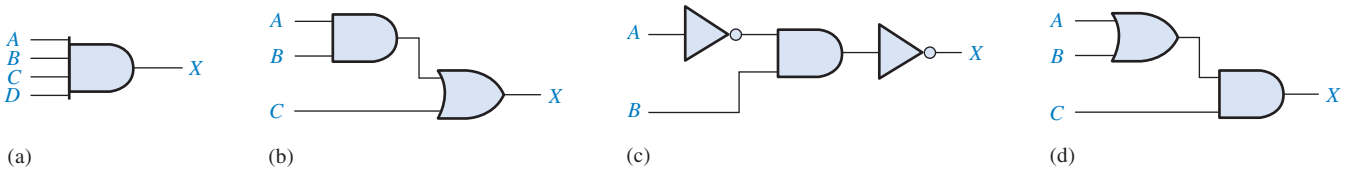
12. Escreva a expressão Booleana para cada uma das portas lógicas mostradas na Figura 4–55.



(a) (b) (c) (d)

▶ FIGURA 4–55

13. Escreva as expressões Booleanas para cada um dos circuitos lógicos dados na Figura 4–56.



▲ **FIGURA 4–56**

14. Desenhe o circuito lógico representado por cada uma das seguintes expressões:
(a) $A + B + C$ **(b)** ABC **(c)** $AB + C$ **(d)** $AB + CD$

15. Desenhe o circuito lógico representado por cada expressão a seguir:
(a) $A\overline{B} + \overline{A}B$ **(b)** $AB + \overline{A}\,\overline{B} + \overline{A}BC$
(c) $\overline{A}B(C + \overline{D})$ **(d)** $A + B[C + D(B + \overline{C})]$

16. Construa uma tabela-verdade para cada uma das seguintes expressões Booleanas:
(a) $A + B$ **(b)** AB **(c)** $AB + BC$
(d) $(A + B)C$ **(e)** $(A + B)(\overline{B} + C)$

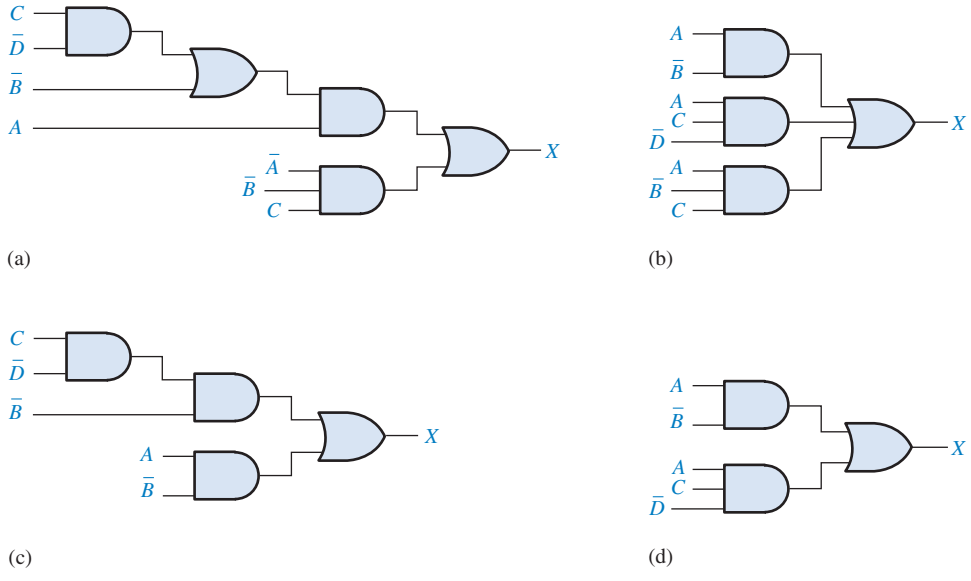
SEÇÃO 4–5 Simplificação Usando a Álgebra Booleana

17. Usando técnicas da álgebra Booleana, simplifique as seguintes expressões tanto quanto possível:
(a) $A(A + B)$ **(b)** $A(\overline{A} + AB)$ **(c)** $BC + \overline{B}C$
(d) $A(A + \overline{A}B)$ **(e)** $A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}C$

18. Usando a álgebra Booleana, simplifique as seguintes expressões:
(a) $(A + \overline{B})(A + C)$ **(b)** $\overline{A}B + \overline{A}B\overline{C} + \overline{A}BCD + \overline{A}B\overline{C}\,\overline{D}E$
(c) $AB + A\overline{B}C + A$ **(d)** $(A + \overline{A})(AB + AB\overline{C})$
(e) $AB + (\overline{A} + \overline{B})C + AB$

19. Usando a álgebra Booleana, simplifique cada expressão a seguir:
(a) $BD + B(D + E) + \overline{D}(D + F)$ **(b)** $\overline{A}\,\overline{B}C + \overline{(A + B + \overline{C})} + \overline{A}\,\overline{B}\,\overline{C}D$
(c) $(B + BC)(B + \overline{B}C)(B + D)$ **(d)** $ABCD + AB(\overline{CD}) + (\overline{AB})CD$
(e) $ABC[AB + \overline{C}(BC + AC)]$

20. Determine quais dos circuitos lógicos mostrados na Figura 4–57 são equivalentes:



▶ **FIGURA 4–57**

SEÇÃO 4–6 Formas Padronizadas de Expressões Booleanas

21. Converta as seguintes expressões para a forma de soma-de-produtos:

(a) $(A + B)(C + \overline{B})$ **(b)** $(A + \overline{B}C)C$ **(c)** $(A + C)(AB + AC)$

22. Converta as seguintes expressões para a forma de soma-de-produtos:

(a) $AB + CD(A\overline{B} + CD)$ **(b)** $AB(\overline{B}\,\overline{C} + BD)$ **(c)** $A + B[AC + (B + \overline{C})D]$

23. Defina o domínio de cada expressão de soma-de-produtos dada no Problema 21 e converta as expressões para a forma de soma-de-produtos padrão.

24. Converta cada expressão de soma-de-produtos dada no Problema 22 para a forma de soma-de-produtos padrão.

25. Determine o valor binário de cada termo nas expressões de soma-de-produtos padrão a partir do Problema 23.

26. Determine o valor binário de cada termo nas expressões de soma-de-produtos padrão a partir do Problema 24.

27. Converta cada expressão de soma-de-produtos no Problema 23 para a forma de produto-de-somas padrão.

28. Converta cada expressão de soma-de-produtos no Problema 24 para a forma de produto-de-somas padrão.

SEÇÃO 4–7 Expressões Booleanas e Tabelas-Verdade

29. Desenvolva uma tabela-verdade para cada uma das seguintes expressões de soma-de-produtos padrão:

(a) $A\overline{B}C + \overline{A}B\overline{C} + ABC$ **(b)** $\overline{X}\overline{Y}\overline{Z} + \overline{X}\,\overline{Y}Z + XY\overline{Z} + X\overline{Y}Z + \overline{X}YZ$

30. Desenvolva uma tabela-verdade para cada uma das seguintes expressões de soma-de-produtos padrão:

(a) $\overline{A}B\overline{C}D + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

(b) $WXYZ + WXY\overline{Z} + \overline{W}XYZ + W\overline{X}YZ + WX\overline{Y}Z$

31. Desenvolva uma tabela-verdade para cada uma das seguintes expressões de soma-de-produtos:

(a) $\overline{A}B + AB\overline{C} + \overline{A}\,\overline{C} + A\overline{B}C$ **(b)** $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z$

32. Desenvolva uma tabela-verdade para cada uma das seguintes expressões de produto-de-somas padrão:

(a) $(\overline{A} + \overline{B} + \overline{C})(A + B + C)(A + \overline{B} + C)$

(b) $(\overline{A} + B + \overline{C} + D)(A + \overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)(\overline{A} + B + C + \overline{D})$

33. Desenvolva uma tabela-verdade para cada uma das seguintes expressões de produto-de-somas padrão:

(a) $(A + B)(A + C)(A + B + C)$

(b) $(A + \overline{B})(A + \overline{B} + \overline{C})(B + C + \overline{D})(\overline{A} + B + \overline{C} + D)$

34. Para cada tabela-verdade na Figura 4–58, desenvolva uma expressão de soma-de-produtos padrão e outra de produto-de-somas padrão.

ABC	X
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

(a)

ABC	X
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

(b)

ABCD	X
0000	1
0001	1
0010	0
0011	1
0100	0
0101	1
0110	1
0111	0
1000	0
1001	1
1010	0
1011	0
1100	1
1101	0
1110	0
1111	0

(c)

ABCD	X
0000	0
0001	0
0010	1
0011	0
0100	1
0101	1
0110	0
0111	1
1000	0
1001	0
1010	0
1011	1
1100	1
1101	0
1110	0
1111	1

(d)

▶ **FIGURA 4–58**

SEÇÃO 4–8 O Mapa de Karnaugh

35. Desenhe um mapa de Karnaugh de 3 variáveis e rotule cada célula de acordo com o valor binário de cada uma.

36. Desenhe um mapa de Karnaugh de 4 variáveis e rotule cada célula de acordo com o valor binário de cada uma.

37. Escreva o termo-produto padrão para cada célula num mapa de Karnaugh de 3 variáveis.

SEÇÃO 4–9 Minimização de Soma-de-Produtos Usando o Mapa de Karnaugh

38. Use um mapa de Karnaugh para determinar a forma de soma-de-produtos para cada expressão a seguir:

(a) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}C$ **(b)** $AC(\overline{B} + C)$

(c) $\overline{A}(BC + B\overline{C}) + A(BC + B\overline{C})$ **(d)** $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + AB\overline{C}$

39. Use um mapa de Karnaugh para simplificar cada expressão a seguir para a forma de soma-de-produtos mínima:

(a) $\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + \overline{A}BC + AB\overline{C}$ **(b)** $AC[\overline{B} + B(B + \overline{C})]$

(c) $DE\overline{F} + \overline{D}E\overline{F} + \overline{D}\,\overline{E}\,\overline{F}$

40. Faça a expansão de cada expressão para a forma de soma-de-produtos mínima:

(a) $AB + A\overline{B}C + ABC$ **(b)** $A + BC$

(c) $A\overline{B}\,\overline{C}D + AC\overline{D} + B\overline{C}D + \overline{A}BC\overline{D}$ **(d)** $A\overline{B} + A\overline{B}\,\overline{C}D + CD + B\overline{C}D + ABCD$

41. Minimize cada expressão dada no Problema 40 usando um mapa de Karnaugh.

42. Use um mapa de Karnaugh para reduzir cada expressão para a forma de soma-de-produtos mínima:

(a) $A + B\overline{C} + CD$

(b) $\overline{A}\,\overline{B}C\overline{D} + \overline{A}B\overline{C}D + ABCD + ABC\overline{D}$

(c) $\overline{A}B(\overline{C}\,\overline{D} + \overline{C}D) + AB(\overline{C}\,\overline{D} + \overline{C}D) + A\overline{B}\,\overline{C}D$

(d) $(\overline{A}\,\overline{B} + A\overline{B})(CD + C\overline{D})$

(e) $\overline{A}\,\overline{B} + A\overline{B} + \overline{C}\,\overline{D} + C\overline{D}$

43. Reduza a função especificada na tabela-verdade dada na Figura 4–59 para a forma de soma-de-produtos mínima usando um mapa de Karnaugh.

44. Use o método do mapa de Karnaugh para implementar a expressão de soma-de-produtos mínima para a função lógica especificada na tabela-verdade mostrada na Figura 4–60.

Entradas			Saída
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

▲ FIGURA 4–59

Entradas				Saída
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

▲ FIGURA 4–60

45. Resolva o Problema 44 para uma situação na qual as últimas seis combinações binárias não são permitidas (não acontecem).

SEÇÃO 4–10 Minimização de Produto-de-Somas Usando o Mapa de Karnaugh

46. Use um mapa de Karnaugh para determinar o produto-de-somas mínimo para cada expressão:

(a) $(A + B + C)(\overline{A} + \overline{B} + \overline{C})(A + \overline{B} + C)$

(b) $(X + \overline{Y})(\overline{X} + Z)(X + \overline{Y} + \overline{Z})(\overline{X} + \overline{Y} + Z)$

(c) $A(B + \overline{C})(\overline{A} + C)(A + \overline{B} + C)(\overline{A} + B + \overline{C})$

47. Use o mapa de Karnaugh para simplificar cada expressão para a forma de produto-de-somas mínimo.

(a) $(A + \overline{B} + C + \overline{D})(\overline{A} + B + \overline{C} + D)(\overline{A} + \overline{B} + \overline{C} + \overline{D})$

(b) $(X + \overline{Y})(W + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z})(W + X + Y + Z)$

48. Para a função especificada na tabela-verdade da Figura 4–59, determine a expressão de produto-de-somas mínima usando um mapa de Karnaugh.

49. Determine a expressão de produto-de-somas mínima para a função na tabela-verdade da Figura 4–60.

50. Converta cada uma das seguintes expressões de produto-de-somas em expressões de soma-de-produtos mínimas usando um mapa de Karnaugh:

(a) $(A + \overline{B})(A + \overline{C})(\overline{A} + \overline{B} + C)$

(b) $(\overline{A} + B)(\overline{A} + \overline{B} + \overline{C})(B + \overline{C} + D)(A + \overline{B} + C + \overline{D})$

SEÇÃO 4–11 Mapas de Karnaugh de Cinco Variáveis

51. Minimize a seguinte expressão de soma-de-produtos usando um mapa de Karnaugh:

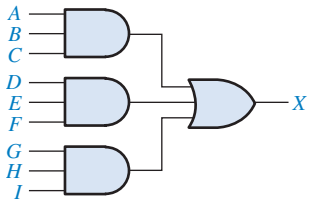
$$X = \overline{A}B\overline{C}D\overline{E} + \overline{A}\,\overline{B}\,\overline{C}DE + A\overline{B}\,\overline{C}DE + AB\overline{C}\,\overline{D}\,\overline{E} + \overline{A}BCD\overline{E} + \overline{A}BC\overline{D}E + \overline{A}\,\overline{B}C\overline{D}\,\overline{E} + \overline{A}\,\overline{B}CDE + AB\overline{C}D\overline{E} + AB\overline{C}DE$$

52. Aplique o método do mapa de Karnaugh para minimizar a seguinte expressão de soma-de-produtos:

$$A = \overline{V}WXYZ + V\overline{W}XYZ + VW\overline{X}YZ + VWX\overline{Y}Z + VWXY\overline{Z} + \overline{V}\,\overline{W}\overline{X}\overline{Y}\overline{Z} + \overline{V}\,\overline{W}\overline{X}YZ + \overline{V}W\overline{X}\overline{Y}\overline{Z} + \overline{V}W\overline{X}Y\overline{Z}$$

SEÇÃO 4–12 VHDL (Opcional)

53. Escreva um programa VHDL para o circuito lógico dado na Figura 4–61.



▶ FIGURA 4–61

54. Escreva um programa em VHDL para a expressão

$$Y = A\overline{B}C + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}BC$$

Aplicações em Sistemas Digitais

55. Se um técnico é solicitado para escolher um tipo de display digital para um ambiente de baixa luminosidade, ele deve selecionar um display de LEDs ou um LCD de 7 segmentos? Por quê?

56. Explique porque os códigos 1010, 1011, 1100, 1101, 1110 e 1111 se enquadram na categoria de "don't care" em aplicações de displays de 7 segmentos?

57. Para o segmento *b*, qual a quantidade de portas e inversores a menos são necessários para implementar a expressão de soma-de-produtos mínima em relação à expressão de soma-de-produtos padrão.

58. Repita o Problema 57 para a lógica dos segmentos de *c* até *g*.

Problemas Especiais de Projeto

59. A lógica para o segmento *a* na Figura 4–52 produz uma saída de nível ALTO para ativar o segmento assim como os circuitos para cada um dos outros segmentos. Se um tipo de display de 7 segmentos que for usado necessitar de um nível BAIXO para ativar cada segmento, modifique a lógica de segmento para ficar de acordo.

60. Reprojete a lógica para o segmento *a* usando uma expressão de produto-de-somas mínima. Qual das expressões é mais simples, a de produto-de-somas mínima ou a de soma-de-produtos mínima?

61. Repita o Problema 60 para os segmentos de *b* até *g*.

62. Sintetize os resultados dos seus esforços de reprojeto despendidos nos Problemas 60 e 61 e recomende o melhor projeto baseado num menor número de CIs. Especifique os tipos de CIs.

Prática de Análise de Defeito Usando o Multisim

63. Abra o arquivo P04-63, aplique os sinais de entrada e observe a operação do circuito lógico. Determine se existe ou não um defeito.

64. Abra o arquivo P04-64, aplique os sinais de entrada e observe a operação do circuito lógico. Determine se existe ou não um defeito.

65. Abra o arquivo P04-65, aplique os sinais de entrada e observe a operação do circuito lógico. Determine se existe ou não um defeito.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 4–1 Operações e Expressões Booleanas

1. $\overline{A} = \overline{0} = 1$ **2.** $A = 1, B = 1, C = 0; \overline{A} + \overline{B} + C = \overline{1} + \overline{1} + 0 = 0 + 0 + 0 = 0$

3. $A = 1, B = 0, C = 1; A\overline{B}C = 1 \cdot 0 \cdot \overline{1} = 1 \cdot 1 \cdot 1 = 1$

SEÇÃO 4–2 Leis e Regras da Álgebra Booleana

1. $A + (B + C + D) = (A + B + C) + D$ **2.** $A(B + C + D) = AB + AC + AD$

SEÇÃO 4–3 Teoremas de DeMorgan

1. **(a)** $\overline{ABC} + \overline{(\overline{D} + E)} = \overline{A} + \overline{B} + \overline{C} + D\overline{E}$ **(b)** $\overline{(A + B)C} = \overline{A}\,\overline{B} + \overline{C}$

(c) $\overline{A + B + C} + \overline{\overline{D}\overline{E}} = \overline{A}\,\overline{B}\,\overline{C} + D + \overline{E}$

SEÇÃO 4–4 Análise Booleana de Circuitos Lógicos

1. $(C + D)B + A$

2. Tabela-verdade abreviada: a expressão é 1 quando *A* é 1 ou quando *B* e *C* são 1s ou quando *B* e *D* são 1s. A expressão é 0 para todas as outras combinações das variáveis.

SEÇÃO 4–5 Simplificação Usando a Álgebra Booleana

1. **(a)** $A + AB + A\overline{B}C = A$ **(b)** $(\overline{A} + B)C + ABC = C(\overline{A} + B)$

(c) $A\overline{B}C(BD + CDE) + A\overline{C} = A(\overline{C} + \overline{B}DE)$

2. **(a)** *Original*: 2 portas AND, 1 porta OR, 1 inversor; *Simplificado*: nenhuma porta (conexão direta).

(b) *Original*: 2 portas OR, 2 portas AND, 1 inversor; *Simplificado*: 1 porta OR, 1 porta AND, 1 inversor.

(c) *Original*: 5 portas AND, 2 portas OR, 2 inversores; *Simplificado*: 2 portas AND, 1 porta OR, 2 inversores.

SEÇÃO 4–6 Formas Padronizadas de Expressões Booleanas

1. **(a)** soma-de-produtos **(b)** produto-de-somas padrão **(c)** soma-de-produtos padrão **(d)** produto-de-somas

2. **(a)** $AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D}$

(c) Já está na forma padrão

3. **(b)** Já está na forma padrão

(d) $(A + \overline{B} + \overline{C})(A + \overline{B} + C)(A + B + \overline{C})(A + B + C)$

SEÇÃO 4–7 Expressões Booleanas e Tabelas-verdade

1. $2^5 = 32$ **2.** $0110 \longrightarrow \overline{W}XY\overline{Z}$ **3.** $1100 \longrightarrow \overline{W} + \overline{X} + Y + Z$

SEÇÃO 4–8 O Mapa de Karnaugh

1. **(a)** célula superior esquerda: 000 **(b)** célula inferior direita: 101
(c) célula inferior esquerda: 100 **(d)** célula superior direita: 001

2. **(a)** célula superior esquerda: $\overline{X}\,\overline{Y}\,\overline{Z}$ **(b)** célula inferior direita: $X\overline{Y}Z$
(c) célula inferior esquerda: $X\overline{Y}\,\overline{Z}$ **(d)** célula superior direita: $\overline{X}\,\overline{Y}Z$

3. **(a)** célula superior esquerda: 0000 **(b)** célula inferior direita: 1010
(c) célula inferior esquerda: 1000 **(d)** célula superior direita: 0010

4. **(a)** célula superior esquerda: $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ **(b)** célula inferior direita: $W\overline{X}YZ$
(c) célula inferior esquerda: $W\overline{X}\overline{Y}Z$ **(d)** célula superior direita: $W\overline{X}Y\overline{Z}$

SEÇÃO 4–9 Minimização de Soma-de-Produtos Usando o Mapa de Karnaugh

1. Mapa de 8 células para 3 variáveis; mapa de 16 células para 4 variáveis

2. $AB + B\overline{C} + \overline{A}\,\overline{B}C$

3. **(a)** $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + AB\overline{C}$

(b) $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$

(c) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}CD$

(d) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$

SEÇÃO 4–10 Minimização de Produto-de-Somas Usando o Mapa de Karnaugh

1. Inserindo uma expressão de produto-de-somas num mapa, os 0s são colocados em células cujos valores tornam o termo-soma padrão zero; e na inserção de uma expressão de soma-de-produtos num mapa, os 1s são colocados nas células que possuem os mesmos valores que os termos-produto.

2. 0 na célula 1011: $\overline{A} + B + \overline{C} + \overline{D}$ **3.** 1 na célula 0010: $\overline{A}\,\overline{B}C\overline{D}$

SEÇÃO 4–11 Mapas de Karnaugh de Cinco Variáveis

1. Existem 32 combinações de 5 variáveis ($2^5 = 32$)

2. $X = 1$ porque a função é 1 para todas as combinações possíveis de 5 variáveis.

SEÇÃO 4–12 VHDL (Opcional)

1. Um HDL é uma linguagem de descrição de hardware para lógica de programação.

2. Entidade e arquitetura

3. A entidade especifica as entradas e saídas de uma função lógica.

4. A arquitetura especifica a operação de uma função lógica.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

4–1. $\overline{A} + B = 0$ quando $A = 1$ e $B = 0$.

4–2. $\overline{A}\,\overline{B} = 1$ quando $A = 0$ e $B = 0$. **4–3.** XYZ

4–4. $W + X + Y + Z$ **4–5.** $ABC\overline{D}\,\overline{E}$ **4–6.** $(A + \overline{B} + \overline{C}D)\overline{E}$

4–7. $\overline{ABCD} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$ **4-8.** $A\overline{B}$ **4-9.** CD

4–10. $AB\overline{C} + \overline{A}C + \overline{A}\,\overline{B}$

4–11. $\overline{A} + \overline{B} + \overline{C}$ **4–12.** $\overline{A}B\overline{C} + AB + A\overline{C} + A\overline{B} + \overline{B}\,\overline{C}$

4–13. $W\overline{X}YZ + W\overline{X}Y\overline{Z} + W\overline{X}\overline{Y}Z + \overline{W}\,\overline{X}YZ + WX\overline{Y}Z + WX\overline{Y}\,\overline{Z}$

4–14. 011, 101, 110, 010, 111. Sim.

4–15. $(A + \overline{B} + C)(A + \overline{B} + \overline{C})(A + B + C)(\overline{A} + B + C)$

4–16. 010, 100, 001, 111, 011. Sim

4–17. Expressões de soma-de-produtos e produto-de-somas são equivalentes

4–18. Veja a Tabela 4–11. **4–19.** Veja a Tabela 4–12.

▼ **TABELA 4–11**

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

▼ **TABELA 4–11**

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

4–20. As expressões de soma-de-produtos e produto-de-somas são equivalentes.

4–21. Veja a Figura 4–62. **4–22.** Veja a Figura 4–63. **4–23.** Veja a Figura 4–64.

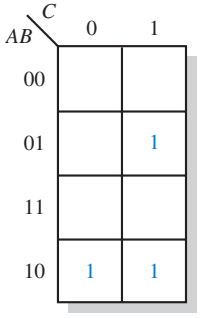
4–24. Veja a Figura 4–65. **4–25.** Não existe nenhuma outra forma.

4–26. $X = B + \overline{A}C + A\overline{C}D + C\overline{D}$

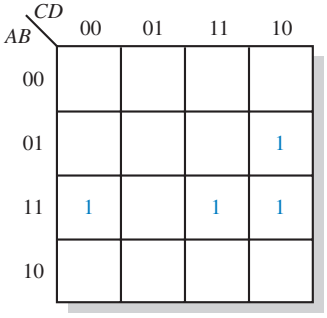
4–27. $X = \overline{D} + A\overline{B}C + B\overline{C} + \overline{A}B$ **4–28.** $Q = X + Y$

4–29. $Q = \overline{X}\overline{Y}\overline{Z} + W\overline{X}Z + \overline{W}YZ$

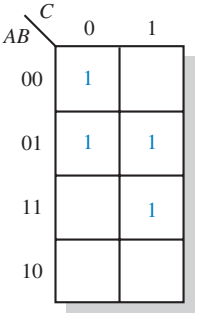
4–30. Veja a Figura 4–66.



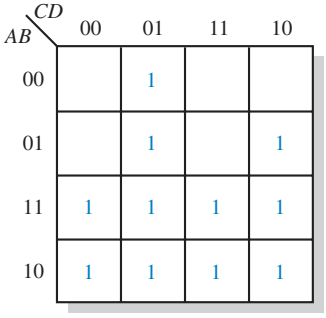
▲ **FIGURA 4–62**



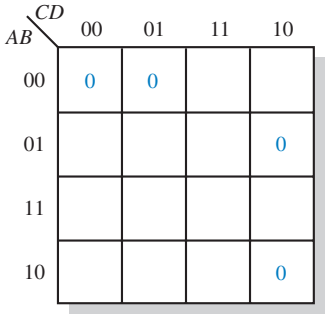
▲ **FIGURA 4–63**



▲ **FIGURA 4–64**



▲ **FIGURA 4–65**



▲ **FIGURA 4–66**

4–31. $Q = (X + \overline{Y})(X + \overline{Z})(\overline{X} + Y + Z)$

4–32. $Q = (\overline{X} + \overline{Y} + Z)(\overline{W} + \overline{X} + Z)(W + X + Y + Z)(W + \overline{X} + Y + \overline{Z})$

4–33. $Q = \overline{Y}\overline{Z} + \overline{X}\overline{Z} + \overline{W}Y + \overline{X}\overline{Y}Z$

4–34. $Y = \overline{D}\overline{E} + \overline{A}\overline{E} + \overline{B}\overline{C}\overline{E}$

4–35. X ⇐ (A **and** B) **nor** (C **and** D)

AUTOTESTE

1. (d) **2.** (a) **3.** (b) **4.** (c) **5.** (d) **6.** (b) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (b) **13.** (b) **14.** (c) **15.** (a) **16.** (c)
17. (a) **18.** (b) **19.** (c) **20.** (b) **21.** (c)

5

Análise Lógica Combinacional

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Analisar circuitos lógicos combinacionais básicos, tais como AND-OR, AND-OR-inversor, EX-OR e EX-NOR
- Usar circuitos AND-OR e AND-OR-inversor para implementar expressões de produto-de-somas e de soma-de-produtos
- Escrever a expressão Booleana de saída para qualquer circuito lógico combinacional
- Desenvolver uma tabela-verdade a partir da expressão de saída para um circuito lógico combinacional
- Usar o mapa de Karnaugh para expandir uma expressão de saída contendo termos com variáveis que não aparecem na expressão de soma-de-produtos original

- Projetar um circuito lógico combinacional para uma dada expressão Booleana de saída
- Projetar um circuito lógico combinacional para uma dada tabela-verdade
- Simplificar um circuito lógico combinacional para a sua forma mínima
- Usar portas NAND para implementar qualquer função lógica combinacional
- Usar porta NOR para implementar qualquer função lógica combinacional
- Escrever programas VHDL para circuitos lógicos simples
- Realizar a análise de defeito de circuitos lógicos
- Realizar a análise de defeito de circuitos lógicos usando a análise de formas de onda e rastreamento de sinal
- Aplicar um circuito lógico combinacional num sistema

TERMOS IMPORTANTES

Termos importantes na ordem em que aparecem no capítulo.

- Porta universal
- OR negativa
- AND negativa
- Componente
- Sinal
- Nó
- Rastreamento de sinal

INTRODUÇÃO

Nos Capítulos 3 e 4, as portas lógicas foram discutidas individualmente e em combinações simples. Foram introduzidas as implementações de soma-de-produtos e produto-de-somas, que são as formas básicas da lógica combinacional. Quando portas lógicas são interconectadas para produzir uma saída especificada para certas combinações das variáveis de entrada, sem o envolvimento de armazenamento de dados, o circuito resultante está na categoria de **lógica combinacional**. Em lógica combinacional, o nível lógico da saída depende todo o tempo da combinação dos níveis lógicos das entradas. Esse capítulo explora o assunto introduzido em capítulos anteriores com uma abordagem de análise de funcionamento, projeto e análise de defeito de diversos circuitos lógicos combinacionais. A abordagem estrutural do VHDL é introduzida e aplicada à lógica combinacional.

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico Aplicações em Sistemas Digitais ilustra os conceitos ensinados nesse capítulo demonstrando como a lógica combinacional pode ser usada para uma finalidade específica numa aplicação prática. Um circuito lógico é usado para controlar o nível e a temperatura de um fluido num tanque de armazenamento. Operando válvulas de entrada e saída, os fluxos de entrada e saída são controlados com base nas entradas dos sensores de nível. A temperatura do fluido é controlada ligando e desligando um elemento de aquecimento com base nas entradas do sensor de temperatura. Como uma opção, é discutido também o uso do VHDL para descrição de circuitos lógicos.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

5-1 CIRCUITOS LÓGICOS COMBINACIONAIS BÁSICOS

No Capítulo 4, aprendemos que expressões de soma-de-produtos são implementadas com uma porta AND para cada termo-produto e uma porta OR para somar todos os termos-produto. Como já sabemos, essa implementação de soma-de-produtos é denominada lógica AND-OR e é a forma básica para implementar funções Booleanas padrão. Nesta seção, as lógicas AND-OR e AND-OR-inversor são analisadas; as portas EX-OR e EX-NOR, que são na realidade uma forma de lógica AND-OR, também são abordadas.

Ao final do estudo desta seção você deverá ser capaz de:

- Analisar e aplicar circuitos AND-OR
- Analisar e aplicar circuitos AND-OR-inversor
- Analisar e aplicar portas EX-OR
- Analisar e aplicar portas EX-NOR

Lógica AND-OR

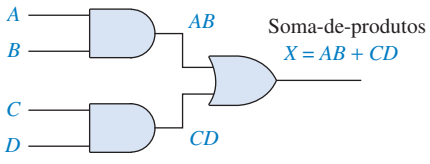
A lógica AND-OR produz uma expressão de soma-de-produtos.

A Figura 5–1(a) mostra um circuito AND-OR que consiste em duas portas AND de 2 entradas e uma porta OR de 2 entradas; a Figura 5–1(b) é o símbolo retangular padrão ANSI. As expressões Booleanas para as saídas das portas AND e a expressão de soma-de-produtos resultante para a saída X são mostradas no diagrama. Em geral, um circuito AND-OR pode ter qualquer número de portas AND, cada uma com um número qualquer de entradas.

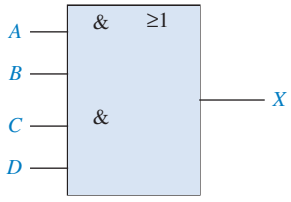
A tabela-verdade para um circuito lógico AND-OR de 4 entradas é mostrada na Tabela 5–1. A saída da porta AND intermediária (as colunas AB e CD) também são mostradas na tabela.

FIGURA 5–1

Um exemplo de lógica AND-OR. Abra o arquivo F05-01 para verificar a operação.



(a) Diagrama lógico (símbolos característicos padrão ANSI)



(b) Símbolo retangular (padrão ANSI)

TABELA 5–1

Tabela-verdade para a lógica AND-OR do circuito mostrado na Figura 5–1

ENTRADAS						SAÍDA
A	B	C	D	AB	CD	X
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Um circuito AND-OR implementa diretamente uma expressão de soma-de-produtos, considerando que o complemento (se houver) das variáveis estejam disponíveis. A operação do circuito AND-OR mostrado na Figura 5–1 é expressa da seguinte forma:

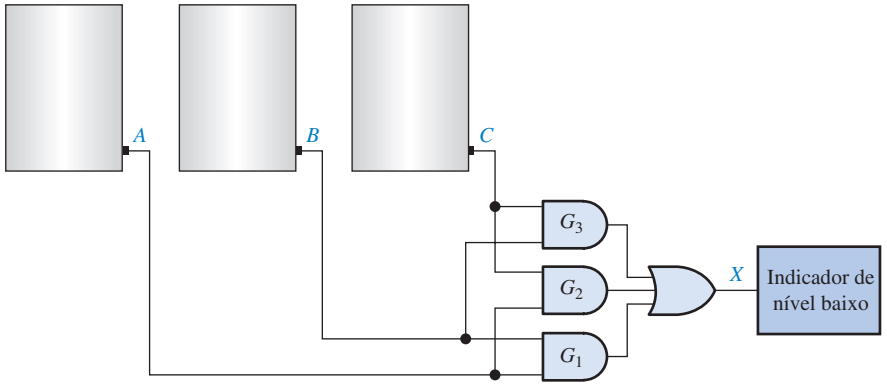
Para um circuito lógico AND-OR de 4 entradas, a saída *X* é nível ALTO (1) se as entradas *A* e *B* estiverem em nível ALTO (1) ou as entradas *C* e *D* estiverem em nível ALTO (1).

EXEMPLO 5–1

Em uma certa planta de um processo químico, uma substância química na forma líquida é usada num processo industrial. O líquido é armazenado em três tanques diferentes. Um sensor de nível em cada tanque produz uma tensão de nível ALTO quando o nível do líquido no tanque cai abaixo de um ponto especificado.

Projete um circuito que monitore o nível do líquido em cada tanque e indique quando o nível em dois tanques quaisquer cai abaixo do ponto especificado.

Solução O circuito AND-OR mostrado na Figura 5–2 tem entradas a partir de sensores nos tanques *A*, *B* e *C* conforme mostrado. A porta AND G_1 monitora os níveis nos tanques *A* e *B*, a porta G_2 monitora os tanques *A* e *C* e a parta G_3 monitora os tanques *B* e *C*. Quando o nível do líquido em dois tanques quaisquer se torna baixo, uma das portas AND terá níveis ALTOs em suas duas entradas, fazendo com que a saída seja nível ALTO; assim a saída final *X* a partir da porta OR é nível ALTO. Essa entrada de nível ALTO é então usada para ativar um indicador tal como uma lâmpada ou um alarme audível, conforme mostra a figura.



▲ **FIGURA 5–2**

Problema relacionado* Escreva a expressão Booleana de soma-de-produtos para a lógica AND-OR vista na Figura 5–2.

* As respostas estão no final do capítulo.

Lógica AND-OR-Inversor

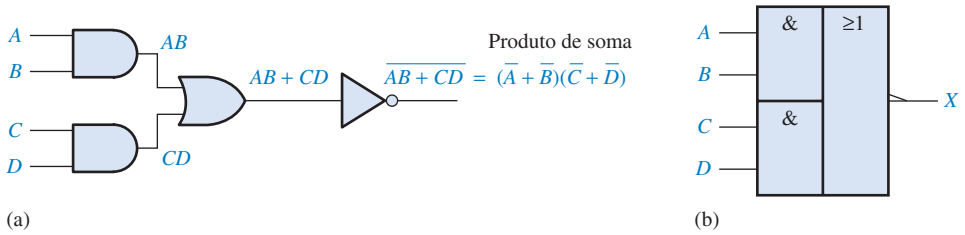
Quando a saída de um circuito AND-OR é complementada (invertida), resulta num circuito AND-OR-Inversor. Lembre-se que as expressões de soma-de-produtos implementam diretamente uma lógica AND-OR. Expressões de produto-de-somas podem ser implementadas com lógica AND-OR-Inversor. Isso está ilustrado a seguir, começando com uma expressão de produto-de-somas e desenvolvendo a correspondente expressão AND-OR-Inversor.

$$X = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) = (\overline{AB})(\overline{CD}) = \overline{\overline{(\overline{AB})(\overline{CD})}} = \overline{\overline{\overline{AB}} + \overline{\overline{CD}}} = \overline{AB + CD}$$

O diagrama lógico mostrado na Figura 5–3(a) mostra um circuito AND-OR-Inversor e o desenvolvimento da expressão de saída de produto-de-somas. O símbolo retangular padrão ANSI é mostrado na parte (b). Em geral, um circuito AND-OR-Inversor pode ter um número qualquer de portas AND tendo cada uma um número qualquer de entradas.

▶ FIGURA 5–3

Um circuito AND-OR-Inversor produz uma saída de produto-de-somas. Abra o arquivo F05-03 para verificar a operação.



A operação do circuito AND-OR-Inversor mostrado na Figura 5–3 é expressa como a seguir:

Para um circuito lógico AND-OR-Inversor de 4 entradas, a saída *X* é nível BAIXO (0) se as entradas *A* e *B* estiverem em nível ALTO (1) ou as entradas *C* e *D* estiverem em nível ALTO (1).

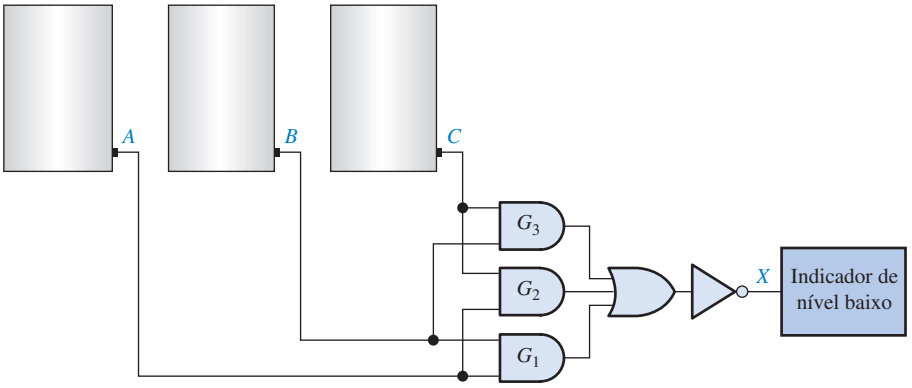
Uma tabela-verdade pode ser desenvolvida a partir da tabela-verdade AND-OR dada na Tabela 5–1 simplesmente trocando todos os 1s por 0s e todos os 0s por 1s na coluna de saída.

EXEMPLO 5–2

Os sensores nos tanques que contêm um produto químico na forma líquida conforme mostra a Figura 5–1 são substituídos por um novo modelo que produz uma tensão de nível BAIXO em vez de uma tensão de nível ALTO quando o nível do líquido não tanque cai abaixo de um ponto crítico.

Modifique o circuito dado na Figura 5–2 para operar com níveis lógicos de entrada diferentes e ainda produzir uma saída de nível ALTO para ativar o indicador quando os níveis em dois tanques quaisquer caírem abaixo do ponto crítico. Mostre o diagrama lógico.

Solução O circuito AND-OR-Inversor visto na Figura 5–4 tem entradas a partir de sensores nos tanques *A*, *B* e *C* como mostrado. A porta AND G_1 monitora os níveis nos tanques *A* e *B*, a porta G_2 monitora os tanques *A* e *C* e a porta G_3 monitora os tanques *B* e *C*. Quando os níveis dos líquidos em dois tanques quaisquer estiverem muito baixos, cada porta AND terá um nível BAIXO em pelo menos uma entrada fazendo com que sua saída tenha um nível BAIXO, assim a saída final *X* a partir do inversor é nível ALTO. Essa saída de nível ALTO é então usada para ativar um indicador.



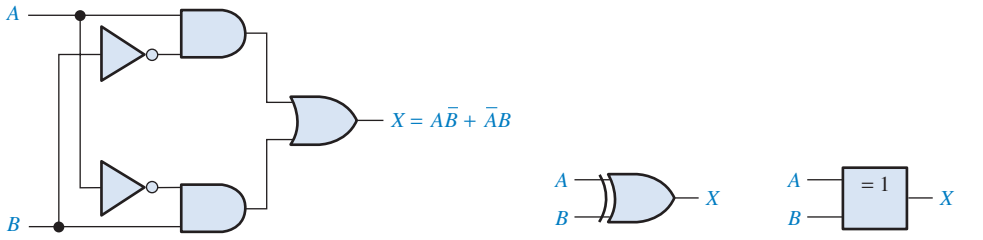
▲ FIGURA 5–4

Problema relacionado Escreva a expressão Booleana para a lógica AND-OR-Inversor mostrada na Figura 5–4 e mostre que a saída é nível ALTO (1) quando duas entradas quaisquer dentre as entradas *A*, *B* e *C* estiverem em nível BAIXO (0).

Lógica EX-OR

A porta EX-OR é uma combinação de outras portas.

A porta EX-OR foi introduzida no Capítulo 3. Embora, devido à sua importância, esse circuito seja considerado um tipo de porta lógica com o seu próprio símbolo, ele é na realidade uma combinação de duas portas AND, uma porta OR e dois inversores, conforme mostrado na Figura 5–5(a). Os dois símbolos lógicos são mostrados nas partes (b) e (c).



(a) Diagrama lógico (b) Símbolo característico ANSI (c) Símbolo retangular ANSI

◀ **FIGURA 5–5**
Diagrama lógico da EX-OR e símbolos. Abra o arquivo F05-05 para verificar a operação.

A expressão de saída para o circuito mostrado na Figura 5–5 é

$$X = A\overline{B} + \overline{A}B$$

A avaliação dessa expressão resulta na tabela-verdade vista na Tabela 5–2. Observe que a saída é nível ALTO apenas quando as duas entradas estão em níveis opostos. Um operador EX-OR especial $\oplus$ é usado sempre, assim a expressão $X = A\overline{B} + \overline{A}B$ pode ser lida como "X é igual a A EX-OR B" e pode ser escrita como:

$$X = A \oplus B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

◀ **TABELA 5–2**
Tabela-verdade para uma EX-OR

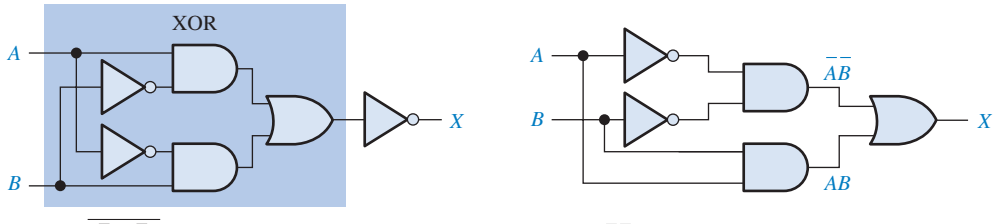
Lógica EX-NOR

Como já sabemos, o complemento de uma função EX-OR é a EX-NOR, deduzida como:

$$X = \overline{A\overline{B} + \overline{A}B} = \overline{(A\overline{B})}\,\overline{(\overline{A}B)} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

Observe que a saída X é nível ALTO apenas quando as duas entradas A e B estão no mesmo nível lógico.

A EX-NOR pode ser implementada simplesmente invertendo a saída de uma EX-OR, conforme mostra a Figura 5–6(a), ou diretamente implementando a expressão $\overline{A}\,\overline{B} + AB$, como mostrado na parte (b).



(a) $X = \overline{A\overline{B} + \overline{A}B}$ (b) $X = \overline{A}\,\overline{B} + AB$

◀ **FIGURA 5–6**
Duas formas equivalentes de implementar a EX-NOR. Abra o arquivo F05-06 para verificar a operação.

SEÇÃO 5–1 REVISÃO
As respostas estão no final do capítulo.

1. Determine a saída (1 ou 0) de um circuito AND-OR-Inversor de 4 variáveis para cada uma das seguintes condições de entrada:
   (a) $A = 1, B = 0, C = 1, D = 0$ **(b)** $A = 1, B = 1, C = 0, D = 1$
   (c) $A = 0, B = 1, C = 1, D = 1$
2. Determine a saída (1 ou 0) de uma porta EX-OR para cada uma das seguintes condições de entrada:
   (a) $A = 1, B = 0$ **(b)** $A = 1, B = 1$
   (c) $A = 0, B = 1$ **(d)** $A = 0, B = 0$
3. Desenvolva a tabela-verdade para um certo circuito lógico de 3 entradas com a expressão de saída $X = A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C} + AB\overline{C} + ABC$.
4. Desenhe o diagrama lógico para um circuito EX-NOR.

5-2 IMPLEMENTAÇÃO DE LÓGICA COMBINACIONAL

Nesta seção, são usados exemplos para ilustrar como implementar um circuito lógico a partir de uma expressão Booleana ou uma tabela-verdade. É incluída também a minimização de um circuito lógico usando métodos abordados no Capítulo 4.

Ao final do estudo desta seção você deverá ser capaz de:

■ Implementar um circuito lógico a partir de uma expressão Booleana ■ Implementar um circuito lógico a partir de uma tabela verdade ■ Minimizar um circuito lógico

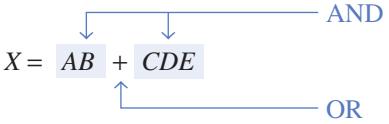
De uma Expressão Booleana para um Circuito lógico

Para cada expressão Booleana existe um circuito lógico e para cada circuito lógico existe uma expressão Booleana.

Vamos examinar a seguinte expressão Booleana:

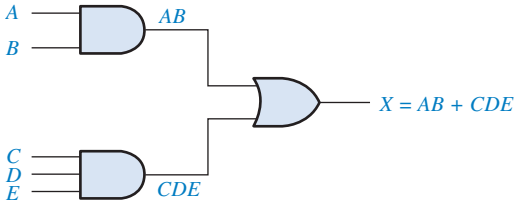
$$X = AB + CDE$$

Uma breve inspeção mostra que essa expressão é composta de dois termos, *AB* e *CDE*, com um domínio de cinco variáveis. O primeiro termo é formado por uma AND entre *A* e *B* e o segundo termo é formado por uma AND entre *C*, *D* e *E*. Os dois termos passam então por uma função OR para formar a saída *X*. Essas operações são indicadas na estrutura da expressão como mostrado a seguir:



Observe que nessa expressão particular, as operações AND formam os dois termos individuais, *AB* e *CDE*, os quais têm que ser formados *antes* de submetê-los a uma operação OR.

Para implementar essa expressão Booleana, uma porta AND de 2 entradas é necessária para formar o termo *AB* e uma porta AND de três entradas é usada para formar o termo *CDE*. Em seguida, é necessário uma porta OR para combinar os dois termos AND. O circuito lógico resultante é mostrado na Figura 5–7.



▶ **FIGURA 5–7**
Circuito lógico para $X = AB + CDE$.

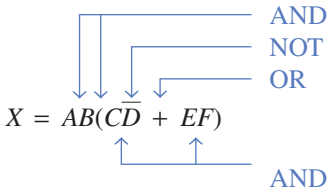
NOTA: COMPUTAÇÃO

Muitos programas de controle necessitam que operações lógicas sejam realizadas por um computador. Um programa driver é um programa de controle usado com os periféricos do computador. Por exemplo, um driver de mouse necessita de testes lógicos para determinar se um botão foi pressionado e outras operações lógicas para determinar se ele foi movido, horizontalmente ou verticalmente. O "coração" de um microprocessador é a unidade lógica e aritmética (ALU), a qual realiza essas operações lógicas diretamente através das instruções do programa. Toda a lógica descrita neste capítulo também pode ser realizada por uma ALU, fornecendo-se as instruções adequadas.

Como um outro exemplo, vamos implementar a seguinte expressão:

$$X = AB(C\overline{D} + EF)$$

Observando essa expressão, vemos que existe uma operação AND entre os termos AB e $(C\overline{D} + EF)$. O termo $C\overline{D} + EF$ é formado pela operação OR entre os resultados das operações AND entre C e e entre E e F. Essa estrutura é indicada a seguir:



Antes de podermos implementar a expressão final, temos que criar o termo-soma $C\overline{D} + EF$; mas antes disso precisamos criar os termos-produto $C\overline{D}$ e EF; mas antes de obtermos o termo $C\overline{D}$, temos que criar $\overline{D}$. Portanto, como podemos ver, as operações lógicas têm que ser feitas na ordem adequada.

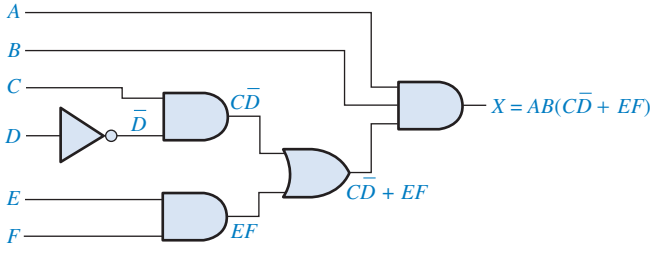
As portas lógicas necessárias para implementar $X = AB(C\overline{D} + EF)$ são as seguintes:

1. Um inversor para se obter $\overline{D}$
2. Duas portas AND de duas entradas para se obter $C\overline{D}$ e EF
3. Uma porta OR de duas entradas para se obter $C\overline{D} + EF$
4. Uma porta AND de três entradas para se obter X

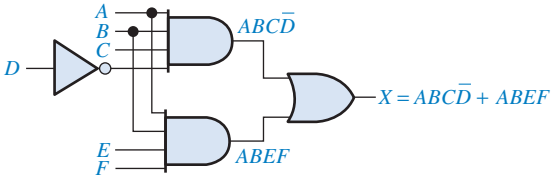
O circuito lógico para essa expressão é mostrado na Figura 5–8(a). Observe que existe um máximo de quatro portas e um inversor entre uma entrada e uma saída no circuito (da entrada D para a saída). Freqüentemente o atraso de propagação total através do circuito lógico é uma consideração importante. Os atrasos de propagação são aditivos, de forma que quanto mais portas ou inversores entre entrada e saída, maior o tempo de atraso de propagação.

A menos que um termo intermediário, tal como $C\overline{D} + EF$ na Figura 5–8(a), seja necessário como uma saída para alguma outra finalidade, normalmente é melhor reduzir um circuito para a sua forma de soma-de-produtos para reduzir o tempo total de atraso de propagação. A expressão é convertida para soma-de-produtos como a seguir, e o circuito resultante é mostrado na Figura 5–8(b).

$$AB(C\overline{D} + EF) = ABC\overline{D} + ABEF$$



(a)



(b) Implementação com soma-de-produtos do circuito da parte (a)

▲ **FIGURA 5–8**

Circuitos lógicos para $AB(C\overline{D} + EF) = ABC\overline{D} + ABEF$

De uma Tabela-Verdade para um Circuito Lógico

Se começarmos com uma tabela-verdade em vez de uma expressão, podemos escrever a expressão de soma-de-produtos a partir da tabela-verdade e então implementar o circuito lógico. A Tabela 5–3 especifica uma função lógica.

TABELA 5–3

ENTRADAS A	B	C	SAÍDA X	TERMO PRODUTO
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

A expressão Booleana de soma-de-produtos obtida a partir da tabela-verdade fazendo a operação OR entre os termos-produto para os quais $X = 1$ é

$$X = \overline{A}BC + A\overline{B}\,\overline{C}$$

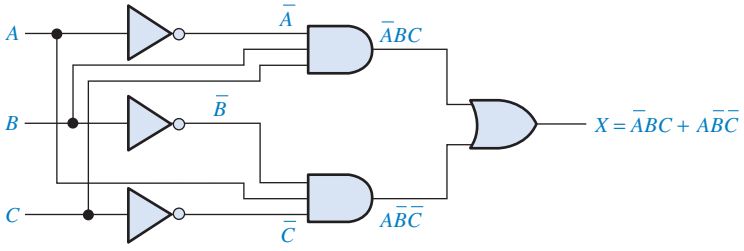
O primeiro termo na expressão é formado pela operação AND entre as três variáveis, $\overline{A}$, B, e C. O segundo termo é formado pela operação AND entre as três variáveis A, $\overline{B}$, e $\overline{C}$.

As portas lógicas necessárias para implementar essa expressão são as seguintes: três inversores para se obter as variáveis, $\overline{A}$, $\overline{B}$, e $\overline{C}$; duas portas AND de 3 entradas para se obter os termos $\overline{A}BC$ e $A\overline{B}\,\overline{C}$; e uma porta OR de 2 entradas para se obter a função de saída final, $\overline{A}BC + A\overline{B}\,\overline{C}$.

A implementação dessa função lógica é ilustrada na Figura 5–9.

FIGURA 5–9

Circuito lógico para $X = \overline{A}BC + A\overline{B}\,\overline{C}$. Abra o arquivo F05-09 para verificar a operação.



EXEMPLO 5–3

Projete um circuito lógico para implementar a operação especificada na tabela-verdade mostrada na Tabela 5–4.

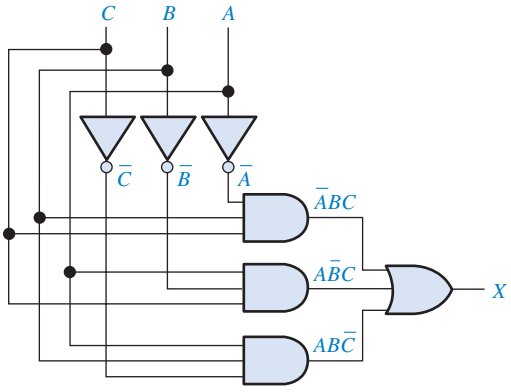
TABELA 5–4

ENTRADAS A	B	C	SAÍDA X	TERMO PRODUTO
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	0	
1	0	1	1	$A\overline{B}C$
1	1	0	1	$AB\overline{C}$
1	1	1	0	

Solução Observe que $X = 1$ para apenas três das condições de entrada. Portanto, a expressão lógica é:

$$X = \overline{A}BC + A\overline{B}C + AB\overline{C}$$

A portas lógicas necessárias são três inversores, três portas AND de 3 entradas e uma porta OR de 3 entradas. O circuito lógico é mostrado na Figura 5–10.



► **FIGURA 5–10**

Abra o arquivo F05-10 para verificar a operação.

Problema relacionado Determine se o circuito lógico mostrado na Figura 5–10 pode ser simplificado.

EXEMPLO 5–4

Desenvolva um circuito lógico com quatro variáveis de entrada que apenas produzirá uma saída 1 quando as três variáveis de entradas forem exatamente 1s.

Solução Dentre as dezesseis combinações possíveis com quatro variáveis, as combinações nas quais existem exatamente três 1s são apresentadas na Tabela 5–5, juntamente com o correspondente termo-produto para cada uma.

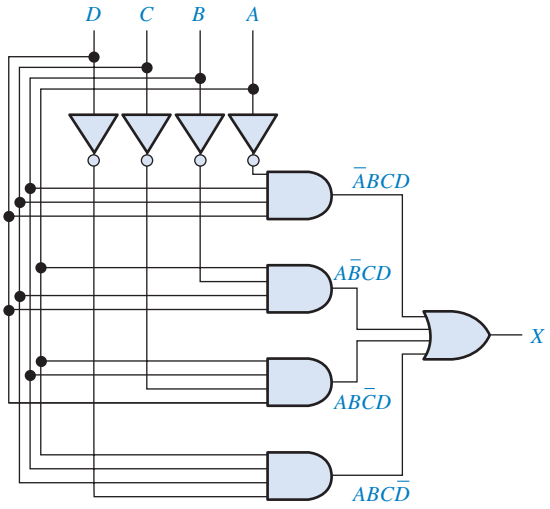
► **TABELA 5–5**

A	B	C	D	TERMO PRODUTO
0	1	1	1	$\overline{A}BCD$
1	0	1	1	$A\overline{B}CD$
1	1	0	1	$AB\overline{C}D$
1	1	1	0	$ABC\overline{D}$

Fazendo uma operação OR entre os termos-produto obtemos a seguinte expressão:

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D}$$

Essa expressão é implementada na Figura 5–11 com uma lógica AND-OR.



▶ **FIGURA 5–11**

Abra o arquivo F05-11 para verificar a operação.

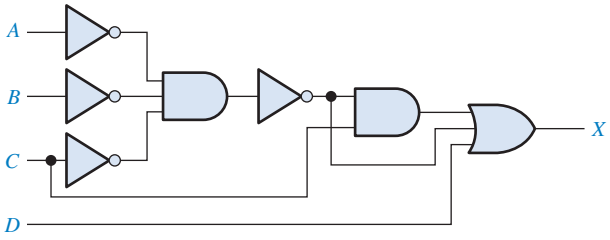
Problema relacionado Determine se o circuito lógico mostrado na Figura 5–11 pode ser simplificado.

EXEMPLO 5–5

Reduza o circuito lógico combinacional mostrado na Figura 5–12 para uma forma mínima.

▶ **FIGURA 5–12**

Abra o arquivo F05-12 para verificar que esse circuito é equivalente ao circuito mostrado na Figura 5–13.



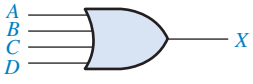
Solução Determine se o circuito lógico mostrado na Figura 5–11 pode ser simplificado.

$$X = (\overline{\overline{A}\,\overline{B}\,\overline{C}})C + \overline{\overline{A}\,\overline{B}\,\overline{C}} + D$$

Aplicando o teorema de DeMorgan e a álgebra Booleana,

$$\begin{aligned} X &= (\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}})C + \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + D \\ &= AC + BC + CC + A + B + C + D \\ &= AC + BC + C + A + B + C + D \\ &= C(A + B + 1) + A + B + D \\ X &= A + B + C + D \end{aligned}$$

O circuito simplificado é uma porta OR de 4 entradas como mostra a Figura 5–13.

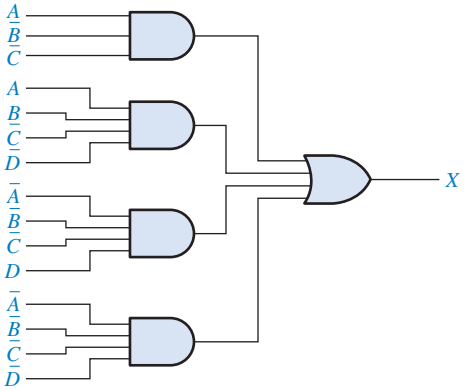


▶ **FIGURA 5–13**

Problema relacionado Verifique a minimização da expressão $A + B + C + D$ usando um mapa de Karnaugh.

EXEMPLO 5–6

Minimize o circuito lógico combinacional mostrado na Figura 5–14. Os inversores para as variáveis complementadas não são mostrados.



▶ FIGURA 5–14

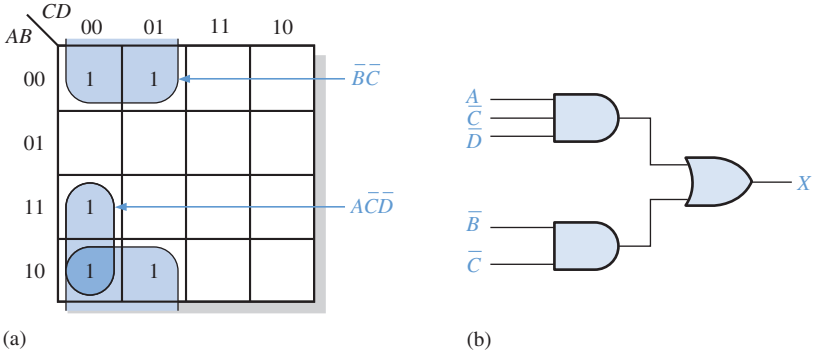
Solução A expressão de saída é:

$$X = A\overline{B}\,\overline{C} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

Expandindo o primeiro termo para incluir as variáveis D e $\overline{D}$ que não aparecem.

$$\begin{aligned} X &= A\overline{B}\,\overline{C}(D + \overline{D}) + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \\ &= A\overline{B}\,\overline{C}D + A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \end{aligned}$$

Essa expressão de soma-de-produtos expandida é inserida no mapa de Karnaugh e simplificada conforme a Figura 5–15(a). A implementação simplificada é mostrada na parte (b). Os inversores não são mostrados.



▲ FIGURA 5–15

Problema relacionado Desenvolva a expressão de produto-de-somas equivalente ao circuito mostrado na Figura 5–15(b).

SEÇÃO 5–2 REVISÃO

1. Implemente as seguintes expressões Booleana conforme elas se apresentam:
   (a) $X = ABC + AB + AC$ **(b)** $X = AB(C + DE)$
2. Desenvolva um circuito lógico que produza um 1 na sua saída apenas quando todas as três entradas são 1s ou quando todas as três entradas são 0s.
3. Reduza os circuitos da Questão 1 para a forma de soma-de-produtos mínima.

5-3 A PROPRIEDADE UNIVERSAL DAS PORTAS NAND E NOR

Já estudamos os circuitos combinacionais implementados com portas AND, portas OR e inversores. Nesta seção, a propriedade universal da porta AND e da porta OR é discutida. A universalidade da porta AND quer dizer que ela pode ser usada como um inversor e que combinações de portas NAND podem ser usadas para implementar operações AND, OR e NOR. De forma similar, a porta NOR pode ser usada para implementar operações AND, OR, NAND e inversor (NOT).

Ao final do estudo desta seção você deverá ser capaz de:

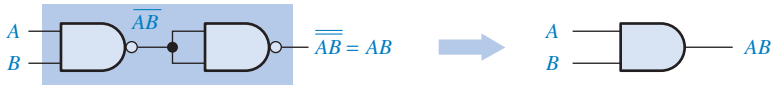
- Usar portas NAND para implementar o inversor, a porta AND, a porta OR e a porta NOR
- Usar portas NOR para implementar o inversor, a porta AND, a porta OR e a porta NAND

As portas NAND podem ser usadas para produzir qualquer função lógica.

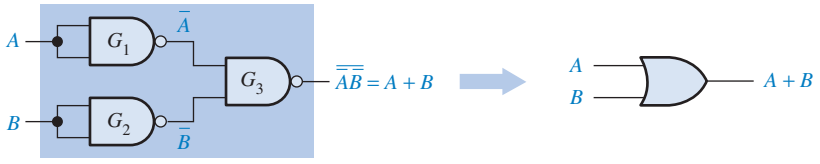
A Porta NAND como um Elemento Lógico Universal

A porta NAND é uma **porta universal** porque ela pode ser usada para produzir as funções NOT, AND, OR e NOR. Um inversor pode ser construído a partir de uma porta NAND interconectando todas as entradas e criando, de fato, uma única entrada, como mostra a Figura 5–16(a) para uma porta de 2 entradas. Uma função AND pode ser gerada a partir do uso de apenas portas NAND, como mostra a Figura 5–16(b). Uma função OR pode ser produzida com portas NAND apenas, conforme ilustra a parte (c). Finalmente, uma função NOR é produzida na parte (d).

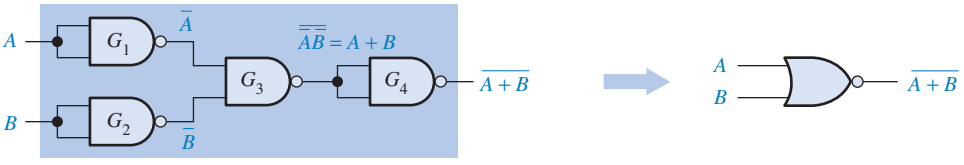
(a) Uma porta NAND usada como um inversor.



(b) Duas portas NAND usadas como uma porta AND.



(c) Três portas NAND usadas como uma porta OR.



(d) Quatro portas NAND usadas como uma porta NOR.

▶ FIGURA 5–16

Aplicação universal de portas NAND. Abra o arquivo F05-16(a), (b), (c) e (d) para verificar cada uma das equivalências.

Na Figura 5–16(b), uma porta NAND é usada para inverter (complementar) a saída de uma NAND para se obter uma função AND, conforme indicado na equação a seguir:

$$X = \overline{\overline{AB}} = AB$$

Na Figura 5-16(c), as portas NAND G_1 e G_2 são usadas para inverter as duas variáveis de entrada antes que elas sejam aplicadas na porta NAND G_3. A saída final da OR é obtida com a aplicação do teorema de DeMorgan:

$$X = \overline{\overline{A}\,\overline{B}} = A + B$$

Na Figura 5-16 (d), a porta NAND G_4 é usada como um inversor conectado ao circuito da parte (c) para produzir a operação NOR $\overline{A + B}$.

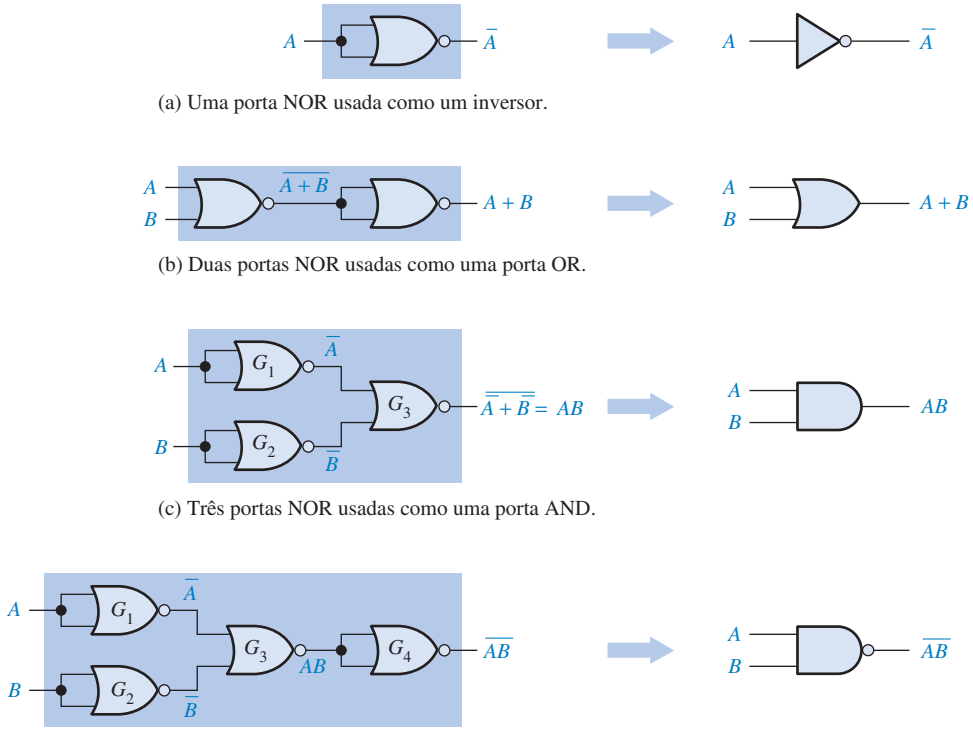
A Porta NOR como um Elemento Lógico Universal

As portas NOR podem ser usadas para produzir qualquer função lógica.

Assim como a porta NAND, a porta NOR pode ser usada para produzir as funções NOT, AND, OR e NAND. Um circuito NOT, ou inversor, pode ser feito a partir de uma porta NOR interconectando todas as entradas para criar efetivamente uma única entrada, como mostra a Figura 5–17(a) com um exemplo de 2 entradas. Além disso, uma porta OR pode ser produzida a partir de portas NOR, como ilustrado na Figura 5–17(b). Uma porta AND pode ser construída usando portas NOR, como mostra a Figura 5–17(c). Nesse caso as portas NOR G_1 e G_2 são usadas como inversores e a saída final é obtida a partir do uso do teorema de DeMorgan da seguinte forma:

$$X = \overline{\overline{A} + \overline{B}} = AB$$

A Figura 5–17(d) mostra como portas NOR são usadas para se obter uma função NAND.



(a) Uma porta NOR usada como um inversor.

(b) Duas portas NOR usadas como uma porta OR.

(c) Três portas NOR usadas como uma porta AND.

(d) Quatro portas NOR usadas como uma porta NAND.

◀ **FIGURA 5–17**

Aplicação universal de portas NOR. Abra os arquivos F05-17(a), (b), (c) e (d) para verificar cada uma das equivalências.

SEÇÃO 5–3 REVISÃO

1. Use portas NAND para implementar cada expressão a seguir:

(a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$

2. Use portas NOR para implementar cada expressão a seguir:

(a) $X = \overline{A} + B$ **(b)** $X = A\overline{B}$

5-4 LÓGICA COMBINACIONAL USANDO PORTAS NAND E NOR

Nesta seção, veremos como portas NAND e NOR podem ser usadas para implementar uma função lógica. Lembre-se, do Capítulo 3, de que a porta NAND também exibe uma operação equivalente denominada OR negativa e que a porta NOR exibe uma operação equivalente denominada AND negativa. Veremos como usar os símbolos apropriados para representar as operações equivalentes que tornam a "leitura" de um diagrama lógico mais fácil.

Ao final do estudo desta seção você deverá ser capaz de:

■ Usar portas NAND para implementar uma função lógica ■ Usar portas NOR para implementar uma função lógica ■ Usar o símbolo dual apropriado num diagrama lógico

Lógica NAND

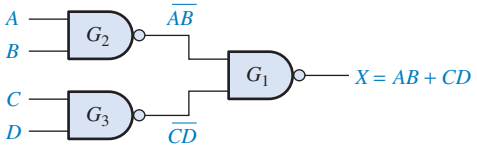
Como já estudamos, uma porta NAND pode funcionar como uma NAND ou uma OR negativa porque, pelo teorema de DeMorgan,

$$\overline{AB} = \overline{A} + \overline{B}$$

NAND ↑ ↑ OR negativa

Considere a lógica NAND na Figura 5–18. A expressão de saída é desenvolvida nos seguintes passos:

$$\begin{aligned} X &= \overline{(\overline{AB})(\overline{CD})} \\ &= \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} \\ &= (\overline{\overline{A} + \overline{B}}) + (\overline{\overline{C} + \overline{D}}) \\ &= \overline{\overline{A}}\,\overline{\overline{B}} + \overline{\overline{C}}\,\overline{\overline{D}} \\ &= AB + CD \end{aligned}$$

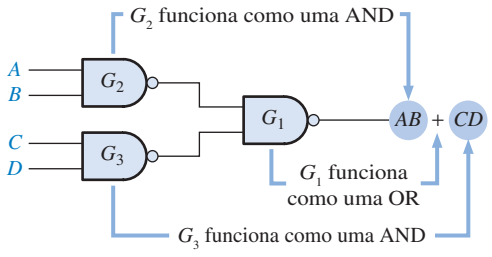


▶ **FIGURA 5–18**
Lógica NAND para $X = AB + CD$.

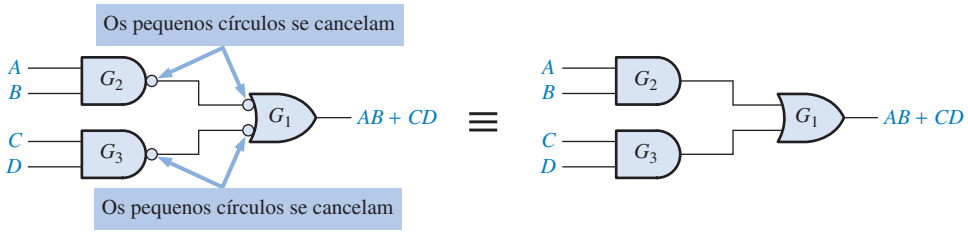
Como podemos ver na Figura 5–18, a expressão de saída, $AB + CD$, está na forma de uma OR entre o resultado de dois termos AND. Isso mostra que as portas G_2 e G_3 funcionam como portas AND e que a porta G_1 funciona como uma porta OR, conforme ilustrado na Figura 5–19(a). Esse circuito aparece redesenhado na parte (b) com símbolos NAND para as portas G_2 e G_3 e o símbolo de uma OR negativa para a porta G_1.

Observe na Figura 5–19(b) as conexões de um pequeno círculo para outro entre as saídas das portas G_2 e G_3 e as entradas da porta G_1. *Como um pequeno círculo representa uma inversão, dois pequenos círculos interconectados representam uma dupla inversão e portanto um cancela o outro*. Esse cancelamento da inversão pode ser visto no desenvolvimento anterior da expressão de saída $AB + CD$ e é indicado pela ausência de termos barrados na expressão de saída. Assim, o circuito na Figura 5–19(b) é *efetivamente* um circuito AND-OR, como mostra a Figura 5–19(c).

Diagrama Lógico NAND Usando Simbologia Dual Todos os diagramas lógicos usando portas NAND devem ser desenhados com cada porta representada pelo símbolo de uma NAND ou pelo símbolo da porta equivalente (OR negativa) para refletir a operação da porta dentro do circuito lógico. O símbolo da NAND e o símbolo da **OR negativa** são denominados de *símbolos duais*. Quando desenhamos um diagrama lógico com portas NAND, sempre usamos os símbolos da por-



(a) Diagrama lógico NAND original mostrando a operação efetiva das portas em relação à expressão de saída.



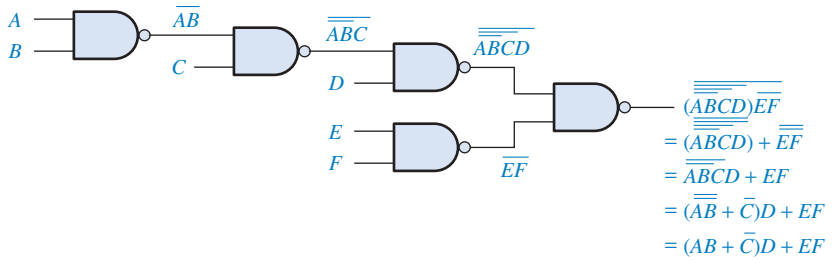
(b) Diagrama lógico equivalente NAND/OR negativa.

(c) Equivalente AND/OR.

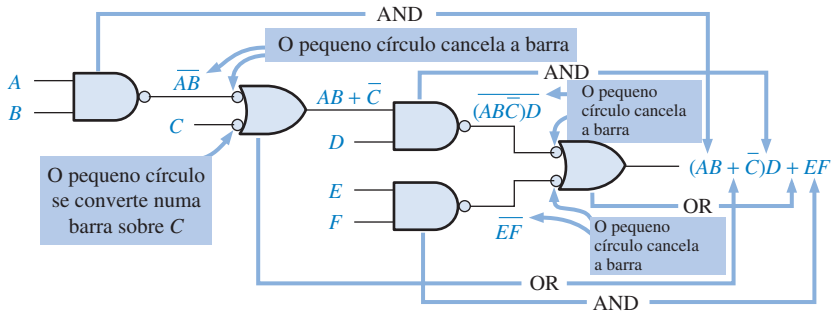
◀ **FIGURA 5–19**
Desenvolvimento do circuito AND-OR equivalente ao circuito dado na Figura 5–18.

ta de tal forma que cada conexão entre a saída de uma porta e a entrada de outra tenha dois pequenos círculos nas extremidades ou então a ausência deles em ambas extremidades. Num diagrama lógico uma saída com um pequeno círculo não pode ser conectada a uma entrada sem o pequeno círculo ou vice-versa.

A Figura 5-20 mostra um arranjo de portas que ilustra o procedimento do uso apropriado de símbolos duais para um circuito NAND com várias seqüências de portas. Embora o uso de todos os símbolos NAND no circuito mostrado na Figura 5–20(a) estejam corretos, o diagrama na parte (b) é muito mais fácil de ser "lido" (analisado) e é o método preferido. Conforme mostra a Figura 5–20(b) a porta de saída é representada com o símbolo da OR negativa. Então o símbolo NAND é usado para as portas que vêm imediatamente antes da porta de saída e os símbolos para as portas na seqüência do circuito são alternados à medida que percorremos o circuito da saída para a entrada.



(a) Algumas etapas de operações Booleanas são necessárias para chegar à expressão final.



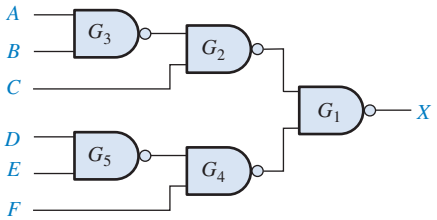
(b) A expressão de saída pode ser obtida diretamente da função de cada símbolo de porta no diagrama.

◀ **FIGURA 5–20**
Ilustração do uso apropriado de símbolos duais num diagrama lógico NAND.

O formato da porta indica a forma com que suas entradas aparecerão na expressão de saída e assim mostra qual a função da porta dentro do circuito lógico. Para um símbolo NAND, a expressão de saída mostra uma operação AND entre as entradas; e para o símbolo de uma OR negativa, a expressão de saída mostra uma operação OR entre as entradas, conforme ilustra a Figura 5–20(b). O diagrama com símbolos duais na parte (b) torna mais fácil determinar a expressão de saída diretamente a partir do diagrama lógico porque cada símbolo de porta indica a relação entre suas variáveis conforme elas aparecem na expressão de saída.

EXEMPLO 5–7

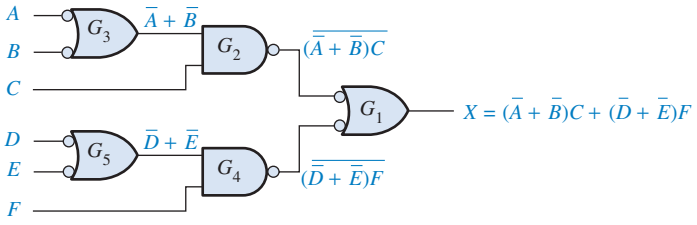
Redesenhe o diagrama lógico e desenvolva a expressão de saída para o circuito dado na Figura 5–21 usando os símbolos duais apropriados.



▶ FIGURA 5–21

Solução Redesenhe o diagrama lógico dado na Figura 5–21 fazendo uso do símbolo da porta equivalente (OR negativa) como mostra a Figura 5–22. A escrita da expressão para X diretamente a partir da operação lógica indicada para cada porta é dada por

$$X = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F.$$



▲ FIGURA 5–22

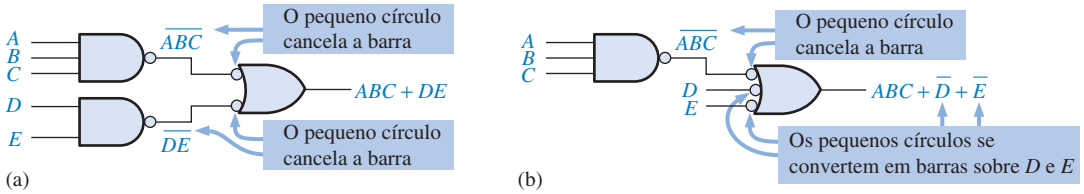
Problema relacionado Obtenha a expressão de saída a partir da Figura 5–21 e mostre que ela é equivalente à expressão dada na solução desse exemplo.

EXEMPLO 5–8

Implemente cada expressão a seguir com lógica NAND usando os símbolos duais apropriados:

(a) $ABC + DE$ **(b)** $ABC + \overline{D} + \overline{E}$

Solução Veja a Figura 5–23.



▲ **FIGURA 5–23**

Problema relacionado Converta os circuitos NAND mostrados na Figura 5–23(a) e (b) para uma lógica AND-OR equivalente.

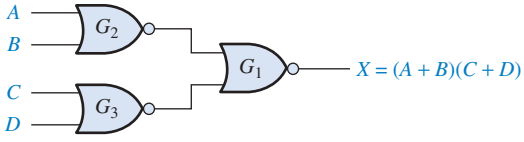
Lógica NOR

Uma porta NOR pode funcionar como uma NOR ou uma **AND negativa**, conforme mostra o teorema de DeMorgan.

$$\overline{A + B} = \overline{A}\,\overline{B}$$

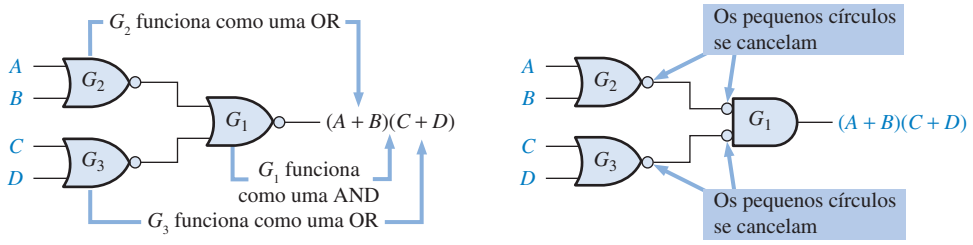
NOR — AND negativa

Considere a lógica NOR dada na Figura 5–24. A expressão de saída é desenvolvida como:



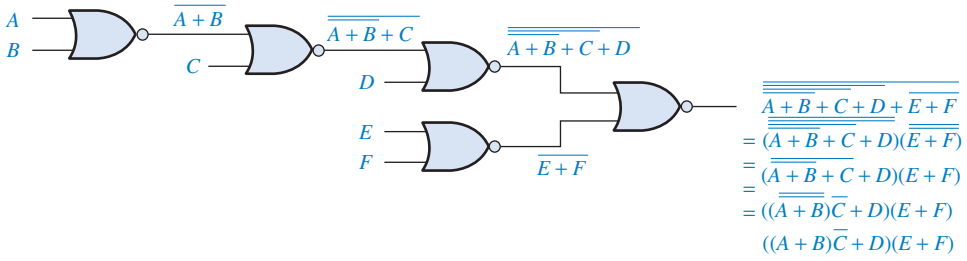
◀ **FIGURA 5–24**
Lógica NOR para $X = (A + B)(C + D)$.

Como podemos ver na Figura 5–24, a expressão de saída $(A + B)(C + D)$ consiste numa AND entre dois termos OR. Isso mostra que as portas G_2 e G_3 funcionam como portas OR e a porta G_1 funciona como uma porta AND, conforme ilustrado na Figura 5–25(a). Esse circuito aparece redesenhado na parte (b) com o símbolo de uma AND negativa para a porta G_1.

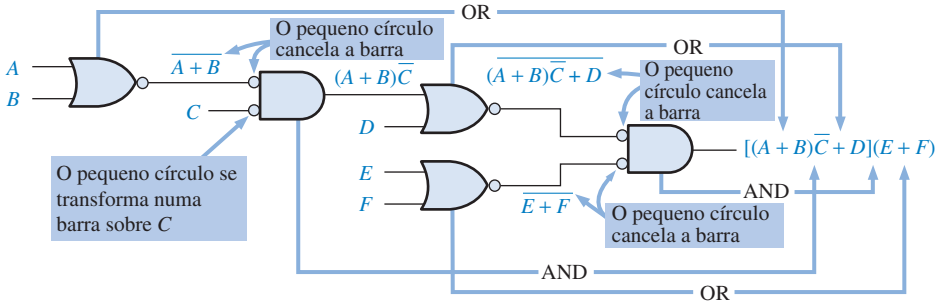


◀ **FIGURA 5–25**

Diagrama Lógico NOR Usando Símbolos Duais Assim como com a lógica NAND, a finalidade do uso de símbolos duais é tornar o diagrama lógico mais fácil de ser analisado, conforme ilustra o circuito com lógica NOR dado na Figura 5–26. Quando o circuito na parte (a) é redesenhado com símbolos duais na parte (b), observe que todas as conexões de uma saída para uma entrada possuem um par de pequenos círculos ou então ambos se mostram ausentes na conexão. Podemos ver novamente que a forma de cada símbolo de porta indica o tipo de termo (AND ou OR) que é produzido na expressão de saída, tornando a expressão de saída mais fácil de ser determinada e o diagrama lógico mais fácil de ser analisado.



(a) A expressão de saída final é obtida após várias operações Booleanas.



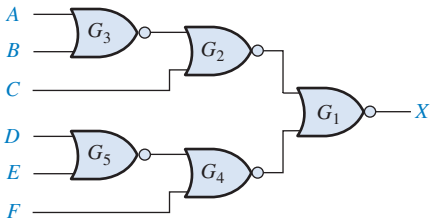
(b) A expressão de saída pode ser obtida diretamente a partir da função de cada símbolo lógico no diagrama.

▶ **FIGURA 5–26**

Ilustração do uso apropriado de símbolos duais num diagrama lógico NOR.

EXEMPLO 5–9

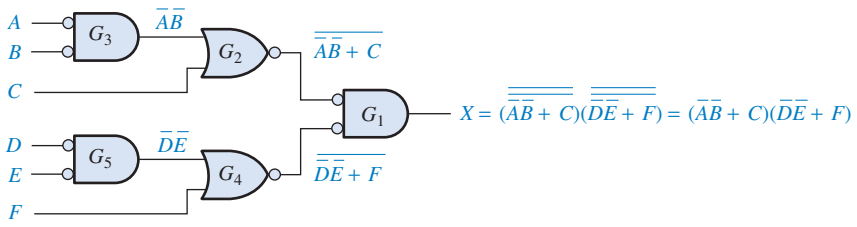
Usando símbolos duais apropriados, redesenhe o diagrama lógico e desenvolva a expressão de saída para o circuito mostrado na Figura 5–27.



▶ **FIGURA 5–27**

Solução Redesenhe o diagrama lógico com os símbolos equivalentes (AND negativa) conforme mostra a Figura 5–28. Escreva a expressão para X diretamente a partir da operação indicada de cada porta,

$$X = (\overline{A}\,\overline{B} + C)(\overline{D}\,\overline{E} + F)$$



▶ **FIGURA 5–28**

Problema relacionado Prove que a saída do circuito NOR dado na Figura 5–27 é a mesma que a saída do circuito dado na Figura 5–28.

SEÇÃO 5–4 REVISÃO

1. Implemente a expressão $X = \overline{(\overline{A} + \overline{B} + \overline{C})DE}$ usando a lógica NAND.
2. Implemente a expressão $X = \overline{\overline{A}\,\overline{B}\,\overline{C} + (D + E)}$ usando a lógica NOR.

5-5 OPERAÇÃO DE CIRCUITOS LÓGICOS COM FORMAS DE ONDA DIGITAIS NAS ENTRADAS

Vários exemplos de circuitos lógicos combinacionais, em geral com formas de onda digitais nas entradas, são analisados nessa seção. Tenha em mente que a operação de cada porta é a mesma para formas de onda digitais nas entradas assim como para níveis constantes nas entradas. A saída de um circuito lógico num dado instante depende das entradas no instante em particular, de forma que a relação temporal nas entradas é de fundamental importância.

Ao final do estudo desta seção você deverá ser capaz de:

- Analisar circuitos lógicos combinacionais com formas de onda digitais nas entradas
- Desenvolver um diagrama de temporização para qualquer circuito lógico combinacional dado com entradas especificadas

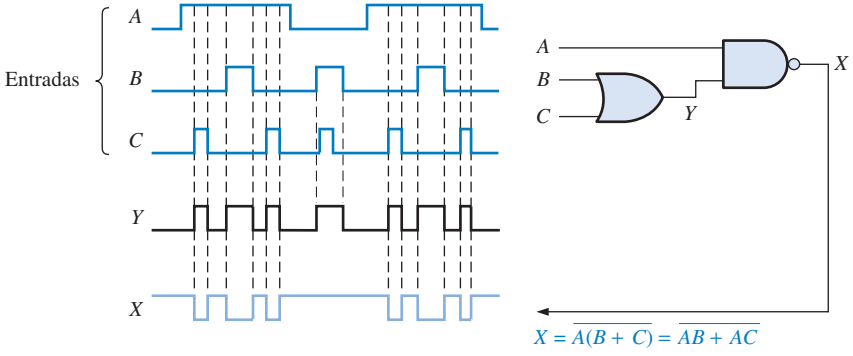
A operação de qualquer porta é a mesma independente se as entradas são pulsantes ou de níveis constantes. A natureza das entradas (pulsantes ou de níveis constantes) não altera a tabela-verdade de um circuito. Os exemplos nessa seção ilustram a análise de circuitos lógicos combinacionais com formas de onda de entrada digitais.

Os tópicos a seguir são uma revisão da operação individual das portas para uso na análise de circuitos combinacionais com formas de onda digitais de entrada:

1. A saída de uma porta AND é nível ALTO apenas quando todas as entradas estiverem em nível ALTO ao mesmo tempo.
2. A saída de uma porta OR é nível ALTO apenas quando pelo menos uma de suas entradas estiver em nível ALTO.
3. A saída de uma porta NAND é nível BAIXO apenas quando todas as entradas estiverem em nível ALTO ao mesmo tempo.
4. A saída de uma porta NOR é nível BAIXO apenas quando pelo menos uma de suas entradas estiver em nível ALTO.

EXEMPLO 5–10

Determine a forma de onda de saída final X para o circuito mostrado na Figura 5–29 tendo as formas de onda das entradas A, B e C conforme mostrado.



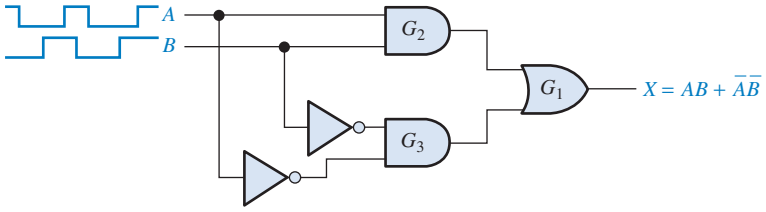
▲ FIGURA 5–29

Solução A expressão de saída, $\overline{AB + AC}$, indica que a saída X é nível BAIXO quando as entradas A e B estão em nível ALTO ou quando as entradas A e C estão em nível ALTO ou quando todas as entradas estão em nível ALTO. A forma de onda na saída X é mostrada no diagrama de temporização visto na Figura 5–29. A forma de onda intermediária Y na saída da porta OR também é mostrada.

Problema relacionado Determine a forma de onda de saída se a entrada A estiver constantemente em nível ALTO.

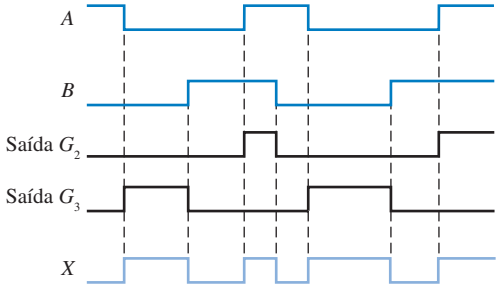
EXEMPLO 5–11

Desenhe o diagrama de temporização para o circuito mostrado na Figura 5–30 mostrando as saídas de G_1, G_2 e G_3 com as formas de onda de entrada A e B conforme indicado.



▲ **FIGURA 5–30**

Solução Quando as duas entradas estão em nível ALTO ou quando as duas entradas estão em nível BAIXO, a saída X é nível ALTO conforme mostra a Figura 5–31. Observe que esse é um circuito de uma EX-NOR. As saídas intermediárias das portas G_1 e G_3 também são mostradas na Figura 5–31.

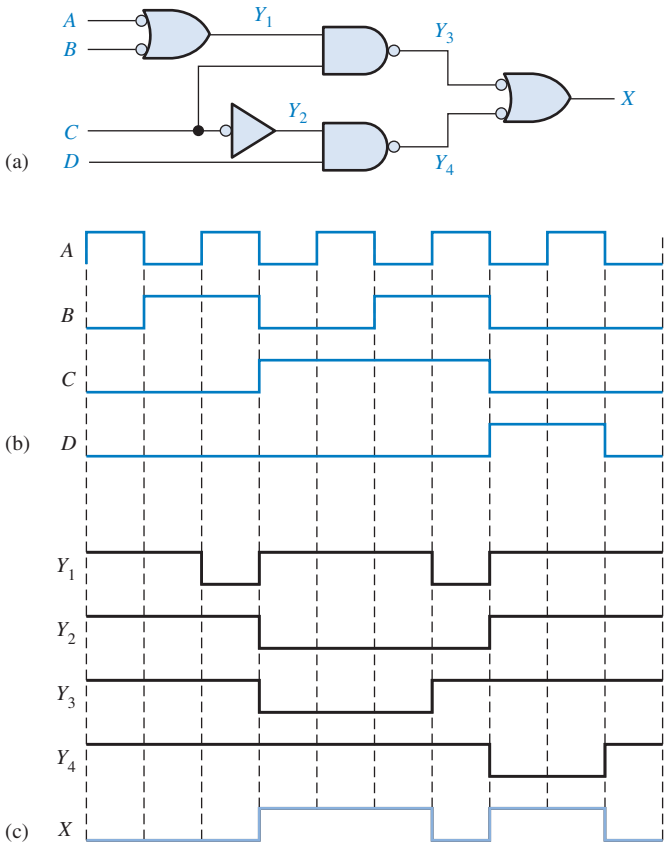


▲ **FIGURA 5–31**

Problema relacionado Determine a saída X do circuito mostrado na Figura 5–30 se a entrada B for invertida.

EXEMPLO 5–12

Determine a forma de onda na saída X para o circuito lógico visto na Figura 5–32(a) determinando primeiro as formas de onda intermediárias em Y_1, Y_2, Y_3 e Y_4. As formas de onda de entrada são mostradas na Figura 5–32(b).



▶ **FIGURA 5–32**

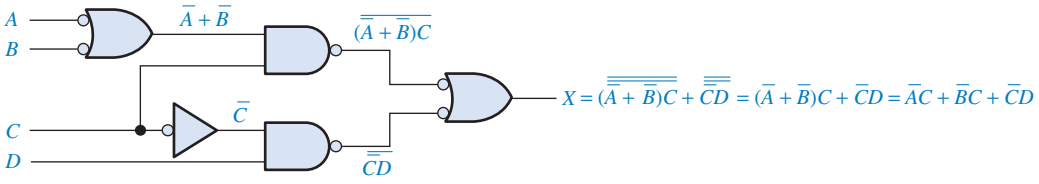
Solução Todas as formas de onda intermediárias e a forma de onda na saída final são mostradas no diagrama de temporização visto na Figura 5–32(c).

Problema relacionado Determine as formas de onda Y_1, Y_2, Y_3, Y_4 e X se a forma de onda da entrada A for invertida.

EXEMPLO 5–13

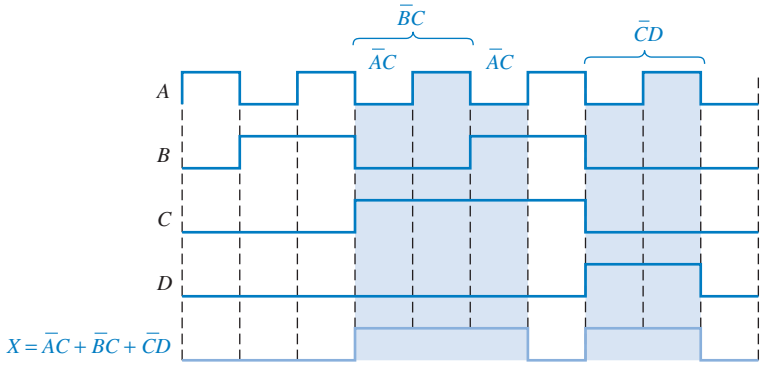
Determine a forma de onda da saída X para o circuito mostrado no Exemplo 5–12, Figura 5–32(a), diretamente a partir da expressão de saída.

Solução A expressão de saída para o circuito é desenvolvida na Figura 5–33. A forma de soma-de-produtos indica que a saída é nível ALTO quando A é nível BAIXO e C é nível ALTO ou quando B é nível BAIXO e C é nível ALTO ou quando C é nível BAIXO e D é nível ALTO.



▲ **FIGURA 5–33**

O resultado é mostrado na Figura 5–34 e é o mesmo que foi obtido pelo método da forma de onda intermediária no Exemplo 5–12. Os termos-produto correspondentes para cada condição da forma de onda que resulta numa saída de nível ALTO são indicados.



▲ FIGURA 5–34

Problema relacionado Repita esse exemplo considerando que todas as formas de onda sejam invertidas.

SEÇÃO 5–5 REVISÃO

1. Um pulso de largura $t_W = 50\ \mu s$ é aplicado numa das entradas de um circuito EX-OR. Um segundo pulso positivo de largura $t_W = 10\ \mu s$ é aplicado na outra entrada começando 15 μs após a borda de subida do primeiro pulso. Mostre a como é a saída em relação às entradas.
2. As formas de onda dos pulsos *A* e *B* na Figura 5–29 são aplicado no circuito EX-OR visto na Figura 5–30. Desenvolva o diagrama de temporização completo.

5-6 LÓGICA COMBINACIONAL COM VHDL (Opcional)

A finalidade da descrição de um circuito lógico usando um programa VHDL é que ele possa ser programado num PLD. A abordagem de fluxo de dados na escrita de um programa VHDL foi feita no Capítulo 4. Nessa seção opcional, tanto a abordagem de fluxo de dados usando expressões Booleanas quanto a abordagem estrutural são usadas para desenvolver o código VHDL na descrição de circuitos lógicos. O componente VHDL é introduzido e usado para ilustrar descrições estruturais. Alguns aspectos das ferramentas de desenvolvimento de software são discutidos.

Ao final do estudo desta seção você deverá ser capaz de:

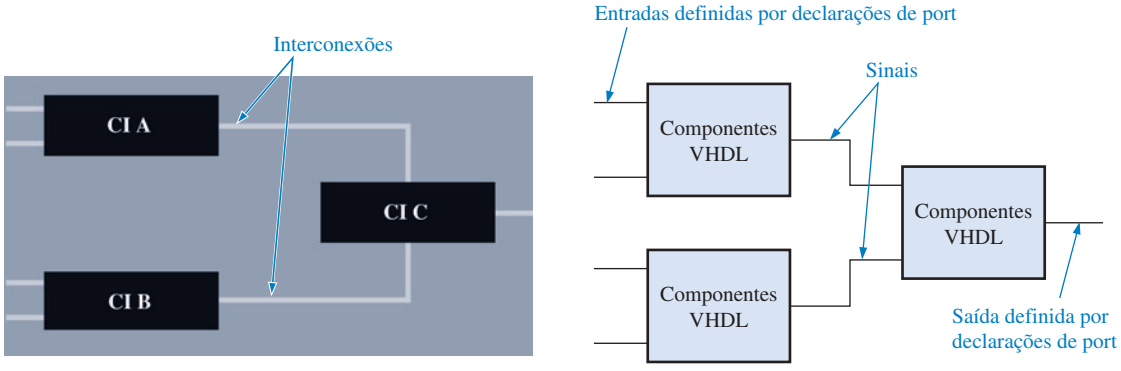
■ Descrever um componente VHDL e discutir como ele é usado num programa ■ Aplicar a abordagem estrutural e a abordagem de fluxo de dados para escrever um código VHDL ■ Descrever duas ferramentas de desenvolvimento de software básicas

Abordagem Estrutural para Programação VHDL

A abordagem estrutural para escrever uma descrição VHDL de uma função lógica pode ser comparada à instalação de CIs numa placa de circuito e interconectá-los com fios. Com a abordagem estrutural, descrevemos funções lógicas e especificamos como elas são interconectadas. O com-

ponente VHDL é uma forma de predefinir uma função lógica para uso repetido num programa ou em outros programas. O componente pode ser usado para descrever qualquer coisa desde uma simples porta lógica até uma função lógica complexa. O **sinal** VHDL pode ser entendido como uma forma de especificar uma conexão de "fio" entre componentes.

A Figura 5–35 apresenta uma comparação simplificada da abordagem estrutural para a implementação de um hardware numa placa de circuito.



(a) Implementação de hardware com lógica de funções fixas

(b) Implementação estrutural VHDL

▲ **FIGURA 5–35**

Comparação simplificada de uma abordagem estrutural VHDL para a implementação de um hardware. Os sinais VHDL correspondem às interconexões na placa de circuito e os componentes VHDL correspondem ao CIs.

Componentes VHDL

Um componente VHDL descreve uma lógica predefinida que pode ser armazenada como um conjunto de declarações numa biblioteca VHDL e utilizada num programa quantas vezes forem necessárias. Podemos usar componentes para evitar a repetição do mesmo código repetidas vezes dentro de um programa. Por exemplo, podemos criar um componente VHDL para uma porta AND e então usá-la todas as vezes que desejarmos sem ter que escrever um programa para uma porta AND sempre que precisarmos de uma.

Os componentes VHDL são armazenados e disponibilizados para uso quando escrevemos um programa. Isso é similar a termos, por exemplo, uma caixa de CIs disponível quando construímos um circuito. Todas as vezes que precisarmos usar um no nosso circuito, o pegamos na caixa e colocamos na nossa placa de circuito.

O programa VHDL para qualquer função lógica pode se tornar um componente e ser usado sempre que for necessário num programa maior com o uso de uma declaração de componente na forma geral mostrada logo a seguir. A palavra **component** em VHDL é uma palavra reservada.

```
component name_of_component is
port (port definitions);
end component name_of_component;
```

Por questão de simplicidade, vamos considerar que existem descrições de fluxo de dados VHDL predefinidas de uma porta AND de 2 entradas com o nome de entidade AND_gate e uma porta OR de 2 entradas com o nome de entidade OR_gate, conforme mostra a Figura 5–36.

Em seguida, consideremos que estamos escrevendo um programa para um circuito lógico que tem várias portas AND. Em vez de reescrever repetidas vezes o programa dado na Figura 5–36, podemos usar uma declaração de componente para especificar a porta AND. A declaração de port

Porta AND de 2 entradas

```
entity AND_gate is
  port (A, B: in bit; X: out bit);
end entity AND_gate;

architecture ANDfunction of AND_gate is
begin
  X <= A and B;
end architecture ANDfunction;
```

Porta OR de 2 entradas

```
entity OR_gate is
  port (A, B: in bit; X: out bit);
end entity OR_gate;

architecture ORfunction of OR_gate is
begin
  X <= A or B;
end architecture ORfunction;
```

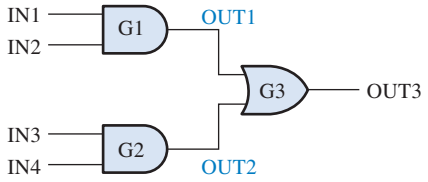
▶ **FIGURA 5–36**

Programas predefinidos para uma porta AND de 2 entradas e uma porta OR de 2 entradas para serem usados como componentes na abordagem de fluxo de dados.

na declaração de componente tem que corresponder à declaração de port na declaração de entidade da porta AND.

```
component AND_gate is
  port (A, B: in bit; X: out bit);
end component AND_gate;
```

Usando Componentes num Programa Para usar um componente num programa, temos que escrever uma declaração instancial do componente para cada instância na qual o componente é usado. Podemos pensar num componente instancial como uma requisição ou pedido do componente a ser usado no programa principal. Por exemplo, o circuito lógico de soma-de-produtos mostrado na Figura 5–37 tem duas portas AND e uma porta OR. Portanto, o programa VHDL para esse circuito terá dois componentes e três componentes instanciais ou requisições.



▶ **FIGURA 5–37**

Sinais Em VHDL, os sinais são equivalentes aos fios que interconectam componentes numa placa de circuito. Os sinais vistos na Figura 5–37 são denominados de OUT1 (saída 1) e OUT2. Os sinais são conexões *internas* no circuito lógico e são tratados diferentemente das entradas e saídas. Ao passo que as entradas e saídas são declaradas na declaração de entidade usando a declaração de port, os sinais são declarados dentro da arquitetura usando a declaração de sinal. A palavra **signal** (sinal) é uma palavra reservada em VHDL.

O Programa O programa para o circuito lógico mostrado na Figura 5–37 começa com a declaração de entidade como mostrado a seguir:

```
--Programa para o circuito lógico mostrado na Figura 5–37
entity AND_OR_Logic is
  port (IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end entity AND_OR_Logic;
```

A declaração de arquitetura contém as declarações de componentes para a porta AND e a porta OR, as definições de sinais e os componentes instanciais.

```
architecture LogicOperation of AND_OR_Logic is
  component AND_gate is
    port (A, B: in bit); X: out bit);
  end component AND_gate;

  component OR_gate is
    port (A, B: in bit; X: out bit);
  end component OR_gate;

  signal OUT1, OUT2: bit;
begin
  G1: AND_gate port map (A => IN1, B => IN2, X => OUT1);
  G2: AND_gate port map (A => IN3, B => IN4, X => OUT2);
  G3: OR_gate port map (A => OUT1, B => OUT2, X => OUT3);
end architecture LogicOperation;
```

Declaração de componente para a porta AND

Declaração de componente para a porta OR

Declaração de sinal

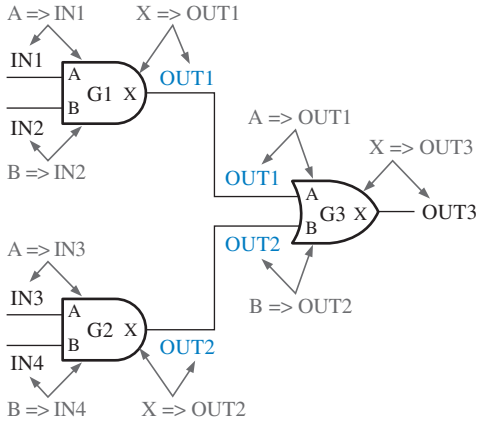
Componentes instanciais

Componentes Instanciais Vamos considerar agora os componentes instanciais. Primeiro, observe que os componentes instanciais aparecem entre as palavras reservadas da declaração **begin** (início) e **end** (fim). Para cada componente instancial um identificador é definido, tal como G1, G2 e G3 nesse caso. Em seguida o nome do componente é especificado. O mapa de port faz essencialmente todas as conexões para a função lógica usando o operador =>. Por exemplo, o primeiro componente instancial,

```
GI: AND_gate port map (A => IN1, B=> IN2. X=>OUT1)
```

pode ser explicado da seguinte forma: *A entrada A da porta AND G1 é conectada à entrada IN1, a entrada B da porta é conectada na entrada IN2 e a saída X da porta é conectadas ao sinal OUT1.*

As três declarações instanciais juntas descrevem completamente o circuito lógico mostrado na Figura 5–37, conforme ilustrado na Figura 5–38.



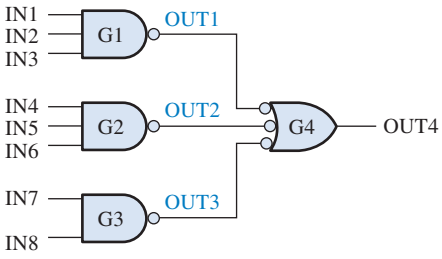
◄ FIGURA 5–38

Ilustração de declarações instanciais e mapa de port aplicado ao circuito lógico AND-OR. Os sinais estão destacados em laranja.

Embora a abordagem por fluxo de dados usando expressões Booleanas tenha sido a mais fácil e provavelmente a melhor forma de descrever esse circuito em particular, usamos esse circuito simples para explicar o conceito da abordagem estrutural. O Exemplo 5–14 compara as abordagens estrutural e por fluxo de dados na escrita de um programa VHDL para um circuito lógico de soma-de-produtos.

EXEMPLO 5–14

Escreva um programa VHDL para o circuito lógico de soma-de-produtos mostrado na Figura 5–39 usando a abordagem estrutural. Considere que os componentes VHDL para uma porta NAND de 3 entradas e para uma porta NAND de 2 entradas estejam disponíveis. Observe que a porta NAND G4 é mostrada como uma porta OR negativa.



▶ FIGURA 5–39

Solução Os componentes e componentes instanciais estão destacados.

```
--Programa para o circuito lógico mostrado na Figura 5–39
entity SOP_Logic is
   port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;

architecture LogicOperation of SOP_Logic is

-- declaração de componente para uma porta NAND de 3 entradas

  component NAND_gate3 is
     port (A, B, C: in bit X: out bit);
  end component NAND_gate3;

-- declaração de componente para uma porta NAND de 2 entradas

  component NAND_gate2 is
     port (A, B: in bit; X: out bit);
  end component NAND_gate;

   signal OUT1, OUT2, OUT3: bit;
begin
  G1: NAND_gate3 port map (A => IN1, B => IN2, C => IN3, X => OUT1);
  G2: NAND_gate3 port map (A => IN4, B => IN5, C => IN6, X => OUT2);
  G3: NAND_gate2 port map (A => IN7, B => IN8, X => OUT3);
  G4: NAND_gate3 port map (A => OUT1, B => OUT2, C => OUT3, X => OUT4);
end architecture LogicOperation;
```

Para fins de comparação, vamos escrever o programa para o circuito lógico mostrado na Figura 5–39 usando a abordagem de fluxo de dados.

```
entity SOP_Logic is
   port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit; OUT4: out bit);
end entity SOP_Logic;
```

```
architecture LogicOperation of SOP_Logic is
begin
   OUT4 <= (IN1 and IN2 and IN3) or (IN4 and IN5 and IN6) or (IN7 and IN8);
end architecture LogicOperation;
```

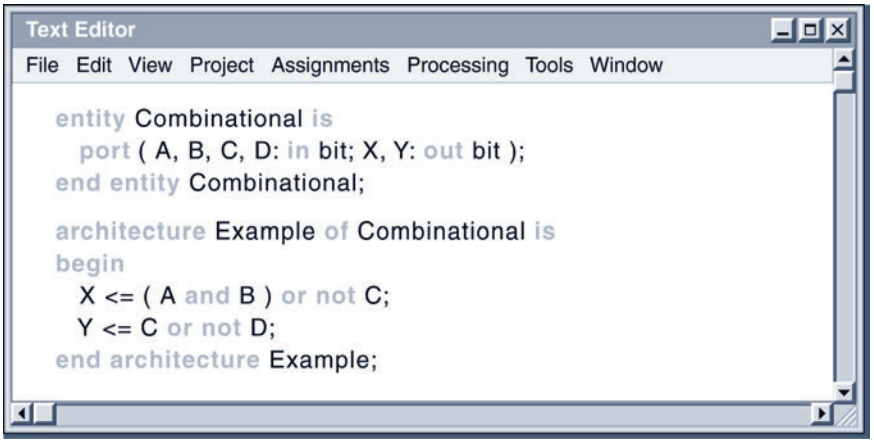
Como podemos ver, a abordagem de fluxo de dados resulta num código muito mais simples para essa função lógica em particular. Entretanto, nas situações em que uma função lógica consiste de muitos blocos de lógica complexa, a abordagem estrutural pode ter uma vantagem sobre a abordagem de fluxo de dados.

Problema relacionado Se uma outra porta NAND for acrescentada ao circuito mostrado na Figura 5–39 com entradas IN9 e IN10, escreva um componente instancial para acrescentar ao programa. Especifique qualquer outra alteração necessária no programa como resultado.

Uso de Ferramentas de Desenvolvimento (Software)

Conforme já estudamos, um pacote de desenvolvimento de software tem que ser usado para implementar um projeto HDL num dispositivo programável. Uma vez descrito o circuito lógico usando uma linguagem HDL e inserido via uma ferramenta de software denominada de editor de código ou texto, ele pode ser testado usando uma simulação para verificar se ele funciona adequadamente antes da programação do dispositivo. O uso de ferramentas de desenvolvimento de software permite realizar o projeto, o desenvolvimento e o teste do circuito lógico combinacional antes que ele seja implementado no hardware. As ferramentas de desenvolvimento de software serão mais exploradas no Capítulo 11.

As ferramentas de desenvolvimento de software típicas nos permitem inserir o código VHDL num editor de texto específico para uma ferramenta de desenvolvimento particular que estamos usando. A Figura 5–40 mostra, para fins de ilustração, a tela de um computador com o código VHDL para um circuito lógico combinacional que foi escrito usando um editor de texto (*text editor*). De acordo com o mostrado na figura, muitos editores de código possuem características melhoradas tais como o destaque das palavras reservadas.

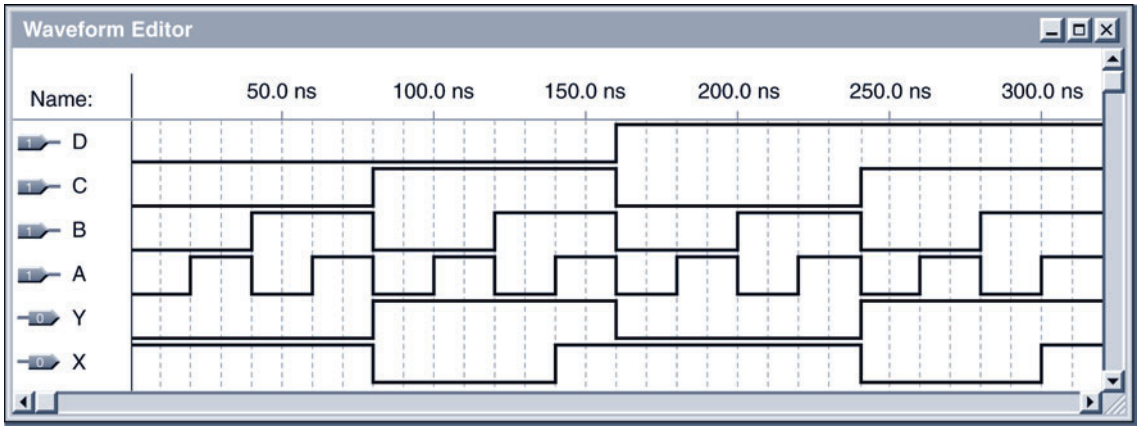


▲ **FIGURA 5–40**

Um programa VHDL para um circuito lógico combinacional após ser inserido num editor de texto genérico que é parte de uma ferramenta de desenvolvimento de software.

Após o programa ser escrito no editor de texto, passa pelo compilador. O compilador pega o código VHDL de alto nível e o converte num arquivo que pode ser transferido para o dispositivo programável. Uma vez compilado o programa, podemos criar uma simulação para testá-lo. Os valores de entrada simulados são inseridos no projeto lógico e permitem a verificação da(s) saída(s).

Especificamos as formas de onda de entrada numa ferramenta denominada de editor de forma de onda, conforme mostra a Figura 5–41. As formas de onda de saída são geradas por uma simulação do código VHDL que inserimos no editor de texto mostrado na Figura 5–40. A simulação por formas de onda fornece as saídas *X* e *Y* resultantes para as entradas *A*, *B*, *C* e *D* em todas as dezesseis combinações desde 0000_2 até 1111_2.



▲ **FIGURA 5–41**

Uma ferramenta de editor de forma de onda típico mostrando as formas de onda simuladas para o circuito lógico descrito pelo código VHDL mostrado na Figura 5–40.

Lembre-se do Capítulo 3 que existem várias características de desempenho dos circuitos lógicos a serem consideradas na criação de qualquer sistema digital. O atraso de propagação, por exemplo, determina a velocidade ou a freqüência na qual um circuito lógico pode operar. Uma simulação de temporização pode ser usada para imitar o atraso de propagação através do projeto lógico no dispositivo programável (destino).

SEÇÃO 5–6 REVISÃO

1. O que é um componente VHDL?
2. Qual é a finalidade de um componente instancial na arquitetura de um programa?
3. Como são feitas as interconexões entre os componentes em VHDL?
4. O uso de componentes em programas VHDL representa qual abordagem?

5-7 ANÁLISE DE DEFEITO

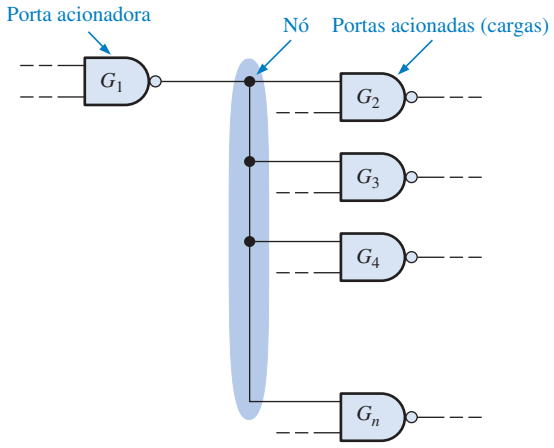
As seções anteriores abordaram superficialmente a operação de circuitos lógicos combinacionais e as relações de entradas e saídas. Esse tipo de entendimento é essencial quando fazemos análise de defeito em circuitos digitais porque temos que saber quais níveis lógicos ou formas de onda esperar ao longo do circuito para um dado conjunto de condições de entrada.

Nesta seção, usamos um osciloscópio para fazer análise de defeito num circuito lógico de funções fixas quando a saída de uma porta é conectada às entradas de várias portas. Além disso, apresentamos em um exemplo um método de análise de forma de onda e rastreamento de sinal usando um osciloscópio ou um analisador lógico para a identificação de defeito num circuito lógico.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir um nó de circuito ■ Usar um osciloscópio para determinar um defeito num nó de circuito ■ Usar um osciloscópio para determinar uma saída de porta aberta ■ Usar um osciloscópio para determinar uma saída ou entrada de porta em curto-circuito ■ Usar um osciloscópio ou analisador lógico para rastrear um sinal num circuito lógico combinacional

Num circuito lógico combinacional, a saída de uma porta pode ser conectada a duas ou mais entradas de portas conforme mostra a Figura 5–42. As interconexões compartilham um ponto elétrico em comum conhecido como **nó**.



◀ **FIGURA 5–42**
Ilustração de um nó num circuito lógico.

A porta G_1 na Figura 5–42 está acionando o nó e as outras portas representam cargas conectadas ao nó. Uma porta pode acionar um determinado número de cargas (entradas de portas) até um valor máximo especificado pelo fan-out. É possível que ocorra vários tipos de defeitos nessa situação. Alguns desses defeitos são difíceis de isolar a uma única porta com defeito porque todas as portas conectadas ao nó são afetadas. Veja alguns tipos comuns de defeitos:

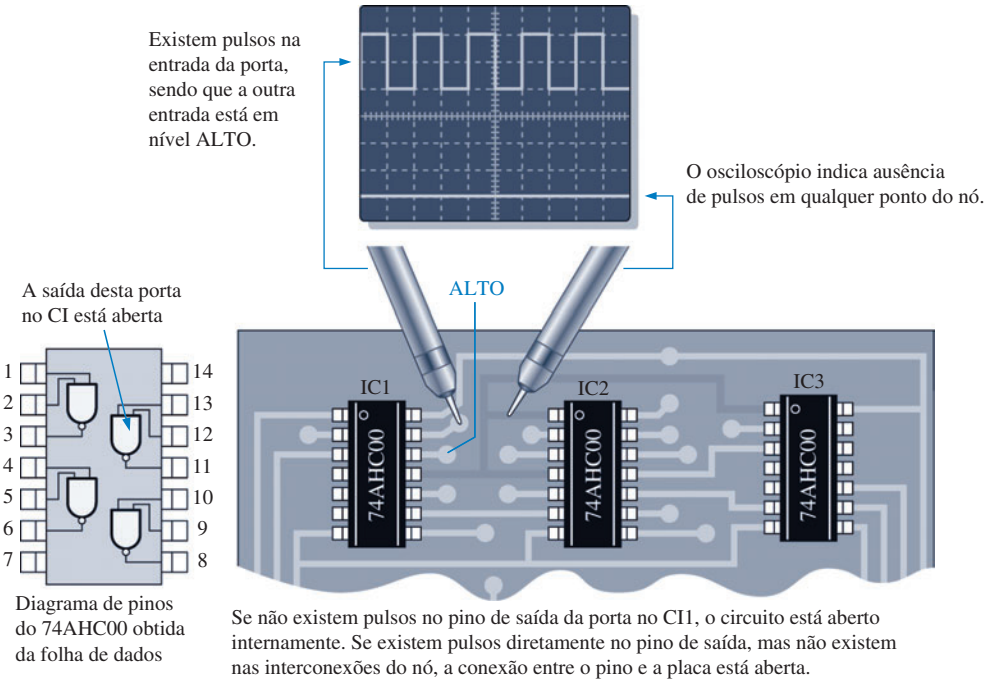
1. *Saída da porta acionadora aberta*. Esse defeito provoca uma perda de sinal em todas as portas que são cargas.
2. *Entrada aberta numa porta que funciona como carga*. Esse defeito não afeta a operação de qualquer uma das outras portas conectadas ao nó, mas resulta na perda de sinal na saída da porta com defeito.
3. *Porta acionadora com saída em curto-circuito*. Esse defeito pode fazer com que o nó fique preso ao estado BAIXO (curto-circuito para GND) ou no estado ALTO (curto-circuito para V_{CC}).
4. *Entrada em curto-circuito de uma porta que funciona como carga*. Esse defeito também pode fazer com que o nó fique preso ao estado BAIXO (curto-circuito para GND) ou ao estado ALTO (curto-circuito para V_{CC}).

Defeitos Comuns em Análise de Defeito

Saída Aberta na Porta Acionadora Nessa situação não existem pulsos no nó. Com o circuito energizado, um nó aberto normalmente resulta num nível flutuante, que freqüentemente é indicado pelo ruído, conforme ilustrado na Figura 5–43.

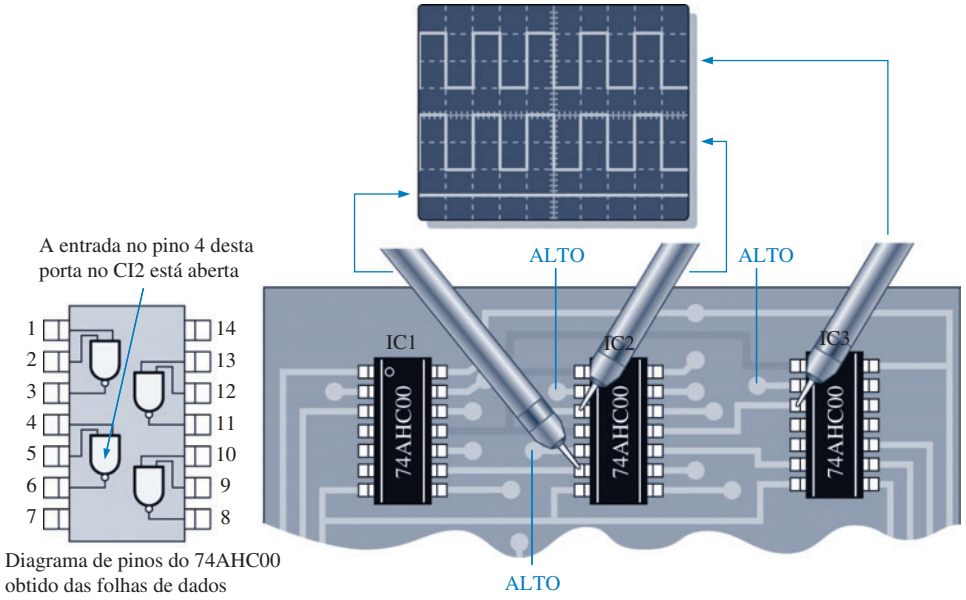
DICA PRÁTICA

Ao fazer análise de defeito em circuitos lógicos, comece com uma verificação visual, procurando por problemas óbvios. Além da observação dos componentes, faça uma inspeção visual também nos conectores. Conectores de borda são usados freqüentemente para levar alimentação, GND e sinais para a placa de circuito. As superfícies de contato do conector precisam ser limpas e ter um bom ajuste mecânico. Um conector sujo pode provocar um defeito constante ou intermitente no circuito. Conectores de borda podem ser limpos com uma borracha de lápis comum e com um pano embebido em álcool. Além disso, todos os conectores devem ser verificados quanto a folgas nos pinos.



▲ **FIGURA 5–43**

Saída aberta numa porta acionadora. Por questão de simplicidade, considere um nível ALTO na entrada da porta.

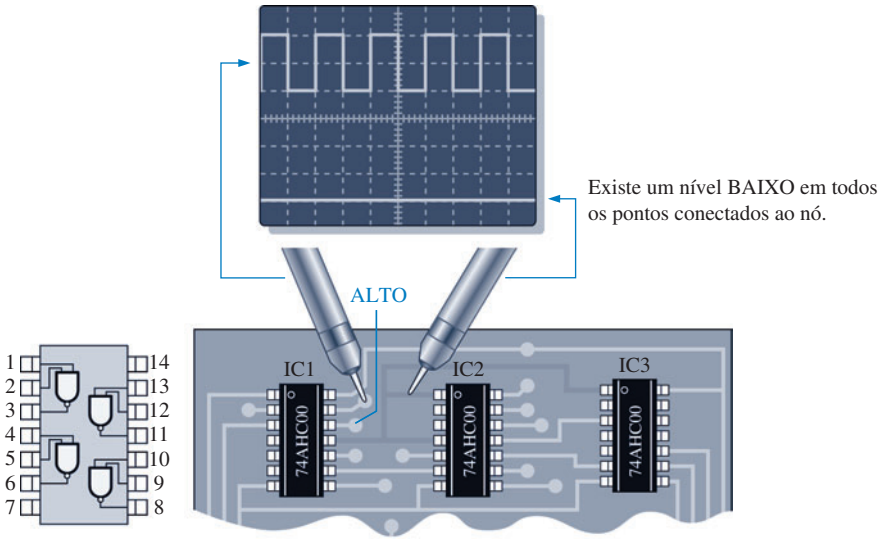


▲ **FIGURA 5–44**

Entrada aberta numa porta que funciona como carga.

Entrada Aberta numa Porta Acionada Se a busca por uma saída aberta na porta acionadora der resultado negativo, então se deve verificar a existência de uma entrada aberta numa porta acionada. Aplique a ponta do pulsador lógico no nó com todas as outras entradas em nível ALTO. Em seguida, verifique a saída de cada porta procurando por pulsos usando uma ponta de prova lógica, conforme ilustra a Figura 5–44. Se uma das entradas que normalmente está conectada ao nó estiver aberta, nenhum pulso será detectado na saída da porta.

Saída ou Entrada em Curto-Circuito para GND Quando a saída da porta acionadora está em curto-circuito com GND ou a entrada de uma porta acionada está em curto-circuito com GND, isso faz com que o nó fique preso ao nível BAIXO, conforme mencionado anteriormente. Uma verificação rápida com a ponta de prova de um osciloscópio indica isso, conforme mostra a Figura 5–45. Um curto-circuito para GND na saída da porta acionadora ou qualquer entrada de porta acionada provoca esse sintoma, sendo que verificações posteriores devem portanto ser feitas para isolar o curto-circuito identificando a porta.



◀ **FIGURA 5–45**
Saída da porta acionadora em curto-circuito ou a entrada de uma porta acionada em curto-circuito.

Rastreamento de Sinais e Análise de Formas de Onda

Embora os métodos para isolar um curto-circuito ou circuito aberto em um nó sejam muito úteis, uma técnica mais geral de análise de defeito denominada **rastreamento de sinal** é aplicável em quase todas as situações de análise de defeito. A medição de forma de onda é realizada com o uso de um osciloscópio ou de um analisador lógico.

Basicamente, o método de rastreamento de sinal necessita que observemos as formas de onda e as suas relações temporais em todos os pontos acessíveis do circuito lógico. Podemos começar pelas entradas e, a partir da análise da forma de onda em cada ponto, determinamos onde ocorre primeiro uma forma de onda incorreta. Com esse procedimento podemos geralmente isolar o defeito a uma determinada porta. O procedimento que começa a análise a partir da saída em direção às entradas também pode ser usado.

O procedimento geral para começar o rastreamento de sinal pelas entradas é esboçado a seguir:

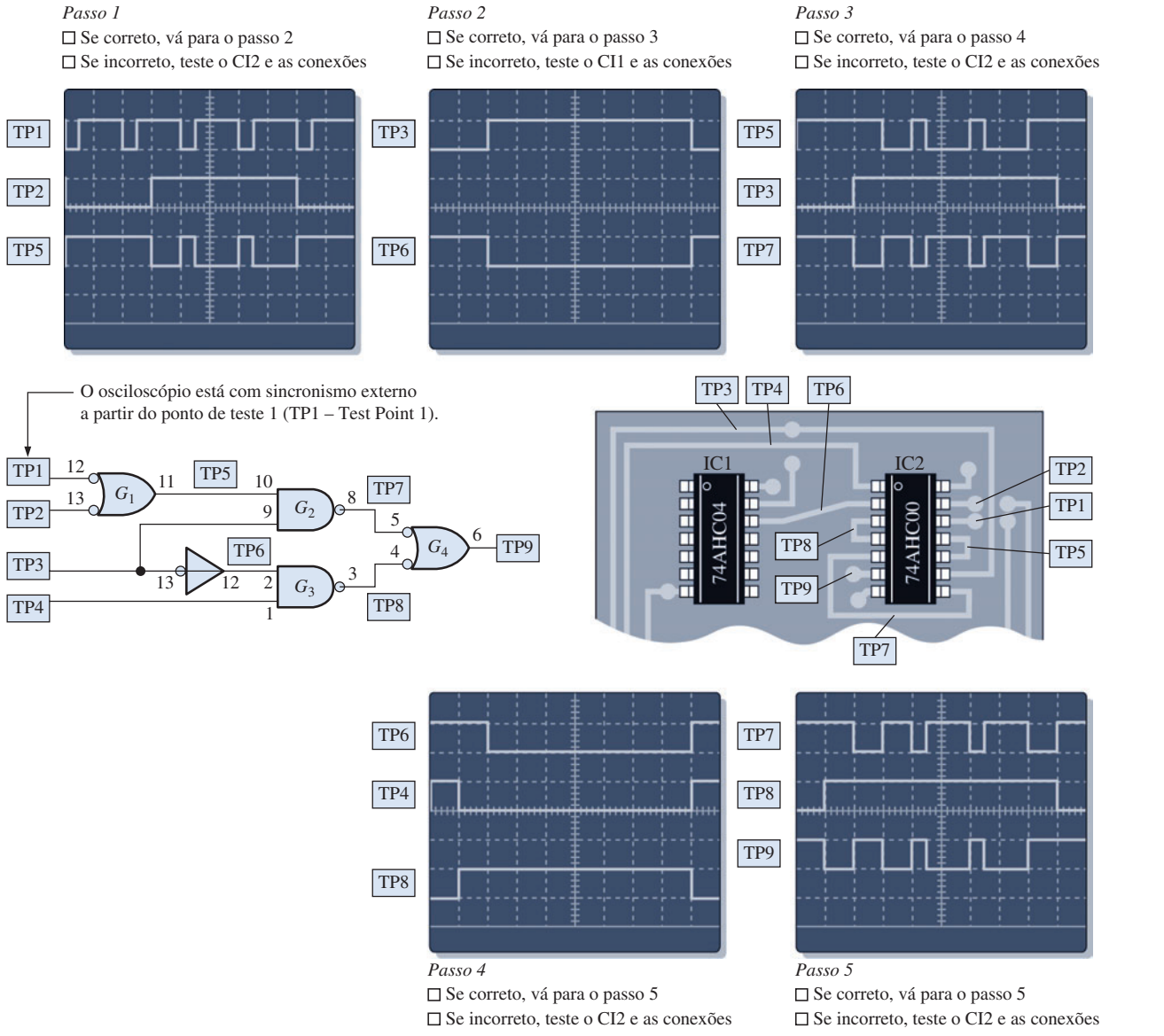
- Dentro de um sistema, defina a seção do circuito lógico suspeita de estar com defeito.
- Comece pelas entradas percorrendo o circuito lógico até a seção sob investigação. Considere, para essa discussão, que as formas de onda de entrada vêm de outras seções do sistema que estão funcionando corretamente.
- Para cada porta, comece pela entrada em direção à saída do circuito lógico, observe a forma de onda de saída da porta e a compare com a forma de onda de entrada usando um osciloscópio ou um analisador lógico.
- Determine se a forma de onda de saída está correta, usando os seus conhecimentos sobre a operação lógica da porta.
- Se a saída estiver incorreta, a porta sob teste pode estar com defeito. Retire o CI que contém a porta suspeita de estar com defeito e teste ela fora do circuito. Se for identificado que a porta está com defeito, substitua o CI. Se ela estiver funcionando corretamente, o defeito está no circuito externo ou em outro CI no qual esta porta está conectada.
- Se a saída estiver correta, passe para a próxima porta. Continue verificando cada porta até que uma forma de onda incorreta seja observada.

A Figura 5–46 é um exemplo que ilustra o procedimento geral para um circuito lógico específico conforme os passos a seguir:

Passo 1 Observe a saída da porta G_1 (ponto de teste 5) em relação às suas entradas. Se ela estiver correta, verifique em seguida o inversor. Se a saída dele não estiver correta, a porta ou suas conexões estão com problema; ou, se a saída estiver em nível BAIXO, a entrada da porta G_2 pode estar em curto-circuito.

Passo 2 Observe a saída do inversor (TP6) em relação à sua entrada. Se ela estiver correta, verifique em seguida a porta G_2. Se a saída do inversor não estiver correta, o inversor ou suas conexões estão com problema; ou, se a saída estiver em nível BAIXO, a entrada da porta G_3 pode estar em curto-circuito.

Passo 3 Observe a saída da porta G_2 (TP7) em relação às suas entradas. Se ela estiver correta, verifique em seguida a porta G_3. Se a saída não estiver correta, a porta ou suas conexões estão com problema; ou, se a saída estiver em nível BAIXO, a entrada da porta G_4 pode estar em curto-circuito.



▲ **FIGURA 5–46**

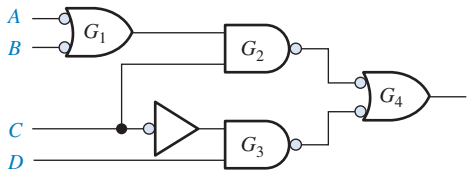
Exemplo de rastreamento de sinal e análise de forma de onda numa parte de uma placa de circuito impresso. O TP indica ponto de teste (*test point*).

Passo 4. Observe a saída da porta G_3 (TP8) em relação às suas entradas. Se ela estiver correta, verifique em seguida a porta G_4. Se a saída não estiver correta, a porta ou suas conexões estão com problema; ou, se a saída estiver em nível BAIXO, a entrada da porta G_4 (TP7) pode estar em curto-circuito.

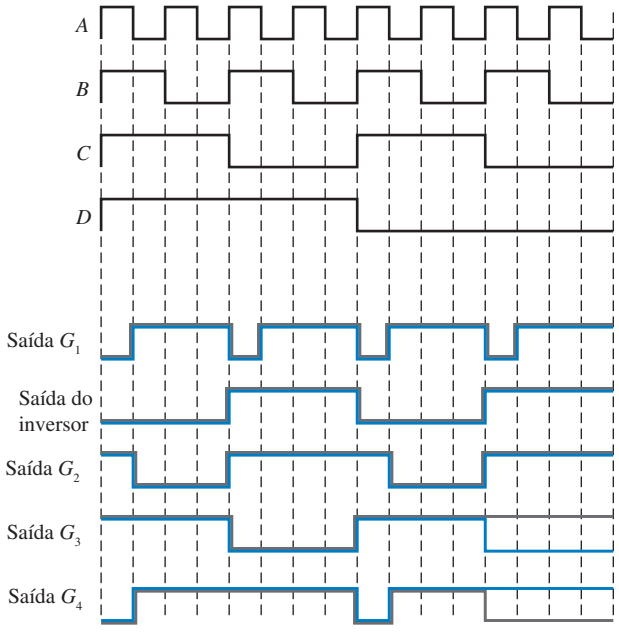
Passo 5. Observe a saída da porta G_4 (TP9) em relação às suas entradas. Se ela estiver correta, o circuito está funcionando bem. Se a saída não estiver correta, a porta ou suas conexões estão com problema.

EXEMPLO 5–15

Determine o defeito no circuito lógico mostrado na Figura 5–47(a) usando a análise de forma de onda. Você observou as formas de onda mostradas em laranja na Figura 5–47(b). As formas de onda em cinza são as corretas e foram fornecidas para comparação.



(a)



(b)

▲ FIGURA 5–47

Solução

1. Determine a forma de onda correta para cada porta. As formas de onda corretas são mostradas em cinza, sobreposta à forma de onda real medida, conforme a Figura 5–47(b).

2. Compare as formas de onda de porta para porta até encontrar uma forma de onda medida que não coincide com a forma de onda correta.

Neste exemplo, os testes estão corretos até o teste da porta G_3. A saída dessa porta não está correta conforme a diferença indicada nas formas de onda. Uma análise das formas de onda indica que se a entrada D da porta G_3 estiver aberta e funcionando como um nível ALTO, obtemos a forma de onda medida (mostrada em cinza). Observe que a saída de G_4 também está incorreta devido às entradas incorretas a partir de G_3.

Substitua o CI que contém a porta G_3, e verifique a operação do circuito novamente.

Problema relacionado Para as entradas do circuito mostrado na Figura 5–47(a), determine a forma de onda de saída (saída de G_4) se o inversor estiver com a saída aberta.

DICA PRÁTICA

Como sabemos, o teste e a análise de defeito de circuitos lógicos freqüentemente requer observação e comparação de duas formas de onda simultaneamente, tal como a entrada e a saída de uma porta, num osciloscópio de dois canais. Para formas de onda digitais, o osciloscópio deve sempre ser ajustado para o acoplamento DC (cc) em cada canal de entrada para evitar "deslocamento" no nível de GND. Temos que determinar onde o nível 0 V está na tela para os dois canais.

Para comparar a temporização das formas de onda, o osciloscópio deve ter o sincronismo (trigger) proveniente apenas de um canal (não use o modo vertical ou trigger composto). O canal selecionado para sincronismo deve sempre ser o que tem a forma de onda de menor freqüência, se possível.

SEÇÃO 5–7 REVISÃO

1. Faça uma lista com os quatro tipos de defeitos mais comuns em portas lógicas.
2. Uma entrada de uma porta NOR tem um curto-circuito externo para V_{CC}. Como essa condição afeta a operação da porta?
3. Determine a saída da porta G_4 no circuito mostrado na Figura 5–47(a), com entradas conforme mostrado na parte (b) para os seguintes defeitos:
   (a) uma entrada de G_1 em curto-circuito para GND
   (b) a entrada do inversor em curto-circuito para GND
   (c) a saída de G_3 aberta

Os problemas de análise de defeito que se encontram no CD-ROM estão disponíveis na Seção "Prática de Análise de Defeito Usando o Multisim" no final do capítulo.

APLICAÇÕES EM SISTEMAS DIGITAIS

Neste tópico de aplicações em sistemas digitais, desenvolvemos um circuito lógico de controle digital para o controle de fluido num tanque de armazenamento. A finalidade do circuito lógico é manter um nível apropriado do fluido controlando as válvulas de entrada e saída. Além disso, o circuito lógico tem que controlar a temperatura do fluido dentro de uma certa faixa e emitir um alarme conforme indicação dos sensores de nível ou temperatura.

Operação Básica do Sistema

As saídas do circuito lógico de controle do sistema controla a entrada de fluido, a saída de fluido e a temperatura do fluido. O circuito lógico de controle opera uma válvula de entrada que permite o fluido entrar no tanque até que o sensor de nível alto seja ativado ao ficar imerso no fluido. Quando o sensor de nível alto está imerso (ativado), o circuito lógico de controle fecha a válvula de entrada. O fluido no tanque que tem que ser mantido dentro de uma faixa de temperatura especificada conforme determinado pelos dois sensores de temperatura. Um sensor de temperatura indica quando o fluido está muito quente e o outro indica quando o fluido está muito frio. O circuito lógico de controle liga um elemento aquecedor se os sensores de temperatura indicarem que o fluido está muito frio. O circuito lógico de controle mantém a válvula de saída aberta enquanto o sensor de nível baixo estiver imerso e o fluido estiver na temperatura adequada. Quando o nível do fluido cai abaixo do nível mínimo, o circuito lógico de controle fecha a válvula de saída.

Requisitos Operacionais

Os níveis máximo e mínimo do fluido são determinados pelas posições dos sensores de nível no tanque. A saída de cada sensor será nível ALTO quando ele estiver imerso no fluido e será nível BAIXO quando ele não estiver imerso. Quando a saída do sensor de nível alto estiver no estado BAIXO, o circuito lógico de controle produz um nível lógico ALTO e abre a válvula de entrada. Quando a saída do sensor de nível alto estiver no estado ALTO, o circuito lógico de controle produz um nível lógico BAIXO e fecha a válvula de entrada.

O fluido tem que estar dentro de uma faixa de temperatura especificada antes que a válvula de saída seja aberta. Um

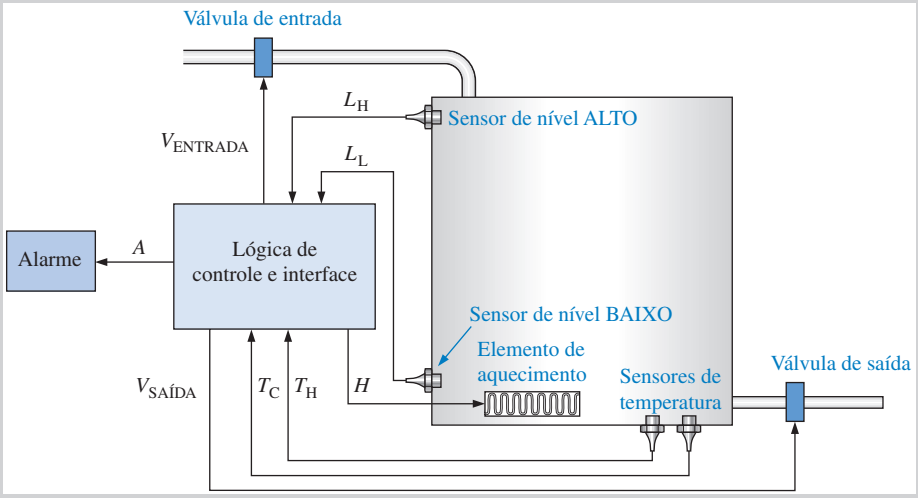


FIGURA 5–48

Tanque de armazenamento do fluido com os sensores de temperatura e nível e o circuito de controle.

TABELA 5–6

Entradas e saídas para o circuito lógico de controle do tanque

ENTRADAS PARA A LÓGICA DE CONTROLE

Variável	Descrição	Nível ativo	Comentários
L_H	Sensor de nível ALTO	ALTO (1)	O sensor está imerso
L_L	Sensor de nível BAIXO	ALTO (1)	O sensor está imerso
T_H	Sensor de temperatura alta	ALTO (1)	Temperatura muito quente
T_C	Sensor de temperatura baixa	ALTO (1)	Temperatura muito fria

SAÍDAS A PARTIR DA LÓGICA DE CONTROLE

Variável	Descrição	Nível ativo	Comentários
$V_{ENTRADA}$	Válvula de entrada	ALTO (1)	Válvula aberta
$V_{SAÍDA}$	Válvula de saída	ALTO (1)	Válvula aberta
H	Elemento de aquecimento	ALTO (1)	Aquecedor ligado
A	Alarme	ALTO (1)	Falha do sensor ou condição de superaquecimento

sensor produz um nível lógico ALTO quando a temperatura estiver muito quente e o outro sensor de temperatura produz um nível lógico ALTO quando a temperatura estiver muito fria. O circuito lógico de controle produz um nível ALTO para ligar o elemento aquecedor quando uma condição de "muito frio" for indicada; caso contrário, o elemento de aquecimento é desligado. Quando uma condição de "muito quente" for indicada, um alarme é ativado.

Quando o sensor de nível baixo produz um nível lógico ALTO (indicando que está imerso) e quando a saída dos dois sensores de temperatura estiverem em nível lógico BAIXO (indicando uma temperatura correta), o circuito lógico de controle abre a válvula de saída. Se a saída do sensor de nível baixo for para o nível lógico BAIXO ou se a saída de qualquer um dos sensores de temperatura for para o nível lógico BAIXO, o circuito lógico de controle fecha a válvula de saída.

Se o controle detecta uma falha em qualquer um dos sensores ou uma condição de "muito quente", um alarme é ativado. Uma falha no sensor de nível é indicada quando o sensor de nível alto é ativado e o sensor de nível baixo não é ativado. Uma falha no sensor de temperatura é indicada quando os dois sensores estiverem ativos ao mesmo tempo. A Figura 5–48 mostra o sistema de controle do tanque.

As entradas e saídas do sistema aparecem reunidas na Tabela 5–6 e a tabela-verdade é mostrada na Tabela 5–7.

Projeto da Lógica de Controle

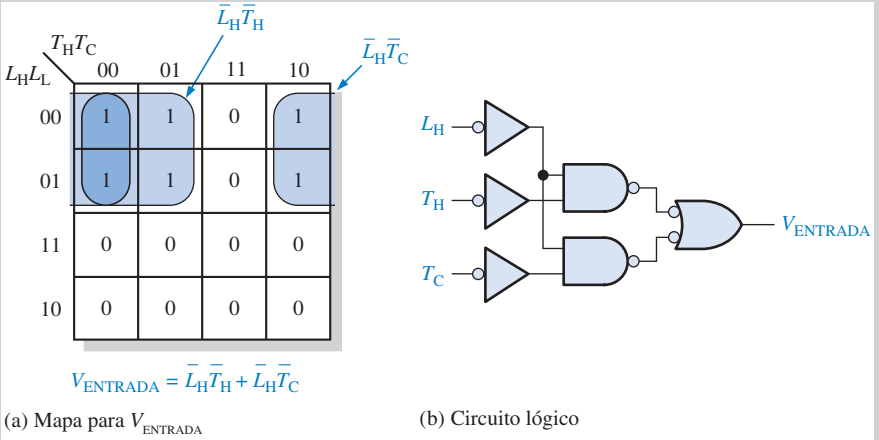
Existem quatro saída separadas: uma para a válvula de entrada, uma para a válvula de saída, uma para o aquecedor e uma para o alarme. Abordamos esse projeto separando-o em quatro circuitos lógicos.

Lógica para Válvula de Entrada Vamos começar projetando o circuito lógico da

TABELA 5–7

Tabela-verdade para o circuito lógico de controle do tanque

ENTRADAS				SAÍDAS				
L_H	L_L	T_H	T_C	*Entrada*	*Saída*	*H*	*A*	COMENTÁRIOS
0	0	0	0	1	0	0	0	Encher/aquecer desligado
0	0	0	1	1	0	1	0	Encher/aquecer ligado
0	0	1	0	1	0	0	1	Encher/aquecer desligado/alarme
0	0	1	1	0	0	0	1	Falha/alarme do sensor de temperatura
0	1	0	0	1	1	0	0	Encher e drenar/aquecer desligado
0	1	0	1	1	0	1	0	Encher/aquecer ligado
0	1	1	0	1	0	0	1	Encher/aquecer desligado/alarme
0	1	1	1	0	0	0	1	Falha/alarme do sensor de temperatura
1	0	0	0	0	0	0	1	Falha/alarme do sensor de nível
1	0	0	1	0	0	0	1	Falha/alarme do sensor de nível
1	0	1	0	0	0	0	1	Falha/alarme do sensor de nível
1	0	1	1	0	0	0	1	Falha múltipla de sensores/alarme
1	1	0	0	0	1	0	0	Drenar/aquecer desligado
1	1	0	1	0	0	1	0	Aquecedor ligado
1	1	1	0	0	0	0	1	Aquecedor desligado/alarme
1	1	1	1	0	0	0	1	Falha/alarme do sensor de temperatura



(a) Mapa para $V_{ENTRADA}$ (b) Circuito lógico

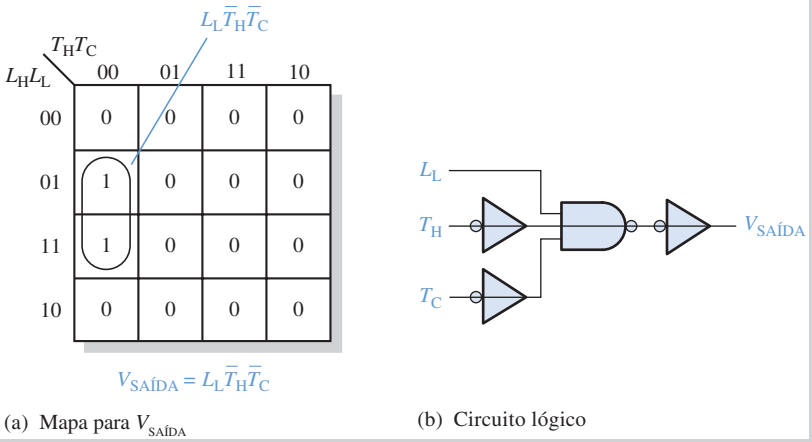
FIGURA 5–49

Simplificação por mapa de Karnaugh e implementação do circuito lógico de controle da válvula de entrada.

válvula de entrada. A saída desse circuito lógico é a variável $V_{ENTRADA}$. O primeiro passo é transferir os dados da tabela-verdade para o mapa de Karnaugh e desenvolver uma expressão de soma-de-produtos.

As variáveis de entrada, L_H, L_L, T_H e T_C são inseridas no mapa e os estados de $V_{ENTRADA}$ são impressos e agrupados como mostrado na Figura 5–49(a). Os 0s no mapa são para as condições de entrada quando a válvula de entrada é fechada e os 1s são as condições de entrada quando a válvula de entrada está aberta. A expressão de soma-de-produtos resultante para o circuito lógico da válvula de entrada é a implementação NAND mostrada na parte (b).

Lógica para Válvula de Saída Em seguida, vamos projetor o circuito lógico para a válvula de saída. A saída desse circuito lógico é a variável $V_{SAÍDA}$. Novamente, o primeiro passo é transferir os dados da tabela-verdade para o mapa de Karnaugh e de-



(a) Mapa para $V_{SAÍDA}$ (b) Circuito lógico

► FIGURA 5–50

Simplificação por mapa de Karnaugh e implementação do circuito lógico da válvula de saída.

senvolver uma expressão de soma-de-produtos. As variáveis de entrada, L_H, L_L, T_H e T_C são variáveis mapeadas e os estados de $V_{SAÍDA}$ são registrados e agrupados como mostra a Figura 5–50(a). Os 0s no mapa são para as condições de entrada quando a válvula de saída está fechada e os 1s são para as condições de entrada quando a válvula está aberta. A expressão de soma-de-produtos resultante para o circuito lógico da válvula de saída resulta numa implementação NAND mostrada na parte (b).

Código VHDL para os Circuitos Lógicos das Válvulas de Entrada e Saída (opcional)

Uma entidade e arquitetura simples descrevem os circuitos lógicos das válvulas de entrada e saída usando a abordagem de fluxo de dados conforme mostra o programa.

```
entity TankControl is
  port (LL, LH, TH, TC in bit; Vinlet,
    Voutlet: out bit);
end entity TankControl;
architecture ValveLogic of TankControl is
begin
  Vinlet <=(not LH and not TH) or
    (not LH and not TC);
  Voutlet <=LL and not TH and not TC;
end architecture ValveLogic;
```

Os circuitos lógicos das válvulas de entrada e saída foram projetados e o código VHDL foi escrito. Agora é a sua vez de completar o restante do projeto do circuito lógico de controle para o controle do aquecedor e alarme, e escrever o programa VHDL para implementar a lógica no dispositivo destino (programável).

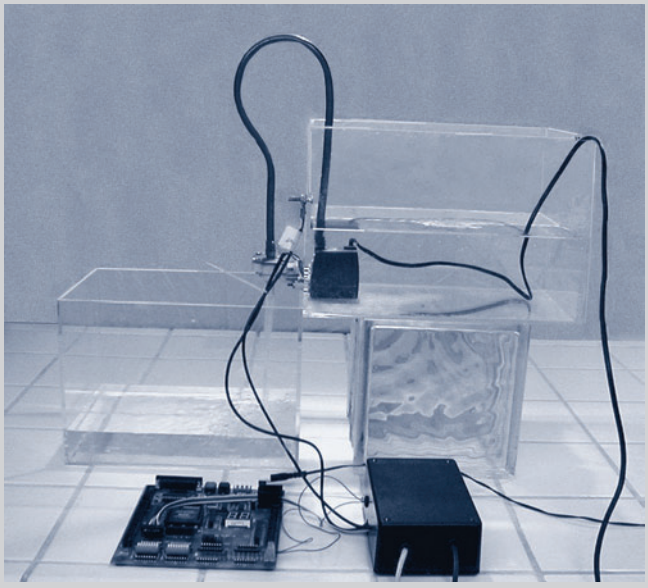


Foto de um protótipo de um tanque de armazenamento no laboratório de eletrônica no Yuba College na Califórnia. O circuito lógico de controle foi programado num PLD usando uma placa de desenvolvimento e conectado a um tanque para controlar a entrada e a saída de fluido no tanque. Foto cortesia de Doug Joksch.

Atribuições do Sistema

- *Atividade 1* Usando a Tabela 5–7 e o método do mapa de Karnaugh, projete o circuito lógico para o controle do elemento aquecedor no tanque. Use portas NAND e inversores para implementar o circuito.
- *Atividade 2* Projete o circuito lógico para a ativação do alarme.
- *Atividade 3* Combine o circuito lógico de cada uma das funções de controle do tanque num diagrama lógico completo.
- *Atividade Opcional* Escreva a entidade e a arquitetura VHDL para o circuito lógico completo modificando o código previamente desenvolvido para os circuitos lógicos das válvulas de controle de entrada e saída.

RESUMO

- A lógica AND-OR produz uma expressão de saída na forma de soma-de-produtos.
- A lógica AND-OR-inversor produz uma soma-de-produtos complementada, que na realidade é uma forma de produto-de-somas.
- O símbolo operacional para a EX-OR é $\oplus$. Uma expressão EX-OR pode ser expressa de duas formas equivalentes:

$$A\overline{B} + \overline{A}B = A \oplus B$$

- Ao fazer a análise de um circuito lógico, desenvolva a expressão Booleana de saída, ou a tabela-verdade ou ainda ambas.
- A implementação de um circuito lógico é o processo no qual começamos com a expressão Booleana de saída ou a tabela-verdade e desenvolvemos um circuito lógico que produz a função de saída.
- Todos os diagramas lógicos de NAND ou NOR devem ser desenhados usando os símbolos duais apropriados de forma que as saídas com os pequenos círculos são conectadas a entradas que possuem também os pequenos círculos e as saídas sem os pequenos círculos conectadas a entradas também sem os pequenos círculos.
- Quando dois indicadores de negação (pequenos círculos) estiverem interconectados, eles se cancelam mutuamente.
- Um componente VHDL é uma função lógica predefinida armazenada para uso ao longo de um programa ou outros programas.
- Um componente instancial é usado na busca por um componente num programa.
- Um sinal VHDL funciona efetivamente como uma interconexão interna na descrição estrutural VHDL.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

AND negativa A operação dual de uma porta NOR quando as entradas são ativas em nível BAIXO.

Componente Uma característica VHDL que pode ser usada para predefinir uma função lógica para ser usada uma ou mais vezes em um ou mais programas.

Nó Um ponto de conexão comum num circuito no qual a saída de uma porta está conectada a uma ou mais entradas de portas.

OR negativa A operação dual de uma porta NAND quando as entradas são ativas em nível BAIXO.

Porta universal Tanto a porta NAND quanto a porta NOR são universais. O termo universal se refere à propriedade de uma porta permitir que qualquer função lógica seja implementada com essa porta ou uma combinação desse tipo de porta.

Rastreamento de sinal Uma técnica de análise de defeito na qual formas de onda são observadas de forma passo a passo começando pela entrada do circuito em direção à saída ou vice-versa. Em cada ponto a forma de onda observada é comparada com o sinal correto para aquele ponto.

Sinal Uma forma de onda; um tipo de objeto VHDL que mantém dados.

AUTOTESTE

As respostas estão no final do capítulo.

1. A expressão de saída para um circuito AND-OR que tem uma porta AND com entradas A, B, C e D e uma porta AND com as entradas E e F é
   (a) $ABCDEF$ **(b)** $A + B + C + D + E + F$
   (c) $(A + B + C + D)(E + F)$ **(d)** $ABCD + EF$

2. Um circuito lógico com uma saída $X = A\overline{B}C + A\overline{C}$ consiste em
(a) duas portas AND e uma porta OR
(b) duas portas AND, uma porta OR e dois inversores
(c) duas portas OR, uma porta AND e dois inversores
(d) duas portas AND, uma porta OR e um inversor

3. Para implementar a expressão $\overline{A}BCD + A\overline{B}CD + AB\overline{C}\,\overline{D}$, gastamos uma porta OR e
(a) uma porta AND
(b) três portas AND
(c) três portas AND e quatro inversores
(d) três portas AND e três inversores

4. A expressão $\overline{A}BCD + ABC\overline{D} + A\overline{B}\,\overline{C}D$
(a) não pode ser simplificada
(b) pode ser simplificada para $\overline{A}BC + A\overline{B}$
(c) pode ser simplificada para $ABC\overline{D} + \overline{A}B\overline{C}$
(d) nenhuma das respostas anteriores está correta.

5. A expressão de saída para um circuito AND-OR-inversor que tem uma porta AND com as entradas *A, B, C* e *D* e uma porta AND com as entradas *E* e *F* é
(a) $ABCD + EF$
(b) $\overline{A} + \overline{B} + \overline{C} + \overline{D} + \overline{E} + \overline{F}$
(c) $\overline{(A + B + C + D)(E + F)}$
(d) $(\overline{A} + \overline{B} + \overline{C} + \overline{D})(\overline{E} + \overline{F})$

6. Uma função EX-OR é expressa como
(a) $\overline{A}\,\overline{B} + AB$ **(b)** $\overline{A}B + A\overline{B}$
(c) $(\overline{A} + B)(A + \overline{B})$ **(d)** $(\overline{A} + \overline{B}) + (A + B)$

7. A operação AND pode ser produzida com
(a) duas portas NAND
(b) três portas NAND
(c) uma porta NOR
(d) três portas NOR

8. A operação OR pode ser produzida com
(a) duas portas NOR
(b) três portas NAND
(c) quatro portas NAND
(d) as alternativas (a) e (b) estão corretas.

9. Ao usar símbolos duais num diagrama lógico
(a) as saídas com pequenos círculos são conectadas a entradas com pequenos círculos.
(b) os símbolos NAND produzem as operações NAND.
(c) os símbolos da OR negativa produzem as operações OR.
(d) todas as alternativas acima são verdadeiras.
(d) nenhuma das alternativas é verdadeira.

10. Todas as expressões Booleanas podem ser implementadas com
(a) apenas com portas NAND
(b) apenas com portas NOR
(c) combinações de portas NAND e NOR
(d) combinações de portas AND, portas OR e inversores

11. Um componente VHDL
(a) pode ser usado uma vez em cada programa.
(b) uma descrição predefinida é uma função lógica.
(c) pode ser usado diversas vezes num programa.
(d) é parte de uma descrição de fluxo de dados.

12. Um componente é solicitado para ser usado num programa através do uso de
(a) um sinal
(b) uma variável
(c) um componente instancial
(d) uma declaração de arquitetura

PROBLEMAS

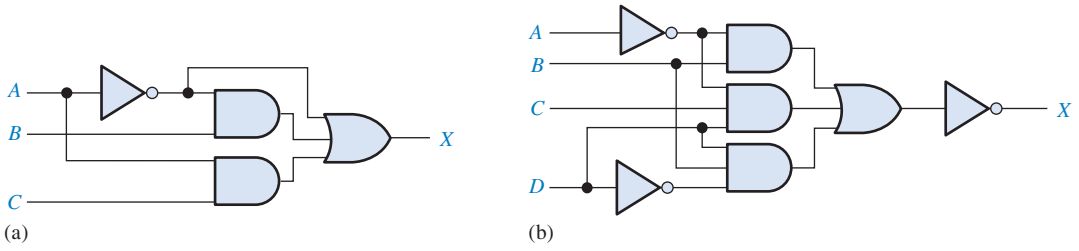
As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 5–1 Circuitos Lógicos Combinacionais Básicos

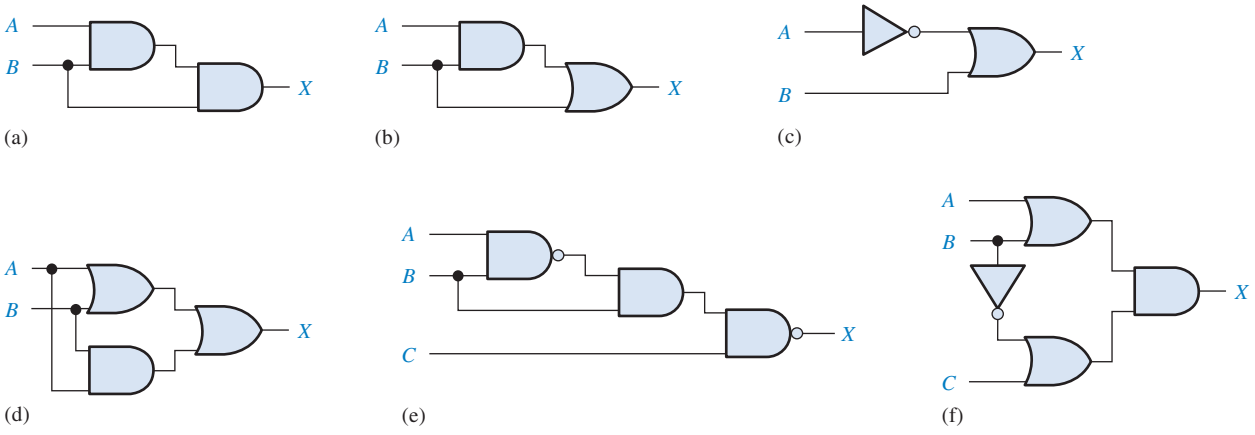
1. Desenhe o diagrama lógico com os símbolos característicos ANSI para um circuito AND-OR-inversor de 4 entradas e 3 estágios.

2. Escreva a expressão de saída para cada circuito mostrado na Figura 5–51.

3. Escreva a expressão de saída para cada circuito mostrado na Figura 5–52.



▲ FIGURA 5–51



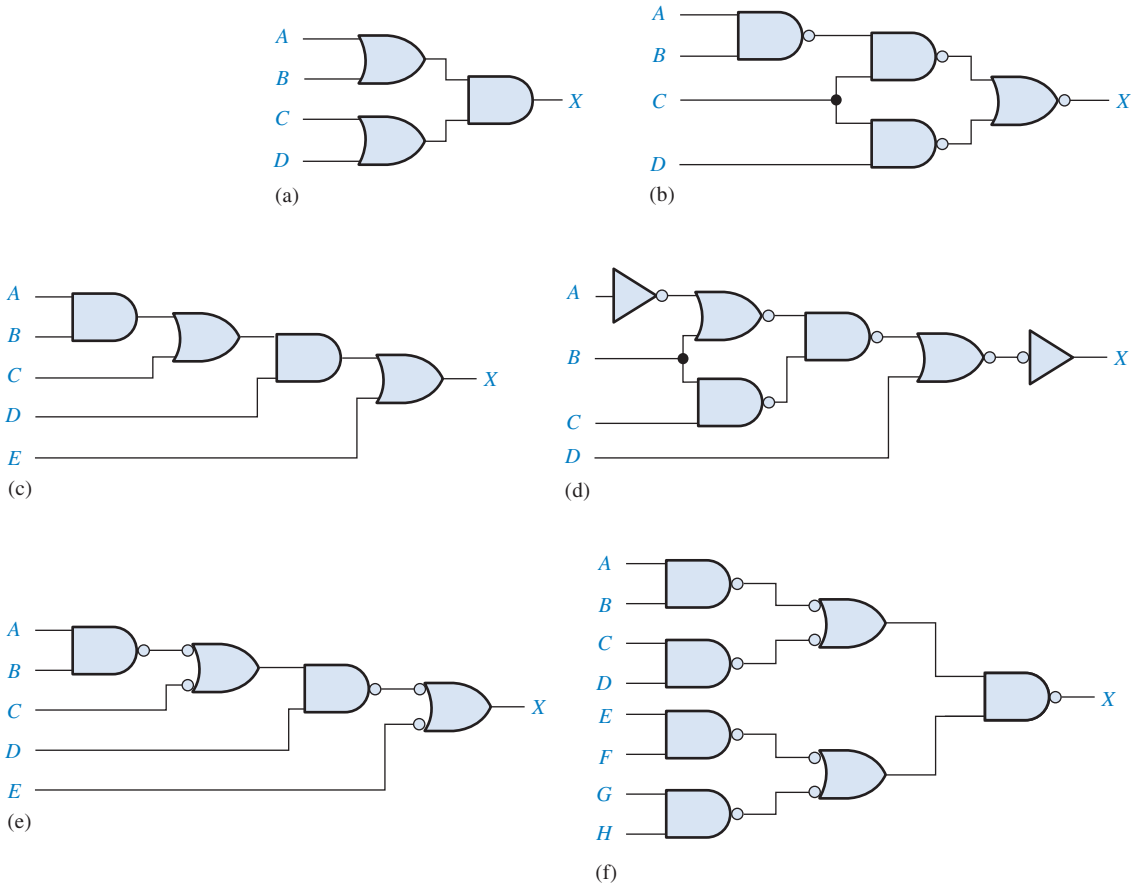
▲ FIGURA 5–52

4. Escreva a expressão de saída para cada circuito mostrado na Figura 5–53 e em seguida passe cada circuito para uma configuração AND-OR equivalente.

5. Desenvolva a tabela-verdade para cada circuito da Figura 5–52.

6. Desenvolva a tabela-verdade para cada circuito da Figura 5–53.

7. Mostre que um circuito EX-NOR produz uma saída de produto-de-somas.



▲ **FIGURA 5–53**

SEÇÃO 5–2 Implementação de Lógica Combinacional

8. Use portas AND, portas OR ou uma combinação de ambas para implementar as seguintes expressões lógicas:

(**a**) $X = AB$
(**b**) $X = A + B$
(**c**) $X = AB + C$
(**d**) $X = ABC + D$
(**e**) $X = A + B + C$
(**f**) $X = ABCD$
(**g**) $X = A(CD + B)$
(**h**) $X = AB(C + DEF) + CE(A + B + F)$

9. Use portas AND, portas OR e inversores conforme necessário para implementar as seguintes expressões:

(**a**) $X = AB + \overline{B}C$
(**b**) $X = A(B + \overline{C})$
(**c**) $X = A\overline{B} + AB$
(**d**) $X = \overline{ABC} + B(EF + \overline{G})$
(**e**) $X = A[BC(A + B + C + D)]$
(**f**) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C)$

10. Use portas NAND, portas NOR ou uma combinação de ambas para implementar as seguintes expressões:

(**a**) $X = \overline{A}B + CD + \overline{(A + B)}(ACD + \overline{BE})$
(**b**) $X = AB\overline{C}\,\overline{D} + D\overline{E}F + \overline{AF}$
(**c**) $X = \overline{A}[B + \overline{C}(D + E)]$

11. Implemente um circuito lógico para a tabela-verdade dada na Tabela 5–8.

TABELA 5–8

ENTRADAS			SAÍDA
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

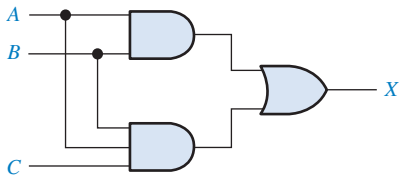
12. Implemente um circuito lógico para a tabela-verdade dada na Tabela 5–9.

TABELA 5–9

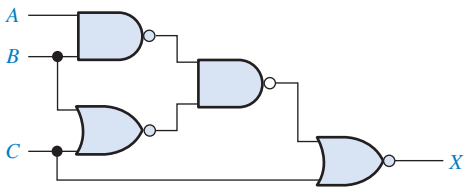
ENTRADAS				SAÍDA
A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

13. Simplifique o circuito mostrado na Figura 5–54 o quanto for possível e verifique que o circuito simplificado é equivalente ao original mostrando que as tabelas-verdade são idênticas.

14. Repita o Problema 13 para o circuito dado na Figura 5–55.



▲ FIGURA 5–54



▲ FIGURA 5–55

15. Minimize as portas necessárias para implementar as funções em cada parte do Problema 9 na forma de soma-de-produtos.

16. Minimize as portas necessárias para implementar as funções em cada parte do Problema 10 na forma de soma-de-produtos.

17. Minimize as portas necessárias para implementar as funções dos circuitos em cada parte da Figura 5–53 na forma de soma-de-produtos.

SEÇÃO 5–3 A Propriedade Universal das Portas NAND e NOR

18. Implemente os circuitos lógicos dados na Figura 5–51 usando apenas portas NAND.

19. Implemente os circuitos lógicos dados na Figura 5–55 usando apenas portas NAND.

20. Repita o Problema 18 usando apenas portas NOR.

21. Repita o Problema 19 usando apenas portas NOR.

SEÇÃO 5–4 Lógica Combinacional Usando Portas NAND e NOR

22. Mostre como as expressões a seguir podem ser implementadas conforme mostradas usando apenas portas NOR.

(a) $X = ABC$ **(b)** $X = ABC$ **(c)** $X = A + B$

(d) $X = A + B + \overline{C}$ **(e)** $X = \overline{AB} + \overline{CD}$ **(f)** $X = (A + B)(C + D)$

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

23. Repita o Problema 22 usando apenas portas NAND.

24. Implemente cada função dada no Problema 8 usando apenas portas NAND.

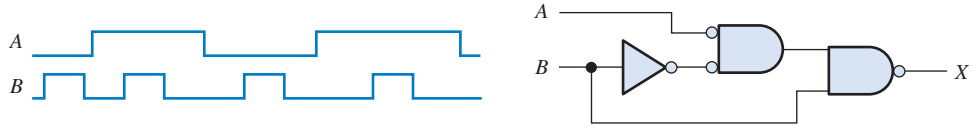
25. Implemente cada função dada no Problema 9 usando apenas portas NAND.

SEÇÃO 5–5 Operação de Circuitos Lógicos com Formas de Onda Digitais nas Entradas

26. Dado um circuito lógico e as formas de onda de entrada vistas na Figura 5–56, desenhe a forma de onda de saída.

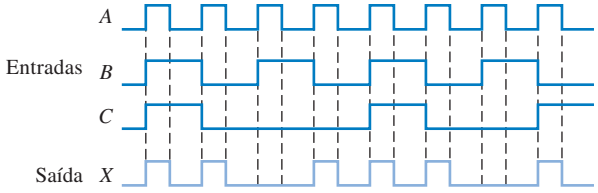
27. Para o circuito lógico visto na Figura 5–57, desenhe a forma de onda de saída relacionando-a com as entradas.

▶ FIGURA 5–56



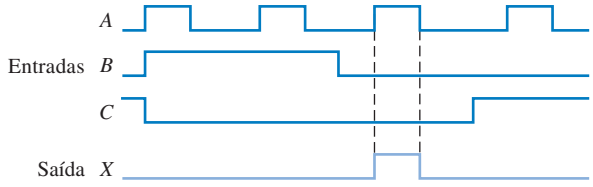
▶ FIGURA 5–57

28. Para as formas de onda de entrada dadas na Figura 5–58, qual é o circuito lógico que gera as formas de onda mostradas?



▶ **FIGURA 5–58**

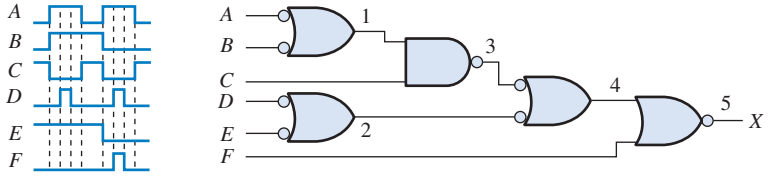
29. Repita o Problema 28 para as formas de onda dadas na Figura 5–59.



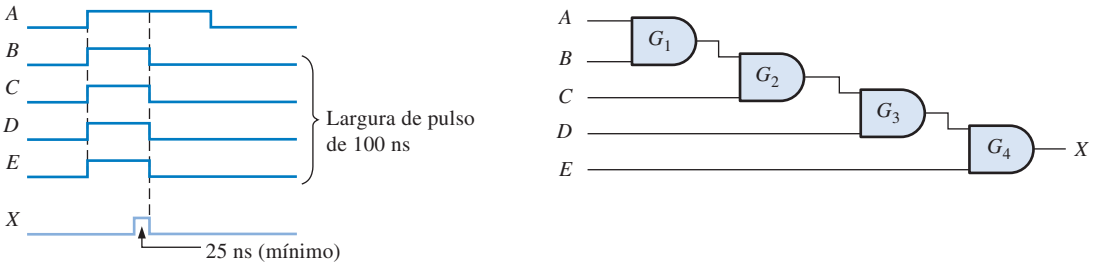
▶ **FIGURA 5–59**

30. Para o circuito visto na Figura 5–60, desenhe as formas de onda nos pontos numerados na mesma base de tempo.

31. Considerando um atraso de propagação através de cada porta de 10 ns, determine se a forma de onda *desejada* na saída *X* vista na Figura 5–61 (um pulso com uma largura mínima de 25 ns posicionado conforme mostrado) seria gerado adequadamente a partir das entradas dadas.



▲ **FIGURA 5–60**



▲ **FIGURA 5–61**

SEÇÃO 5–6 Lógica Combinacional com VHDL (Opcional)

32. Escreva um programa VHDL usando a abordagem de fluxo de dados (expressões Booleanas) para descrever o circuito lógico mostrado na Figura 5–51(b).

33. Escreva programas VHDL usando a abordagem de fluxo de dados (expressões Booleanas) para descrever os circuitos lógicos mostrados na Figura 5–52(e) e (f).

34. Escreva um programa VHDL usando a abordagem estrutural para o circuito lógico mostrado na Figura 5–53(d). Considere que as declarações de componentes para cada tipo de porta já estejam disponíveis.

35. Repita o Problema 34 para o circuito lógico visto na Figura 5–53(f).

36. Descreva a lógica representada pela tabela-verdade mostrada na Tabela 5–8 usando VHDL gerando primeiro a expressão na forma de soma-de-produtos.

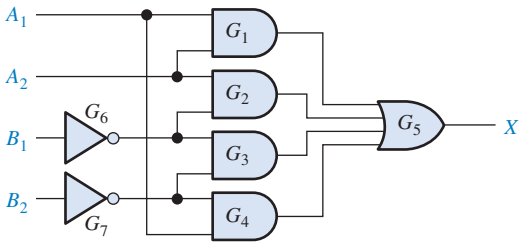
37. Desenvolva um programa VHDL para o circuito lógico mostrado na Figura 5–64, usando tanto a abordagem de fluxo de dados quanto a estrutural. Compare os programas obtidos.

38. Desenvolva um programa VHDL para o circuito lógico mostrado na Figura 5–68, usando tanto a abordagem de fluxo de dados quanto a estrutural. Compare os programas obtidos.

39. Dado o seguinte programa VHDL, gere a tabela-verdade que descreve o circuito lógico.

```
entity CombLogic is
  port (A, B, C, D: in bit; X: out bit);
end entity CombLogic;

architecture Example of CombLogic is
  begin
    X <= not((not A and not B) or (not A and not C) or (not A and not D) or
            (not B and not C) or(not B and not D) or (not D and not C));
end architecture Example;
```

40. Descreva o circuito lógico mostrado na Figura 5–62 com um programa VHDL, usando a abordagem de fluxo de dados.

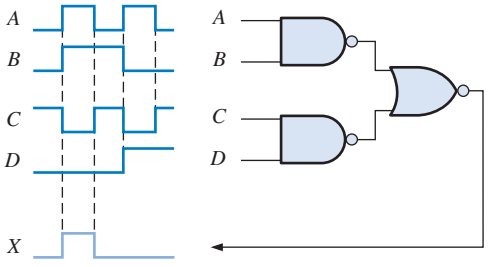


▶ FIGURA 5–62

41. Repita o Problema 40 usando a abordagem estrutural.

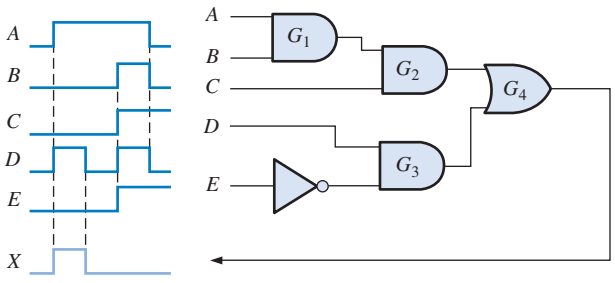
SEÇÃO 5–7 Análise de Defeito

42. Para o circuito lógico e as formas de onda de entrada vistos na Figura 5–63, observa-se a forma de onda de saída indicada. Determine se essa forma de onda de saída está correta.



▶ FIGURA 5–63

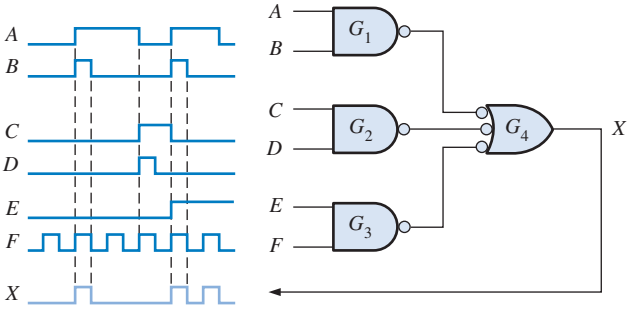
43. A forma de onda de saída mostrada na Figura 5–64 não está correta para as entradas que são aplicadas no circuito. Considerando que uma porta do circuito esteja com defeito, estando a sua saída aparentemente num estado constante de nível ALTO ou de nível BAIXO, determine a porta com defeito e o tipo de defeito (saída com circuito aberto ou em curto-circuito).



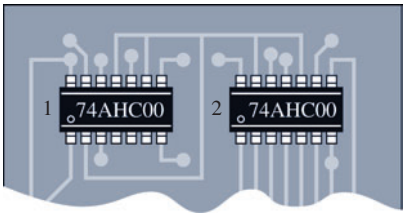
▶ FIGURA 5–64

44. Repita o Problema 43 para o circuito dado na Figura 5–65, sendo as formas de onda de entrada e saída conforme mostradas.

45. Analisando as conexões no circuito da Figura 5–66, determine a porta acionadora e a(s) porta(s) acionada(s). Especifique o dispositivo e os números dos pinos.

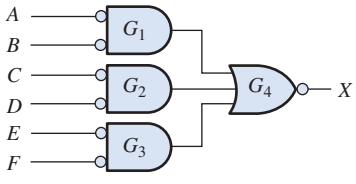


▲ FIGURA 5–65

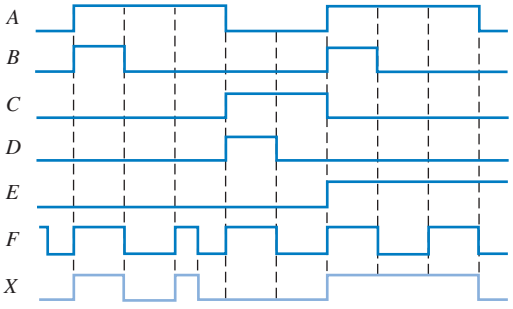


▲ FIGURA 5–66

46. A Figura 5–67(a) é um circuito lógico sob teste. A Figura 5–67(b) mostra as formas de onda observadas com o uso de um analisador lógico. A forma de onda de saída está correta para as entradas que são aplicadas no circuito. Considerando que uma porta do circuito está com defeito, sendo que a sua saída está aparentemente num nível ALTO ou BAIXO constante, determine a porta com defeito e o tipo de defeito.



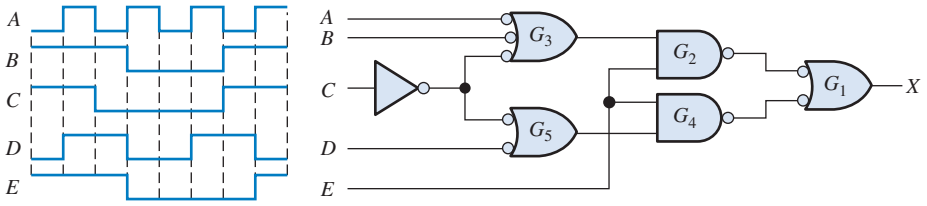
(a)



(b)

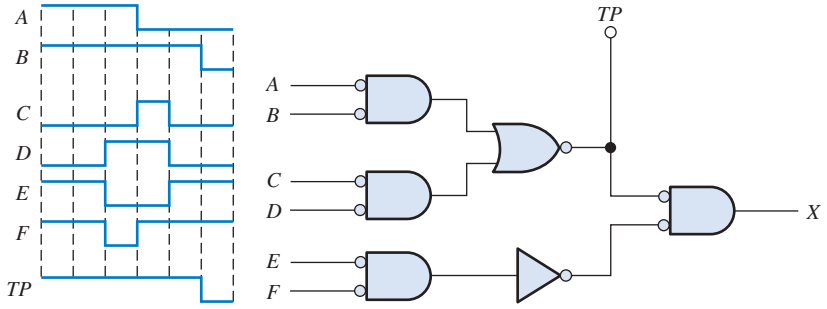
▶ FIGURA 5–67

47. O circuito lógico mostrado na Figura 5–68 tem as formas de onda de entrada mostradas.
(a) Determine a forma de onda de saída correta em relação às entradas.
(b) Determine a forma de onda de saída se a saída da porta G3 estiver aberta.
(c) Determine a forma de onda de saída se a entrada superior da porta G5 estiver em curto-circuito para GND.



▶ **FIGURA 5–68**

48. O circuito lógico mostrado na Figura 5–69 tem apenas um ponto de teste intermediário disponível além da saída, conforme indicado. Para as entradas mostradas, observe a forma de onda indicada no ponto de teste. Essa forma de onda está correta? Em caso negativo, qual o defeito possível que poderia provocar o que vemos?



▶ **FIGURA 5–69**

Aplicações em Sistemas Digitais

49. Implemente o circuito lógico da válvula de entrada mostrado na Figura 5–49(b) usando portas NOR e inversores.

50. Repita o Problema 49 para o circuito lógico da válvula de saída mostrado na Figura 5–50(b).

51. Implemente o circuito lógico do aquecedor e o circuito lógico do alarme usando portas NOR e inversores.

Problemas Especiais de Projeto

52. Projete um circuito lógico para produzir uma saída de nível ALTO apenas se a entrada, representada por um número binário de 4 bits, for maior que doze ou menor que três. Primeiro desenvolva a tabela-verdade e em seguida desenhe o diagrama do circuito lógico.

53. Desenvolva o circuito lógico necessário para atender ao que segue:

Uma lâmpada alimentada por bateria numa sala é operada a partir de duas chaves, uma atrás da porta e a outra na frente da porta. A lâmpada é ligada se a chave da frente estiver ligada e a chave de trás estiver desligada, ou se a chave da frente estiver desligada e a chave de trás estiver ligada. A lâmpada é desligada se as duas chaves estiverem desligadas e se as duas chaves estiverem ligadas. Digamos que a saída em nível ALTO representa a condição ligado e a saída em nível BAIXO representa a condição desligado.

54. Projete um circuito para permitir que um aditivo químico seja introduzido num fluido através de uma outra válvula apenas quando a temperatura não esteja muito fria ou muito quente e o fluido esteja acima do sensor de nível alto.

55. Desenvolva uma lógica NAND para um teclado hexa codificado de forma a converter cada tecla pressionada.

Prática de Análise de Defeito Usando o Multisim

56. Abra o arquivo P05-56 e teste o circuito lógico para determinar se existe algum defeito. Em caso afirmativo, identifique se for possível.

57. Abra o arquivo P05-57 e teste o circuito lógico para determinar se existe algum defeito. Em caso afirmativo, identifique se for possível.

58. Abra o arquivo P05-58 e teste o circuito lógico para determinar se existe algum defeito. Em caso afirmativo, identifique se for possível.

59. Abra o arquivo P05-59 e teste o circuito lógico para determinar se existe algum defeito. Em caso afirmativo, identifique se for possível.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 5–1 Circuitos Lógicos Combinacionais Básicos

1. **(a)** $\overline{AB + CD} = \overline{1 \cdot 0 + 1 \cdot 0} = 1$ **(b)** $\overline{AB + CD} = \overline{1 \cdot 1 + 0 \cdot 1} = 0$
(c) $\overline{AB + CD} = \overline{0 \cdot 1 + 1 \cdot 1} = 0$

2. **(a)** $A\overline{B} + \overline{A}B = 1 \cdot \overline{0} + \overline{1} \cdot 0 = 1$ **(b)** $A\overline{B} + \overline{A}B = 1 \cdot \overline{1} + \overline{1} \cdot 1 = 0$
(c) $A\overline{B} + \overline{A}B = 0 \cdot \overline{1} + \overline{0} \cdot 1 = 1$ **(d)** $A\overline{B} + \overline{A}B = 0 \cdot \overline{0} + \overline{0} \cdot 0 = 0$

3. $X = 1$ quando $ABC = 000, 011, 101, 110$ e 111; e $X = 0$ quando $ABC = 001, 010$ e 100

4. $X = AB + \overline{A}\,\overline{B}$; o circuito consiste de duas portas AND, uma porta OR e dois inversores. Veja o diagrama na Figura 5–6(b).

SEÇÃO 5–2 Implementação de Lógica Combinacional

1. **(a)** $X = ABC + AB + AC$: três portas AND e uma porta OR
(b) $X = AB(C + DE)$; três portas AND e uma porta OR

2. $X = ABC + \overline{A}\,\overline{B}\,\overline{C}$; duas portas AND, uma porta OR e três inversores

3. **(a)** $X = AB(C + 1) + AC = AB + AC$ **(b)** $X = AB(C + DE) = ABC + ABDE$

SEÇÃO 5–3 A Propriedade Universal das Portas NAND e NOR

1. **(a)** $X = \overline{A} + B$; uma porta NAND de 2 entradas sendo as suas entradas A e.
(b) $X = A\overline{B}$; uma porta NAND de 2 entradas sendo as suas entradas A e, seguida de uma NAND usada como inversor.

2. **(a)** $X = \overline{A} + B$; uma porta NOR de 2 entradas sendo as suas entradas e B, seguida por uma NOR usada como inversor.
(b) $X = A\overline{B}$; uma porta NOR de 2 entradas sendo as suas entradas e B.

SEÇÃO 5–4 Lógica Combinacional Usando Portas NAND e NOR

1. $X = \overline{(\overline{A} + \overline{B} + \overline{C})DE}$; uma porta NAND de 3 entradas sendo as suas entradas A, B e C com a saída conectada a uma segunda porta NAND de 3 entradas com as duas outras entradas, D e E.

2. $X = \overline{\overline{A}\,\overline{B}\,\overline{C} + (D + E)}$; uma porta NOR de 3 entradas sendo as suas entradas A, B e C com a saída conectada a uma segunda porta NOR de 3 entradas com as duas outras entradas, D e E.

SEÇÃO 5–5 Operação de Circuitos Lógicos com Formas de Onda Digitais nas Entradas

1. A saída de uma EX-OR é um pulso de 15 μs seguido de um pulso de 25 μs, tendo uma separação entres eles de 10 μs.
2. A saída de uma EX-NOR é nível ALTO quando ambas entradas estiverem em nível ALTO ou quando ambas estiverem em nível BAIXO.

SEÇÃO 5–6 Lógica Combinacional com VHDL (Opcional)

1. Um componente VHDL é um programa predefinido que descreve uma função lógica específica.
2. Um componente instancial é usado para se obter um componente específico na arquitetura de um programa.
3. As interconexões entre componentes são feitas usando sinais VHDL.
4. Os componentes são usados na abordagem estrutural.

SEÇÃO 5–7 Análise de Defeito

1. As falhas comuns em portas são entrada ou saída aberta; entrada ou saída em curto-circuito para GND.
2. Uma entrada em curto-circuito para V_{CC} faz com que a saída fique fixa em nível BAIXO.
3. **(a)** A saída G_4 é nível ALTO até a borda de subida do sétimo pulso, em seguida ela vai para nível BAIXO.
   (b) A saída G_4 é o mesmo que a entrada D.
   (c) A saída G_4 é o inverso da saída G_2 mostrada na Figura 5–47(b).

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

5–1. $X = AB + AC + BC$

5–2. $X = \overline{AB + AC + BC}$

Se $A = 0$ e $B = 0$, $X = \overline{0 \cdot 0 + 0 \cdot 1 + 0 \cdot 1} = \overline{0} = 1$

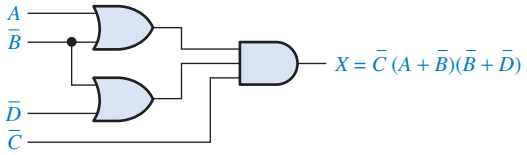
Se $A = 0$ e $C = 0$, $X = \overline{0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0} = \overline{0} = 1$

Se $B = 0$ e $C = 0$, $X = \overline{1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0} = \overline{0} = 1$

5–3. Não pode ser simplificada **5–4.** Não pode ser simplificada

5–5. $X = A + B + C$ é válida

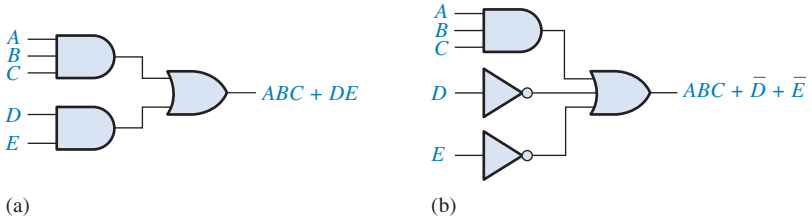
5–6. Veja a Figura 5–70



▶ FIGURA 5–70

5–7. $X = \overline{\overline{(ABC)}\,\overline{(DEF)}} = (\overline{AB})C + (\overline{DE})F = (\overline{A} + \overline{B})C + (\overline{D} + \overline{E})F$

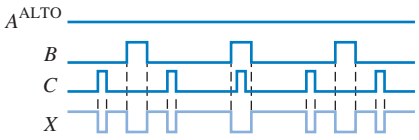
5–8. Veja a Figura 5–71



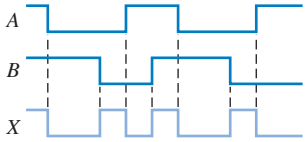
▶ FIGURA 5–71

5–9. $X = \overline{\overline{(A + \overline{B} + C)} + \overline{(D + \overline{E} + F)}} = \overline{(A + \overline{B} + C)}\,\overline{(D + \overline{E} + F)} = (\overline{A}B + C)(\overline{D}E + F)$

5–10. Veja a Figura 5–72



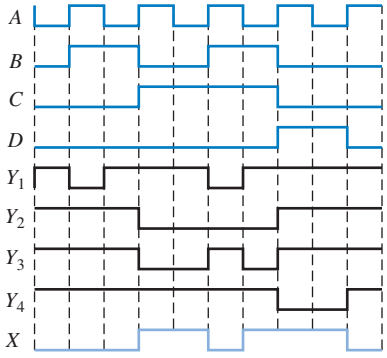
▲ **FIGURA 5–72**



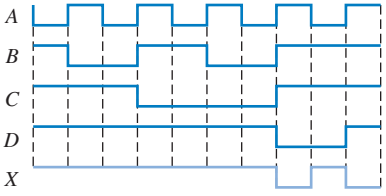
▲ **FIGURA 5–73**

5–11. Veja a Figura 5–73.

5–12. Veja a Figura 5–74.



▲ **FIGURA 5–74**

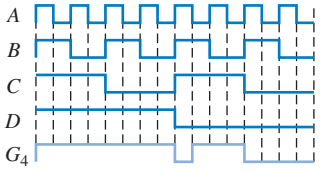


▲ **FIGURA 5–75**

5–13. Veja a Figura 5–75.

5–14. G5: NAND_gate2 **port map** (A=> IN9, B => IN10, X => OUT4);

5–15. Veja a Figura 5–76.



▲ **FIGURA 5–76**

AUTOTESTE

1. (d) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (a) **8.** (d)
9. (d) **10.** (e) **11.** (e) **12.** (c)

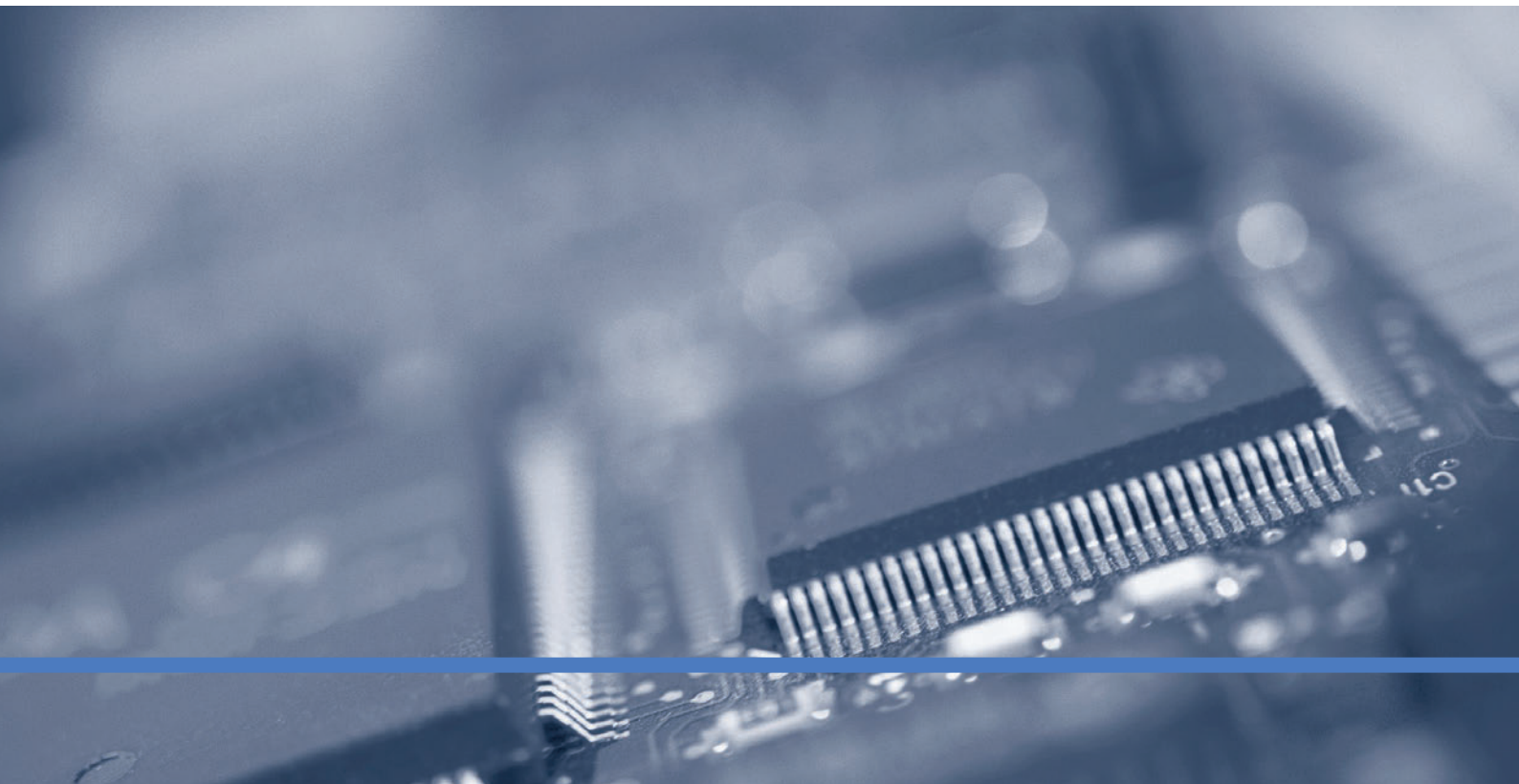
6

FUNÇÕES DE LÓGICA COMBINACIONAL

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Fazer distinção entre meio-somadores e somadores-completos
- Usar somadores-completos para implementar somadores binários em paralelo
- Explicar as diferenças entre somadores em paralelo com carry ondulante e com carry antecipado
- Usar o comparador de magnitude para determinar a relação entre dois números binários e usar comparadores em cascata para conseguir realizar comparações de números com maior número de bits



- Implementar um decodificador binário básico
- Usar decodificadores de BCD para 7 segmentos em sistemas com display
- Usar um codificador de decimal para BCD com prioridade numa aplicação com um teclado simples
- Converter de binário para código Gray e vice-versa usando dispositivos lógicos
- Usar multiplexadores em seleção de dados, displays multiplexados, geração de funções lógicas e sistemas de comunicação simples
- Usar decodificadores como demultiplexadores
- Explicar o significado de paridade
- Usar geradores e verificadores de paridade para detectar erros de bit em sistemas digitais
- Implementar um sistema de comunicação de dados simples
- Identificar glitches, que são problemas comuns em sistemas digitais

TERMOS IMPORTANTES

- Meio-somador
- Somador-completo
- Conexão em cascata
- Carry ondulante
- Carry antecipado
- Decodificador
- Codificador
- Codificador com prioridade
- Multiplexador (MUX)
- Demultiplexador (DEMUX)
- Bit de paridade
- Glitch

INTRODUÇÃO

Neste capítulo, diversos tipos de circuitos lógicos combinacionais são apresentados incluindo somadores, comparadores, decodificadores, codificadores, conversores de código, multiplexadores (seletores de dados), demultiplexadores e geradores/verificadores de paridade. São incluídos também exemplos de circuitos integrados (CIs) de função fixa.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

74XX42	74XX47	74XX85
74XX138	74XX139	74XX147
74XX148	74XX151	74XX154
74XX157	74XX280	74XX283

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico Aplicações em Sistemas Digitais ilustra conceitos abordados neste capítulo e lida com uma parte de um sistema de controle de tráfego de veículos. Esse mesmo tópico nos Capítulos 6, 7 e 8 tem como foco as diversas partes de um sistema de controle de tráfego de veículos. Basicamente, esse sistema controla o trânsito no cruzamento de uma rua movimentada com uma rua secundária de pouco movimento. Esse sistema inclui uma seção lógica combinacional, para a qual os tópicos desse capítulo se aplicam, um circuito de temporização, para o qual se aplica o Capítulo 7 e um circuito lógico seqüencial para o qual se aplica o Capítulo 8.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

6-1 SOMADORES BÁSICOS

Os somadores são importantes em computadores e também em outros tipos de sistemas digitais nos quais dados numéricos são processados. Uma compreensão da operação básica de um somador é fundamental no estudo de sistemas digitais. Esta seção apresenta o meio-somador e o somador-completo.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a função de um meio-somador ■ Desenhar o diagrama lógico de um meio-somador ■ Descrever a função de um somador-completo ■ Desenhar o diagrama lógico de um somador-completo usando meio-somadores ■ Implementar um somador-completo usando lógica AND-OR

O Meio-Somador

Um meio-somador soma dois bits e produz um resultado (soma) e um carry de saída.

Lembre-se das regras básicas para a adição binária abordadas no Capítulo 2.

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

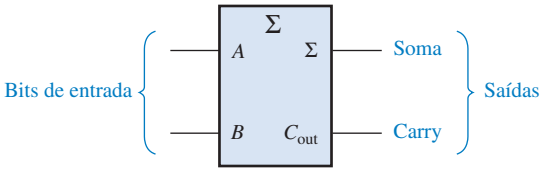
As operações são realizadas por um circuito lógico chamado de **meio-somador**.

O meio-somador aceita dois dígitos binários em suas entradas e produz dois dígitos binários em suas saídas, um bit de soma e um bit de carry.

Um meio-somador é representado pelo símbolo lógico mostrado na Figura 6–1.

▶ **FIGURA 6–1**

Símbolo lógico para um meio-somador. Abra o arquivo F06-01 para verificar a operação.



Lógica do Meio-Somador A partir da operação do meio-somador expressa na Tabela 6–1, podemos deduzir expressões para a soma (resultado) e para o carry de saída como funções das entradas. Observe que o carry de saída (C_{out}) é 1 apenas quando A e B são 1s; portanto, C_{out} pode ser expresso como uma operação AND entre as variáveis de entrada.

Equação 6–1

$$C_{out} = AB$$

▶ **TABELA 6–1**

Tabela-verdade do meio-somador

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

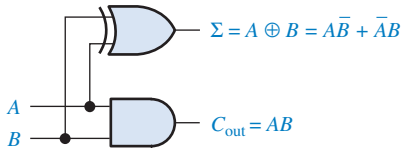
Σ = soma
C_{out} = carry de saída
A e B = variáveis de entrada (operandos)

Agora observe que a saída soma (Σ) é 1 apenas se as variáveis de entrada, A e B, não forem iguais. A soma pode, portanto, ser expressa como a operação EX-OR entre as variáveis de entrada.

$$\Sigma = A \oplus B$$

Equação 6–2

A partir das Equações 6-1 e 6-2 pode-se deduzir a implementação lógica exigida para o meio-somador. O carry de saída é produzido com uma porta AND com A e B nas entradas e a saída da soma é gerada com uma porta EX-OR, conforme mostra a Figura 6–2. Lembre que a EX-OR é implementada com portas AND, uma porta OR e inversores.



◄ FIGURA 6–2
Diagrama lógico de um meio-somador.

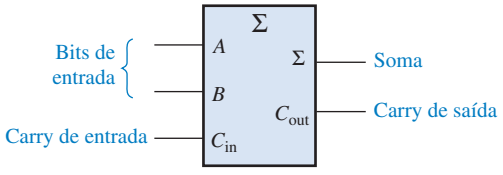
O Somador-Completo

A segunda categoria de somadores é o **somador-completo**.

Um somador-completo tem um carry de entrada, enquanto que um meio-somador não tem.

O somador-completo aceita dois bits de entrada e um carry de entrada, e gera uma saída de soma e um carry de saída.

A diferença básica entre um somador-completo e um meio-somador é que o somador-completo aceita um carry de entrada. A Figura 6–3 mostra o símbolo lógico para um somador-completo e a Tabela 6–2 mostra a operação da tabela-verdade para um somador-completo.



◄ FIGURA 6–3
Símbolo lógico para um somador-completo.
Abra o arquivo F06-03 para verificar a operação.

◄ TABELA 6–2
Tabela-verdade do somador-completo

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{in} = carry de entrada, algumas vezes indicado como *CI*

C_{out} = carry de saída, algumas vezes indicado como *CO*

Σ = soma

A e B = variáveis de entrada (operandos)

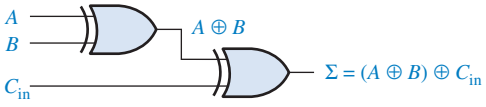
Circuito Lógico do Somador-Completo O somador-completo soma os dois bits de entrada e o carry de entrada. A partir do meio-somador sabemos que a soma dos bits de entrada A e B é a EX-OR

dessas duas variáveis $A \oplus B$. Para o carry de entrada (C_{in}) ser somado aos bits de entrada, deve-se fazer uma EX-OR com $A \oplus B$, resultando na equação para a saída da soma do somador-completo.

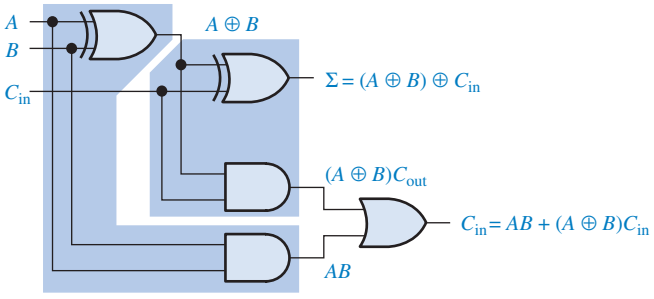
Equação 6–3

$$\Sigma = (A \oplus B) \oplus C_{in}$$

Isso significa que para implementar a função soma do somador-completo, usa-se duas portas EX-OR de 2 entradas. A primeira tem que gerar o termo $A \oplus B$ e a segunda tem como entradas a saída da primeira porta EX-OR e o carry de entrada, conforme ilustra a Figura 6–4(a).



(a) Lógica necessária para construir um somador de três bits.



(b) Circuito lógico completo para um somador-completo (cada meio-somador se encontra numa área sombreada).

▲ FIGURA 6–4

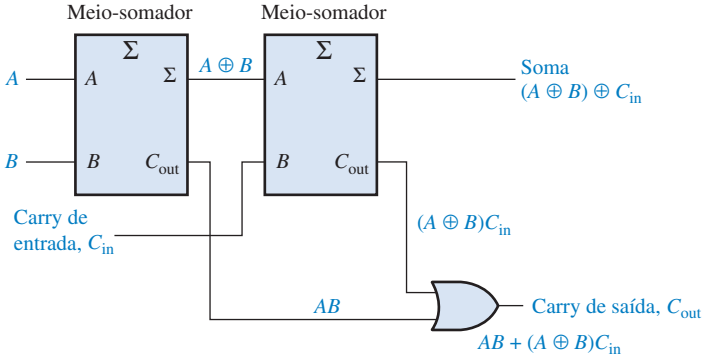
Circuito lógico do somador-completo. Abra o arquivo F06-04 para verificar a operação.

O carry de saída é 1 quando as duas entradas da primeira porta EX-OR são 1s ou quando as duas entradas da segunda porta EX-OR são 1s. Podemos verificar este fato analisando a Tabela 6–2. O carry de saída do somador-completo é portanto produzido pela operação AND de A com B e pela operação AND de com C_{in}. Esses dois termos passam por uma operação OR conforme expresso na Equação 6–4. Essa função é implementada e combinada com a lógica da soma para formar o circuito do somador-completo, conforme mostra a Figura 6–4(b).

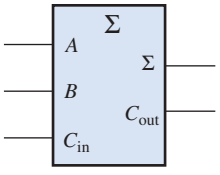
Equação 6–4

$$C_{out} = AB + (A \oplus B)C_{in}$$

Observe na Figura 6–4(b) que existem dois meio-somadores conectados como mostra o diagrama em bloco visto na Figura 6–5(a), com os seus carries de saída passando por uma função OR. O símbolo lógico mostrado na Figura 6–5(b) normalmente é usado para representar um somador-completo.



(a) Associação de dois meio-somadores para construir um somador-completo.



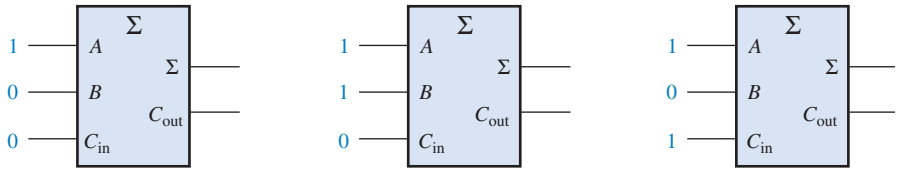
(b) Símbolo lógico do somador-completo.

▲ FIGURA 6–5

Somador-completo implementado com meio-somadores.

EXEMPLO 6–1

Para cada um dos três somadores-completos na Figura 6–6, determine as saídas para as entradas mostradas.



▲ FIGURA 6–6

Solução

(a) Os bits de entrada são $A = 1$, $B = 0$ e $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ sem carry}$$

Portanto, $\Sigma = \mathbf{1}$ e $C_{out} = \mathbf{0}$.

(b) Os bits de entrada são $A = 1$, $B = 1$ e $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ com carry de } 1$$

Portanto, $\Sigma = \mathbf{0}$ e $C_{out} = \mathbf{1}$.

(c) Os bits de entrada são $A = 1$, $B = 0$ e $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ com carry de } 1$$

Portanto, $\Sigma = \mathbf{0}$ e $C_{out} = \mathbf{1}$.

Problema relacionado* Quais são as saídas de um somador-completo para $A = 1$, $B = 1$ e $C_{in} = 1$?

* As respostas estão no final do capítulo.

SEÇÃO 6–1 REVISÃO

As respostas estão no final do capítulo.

1. Determine a soma (Σ) e o carry de saída (C_{out}) de um meio-somador para cada um dos conjuntos de bits de entrada a seguir:
   (a) 01 **(b)** 00 **(c)** 10 **(d)** 11
2. Um somador-completo tem $C_{in} = 1$. Quais são a soma (Σ) e o carry de saída (C_{out}) quando $A = 1$ e $B = 1$?

6-2 SOMADORES BINÁRIOS PARALELOS

Dois ou mais somadores-completos podem ser conectados para construir somadores binários paralelos. Nesta seção você aprenderá a operação básica desse tipo de somador e a associação de suas funções de entrada e saída.

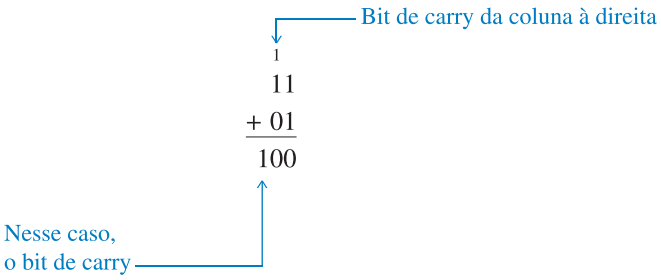
Ao final do estudo desta seção você deverá ser capaz de:

■ Usar somadores-completos para implementar um somador binário paralelo ■ Explicar o processo de adição num somador binário paralelo ■ Usar a tabela-verdade para um somador paralelo de 4 bits ■ Usar dois CIs 74LS283 para implementar a adição de dois números de 4 bits ■ Expandir o somador de 4 bits para realizar adição com números de 8 ou 16 bits

NOTA: COMPUTAÇÃO

A adição é realizada por computadores operando com dois números de cada vez, denominados *operandos*. O *operando fonte* é um número que é somado a um número existente chamado de *operando de destinação*, o qual é armazenado num registro da ALU, como o acumulador. A soma dos dois números é então armazenada de volta no acumulador. A adição é realizada sobre números inteiros ou de ponto flutuante usando instruções do tipo ADD ou FADD, respectivamente.

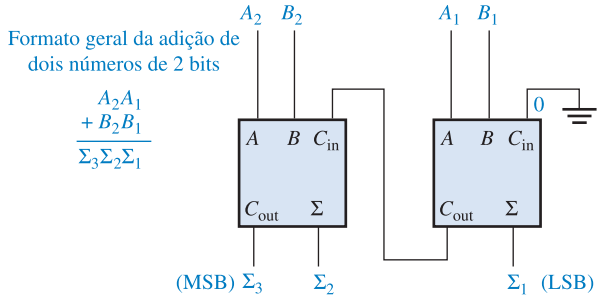
Conforme estudamos na Seção 6–1, um único somador-completo é capaz de somar dois números de 1 bit e um carry de entrada. Para somar números binários com mais de 1 bit, temos que usar somadores-completos adicionais. Quando um número binário é somado a outro, cada coluna gera um bit de soma e um bit de carry (que pode ser 1 ou 0) para a próxima coluna à esquerda, conforme ilustrado a seguir com números de dois bits.



Para somar dois números binários, é necessário um somador-completo para cada bit do número. Assim, para números de dois bits, são necessários dois somadores; para números de 4 bits, são usados quatro somadores; e assim por diante. A saída de carry de cada somador é conectada à entrada de carry do próximo somador de maior ordem, conforme mostra a Figura 6–7 para um somador de dois bits. Observe que um meio-somador pode ser usado na posição menos significativa ou um somador-completo com a entrada de carry colocada em 0 (GND) porque não existe entrada de carry na posição do bit menos significativo.

FIGURA 6–7

Diagrama em bloco de um somador paralelo de 2 bits usando dois somadores-completos. Abra o arquivo F06-07 para verificar a operação.



Na Figura 6–7 o bit menos significativo (LSB) dos dois números são representados por A_1 e B_1. Os próximos bits de ordem maior são representados por A_2 e B_2. E os três bits de soma são Σ_1, Σ_2 e Σ_3. Observe que o carry de saída do somador-completo mais à esquerda se torna o bit mais significativo (MSB) do resultado (soma), Σ_3.

EXEMPLO 6–2

Determine a soma gerada pelo somador paralelo de 3 bits visto na Figura 6–8 e mostre os carries intermediários quando os números binários 1010 e 011 são somados.

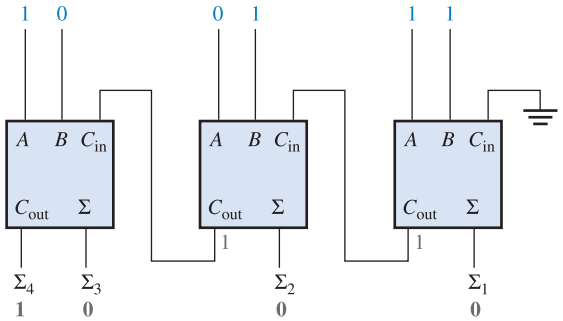


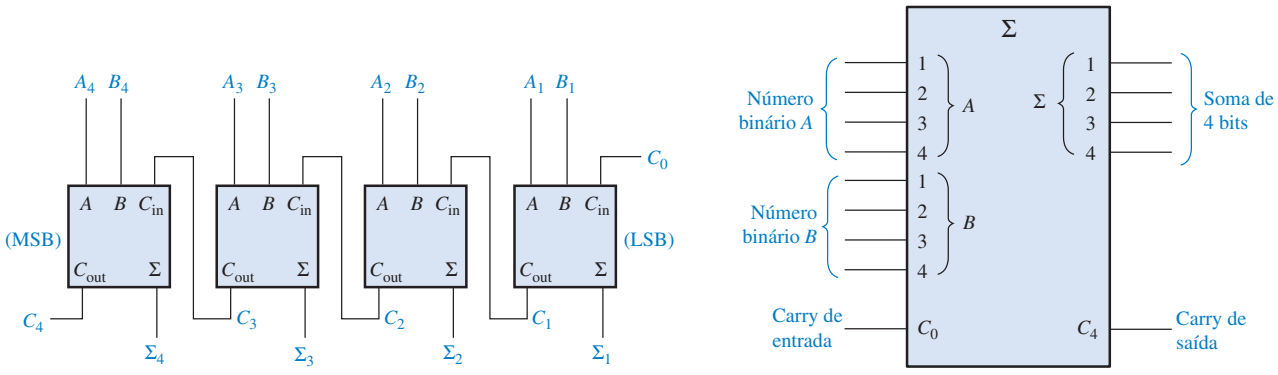
FIGURA 6–8

Solução Os LSBs dos dois números são somados no somador-completo mais à direita. Os bits do resultado e os carries intermediários são indicados em cinza na Figura 6–8.

Problema relacionado Qual é o valor das saídas de soma quando 111 e 101 são somados por um somador paralelo de 3 bits?

Somadores Paralelos de Quatro Bits

Um grupo de quatro bits é denominado de **nibble**. Um somador paralelo de 4 bits básico é implementado com quatro estágios de somadores-completos como mostra a Figura 6–9. Novamente, os bits LSB (A_1 e B_1) em cada número são somados pelo somador-completo mais à direita; os bits de ordem mais alta são inseridos sucessivamente nos somadores de ordem mais alta, com os MSBs (A_4 e B_4) em cada número inseridos no somador-completo mais à esquerda. A saída de carry de cada somador é conectada à entrada de carry para o próximo somador de ordem mais alta conforme indicado. Esses são denominados de *carries internos*.



(a) Diagrama em bloco

(b) Símbolo lógico

▲ **FIGURA 6–9**

Um somador paralelo de 4 bits.

De acordo com as folhas de dados dos fabricantes, a entrada denominada de C_0 é o carry de entrada do bit menos significativo do somador; no caso do quatro bits, C_4 é o carry de saída do bit mais significativo do somador; e Σ_1 (LSB) até Σ_4 (MSB) são as saídas do resultado (soma). O símbolo lógico é mostrado na Figura 6–9(b).

Em termos do método usado para operar com carries em somadores paralelos, existem dois tipos: o somador com *carry ondulante* (*ripple carry*) e o somador com *carry antecipado* (*look-ahead carry*). Esses tipos são discutidos na Seção 6–3.

Tabela-Verdade para um Somador Paralelo de 4 bits

A Tabela 6–3 é a tabela-verdade para um somador de 4 bits. Em algumas folhas de dados, as tabelas-verdade podem ser denominadas *tabelas de funções* ou *tabelas-verdade funcionais*. O subscri-

◀ **TABELA 6–3**

Tabela-verdade para cada estágio de um somador paralelo de 4 bits

C_{n-1}	A_n	B_n	Σ_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

to *n* representa os bits do somador e pode ser 1, 2, 3 ou 4 para o somador de 4 bits. C_{n-1} é o carry do somador anterior. Os carries C_1, C_2 e C_3 são gerados internamente. C_0 é a entrada de carry externo e C_4 é uma saída de carry. O Exemplo 6–3 ilustra como usar a Tabela 6–3.

EXEMPLO 6–3

Use a tabela-verdade do somador paralelo de 4 bits (Tabela 6–3) para determinar a soma e o carry de saída para a adição dos seguintes números de 4 bits se o carry de entrada (C_{n-1}) for 0:

$$A_4A_3A_2A_1 = 1100 \quad \text{e} \quad B_4B_3B_2B_1 = 1100$$

Solução

Para $n = 1$: $A_1 = 0$, $B_1 = 0$ e $C_{n-1} = 0$. A partir da 1ª linha da tabela,

$$\Sigma_1 = \mathbf{0} \quad \text{e} \quad C_1 = 0$$

Para $n = 2$: $A_2 = 0$, $B_2 = 0$ e $C_{n-1} = 0$. A partir da 1ª linha da tabela,

$$\Sigma_2 = \mathbf{0} \quad \text{e} \quad C_2 = 0$$

Para $n = 3$: $A_3 = 1$, $B_3 = 1$ e $C_{n-1} = 0$. A partir da 4ª linha da tabela,

$$\Sigma_3 = \mathbf{0} \quad \text{e} \quad C_3 = 1$$

Para $n = 4$: $A_4 = 1$, $B_4 = 1$ e $C_{n-1} = 1$. A partir da última linha da tabela,

$$\Sigma_4 = \mathbf{1} \quad \text{e} \quad C_4 = \mathbf{1}$$

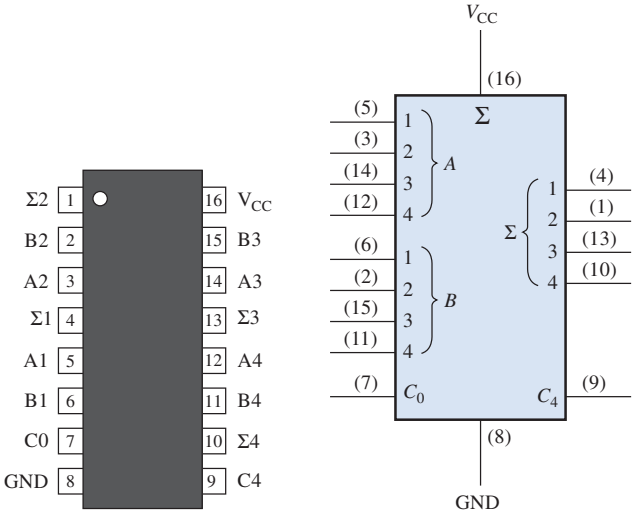
C_4 se torna o carry de saída: a soma de 1100 e 1100 é 11000.

Problema relacionado

Use a tabela-verdade (Tabela 6–3) para determinar o resultado da adição dos números binários 1011 e 1010.

UM SOMADOR PARALELO DE 4 BITS (74LS283)

Um exemplo de um somador paralelo de 4 bits que é comercializado é o CI 74LS283. Nesse CI, V_{CC} é o pino 16 e GND é o pino 8, que é uma configuração padrão. O digrama de pinos e o símbolo lógico para esse dispositivo são mostrados na Figura 6–10 (com os números dos pinos em parênteses no símbolo lógico). Esse dispositivo pode ser comercializado em outras famílias TTL e CMOS. Verifique o site da Texas Instruments em www.ti.com ou o CD-ROM da Texas Instruments que acompanha esse livro.



(a) Diagrama de pinos do 74LS283 (b) Símbolo lógico do 74LS283

► **FIGURA 6–10**
Somador paralelo de 4 bits.

Características Obtidas de Folhas de Dados de CIs Lembre-se que as portas lógicas têm um tempo de atraso de propagação (*propagation delay time*), t_P, de uma entrada para a saída. Para um CI lógico, pode existir diversas especificações diferentes para t_P. O somador paralelo de 4 bits tem as quatro especificações de t_P mostradas na Figura 6–11, que é parte da folha de dados de um 74LS283.

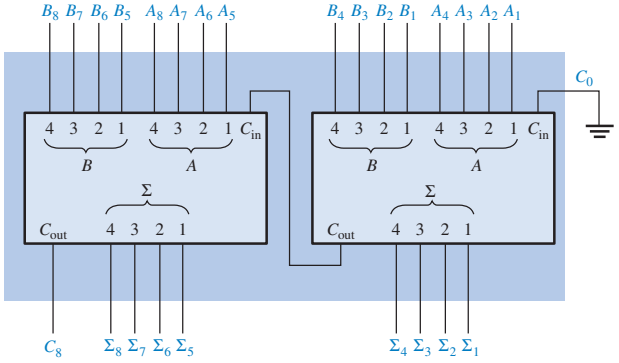
Símbolo	Parâmetro	Limites			Unidade
		Mín	Típ	Mx	
t_{PLH} t_{PHL}	Atraso de propagação da entrada C_0 para qualquer saída Σ		16 15	24 24	ns
t_{PLH} t_{PHL}	Atraso de propagação de qualquer entrada A ou B para as saídas Σ		15 15	24 24	ns
t_{PLH} t_{PHL}	Atraso de propagação da entrada C_0 para a saída C_4		11 11	17 22	ns
t_{PLH} t_{PHL}	Atraso de propagação de qualquer entrada A ou B para a saída C_4		11 12	17 17	ns

◀ **FIGURA 6–11**
Características de atraso de propagação para o 74LS283.

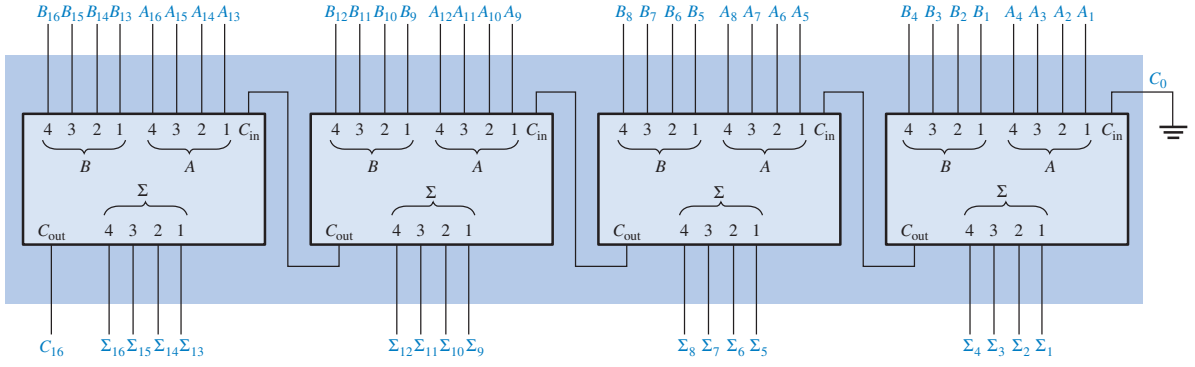
Expansão de um Somador

Somadores podem ser expandidos, conectados em cascata, para operarem mais bits.

Um somador paralelo de 4 bits pode ser expandido para efetuar uma adição de dois números de 8 bits usando dois somadores de 4 bits. A entrada de carry do somador menos significativo (C_0) é conectada em GND porque não existe carry de entrada na posição do bit menos significativo e o carry de saída do somador menos significativo é conectada na entrada de carry do somador mais significativo, conforme mostra a Figura 6–12(a). Esse processo é conhecido como **conexão em**



(a) Associação em cascata de dois somadores de 4 bits para construir um somador de 8 bits.



(b) Associação em cascata de quarto somadores de 4 bits para construir um somador de 16 bits.

▲ **FIGURA 6–12**
Exemplo de expansão de somador.

cascata. Observe que, nesse caso, o carry de saída é indicado por C_8 porque ele é gerado a partir da posição do oitavo bit. O somador menos significativo é o que soma os quatro bits menos significativos, de ordem inferior, e o somador mais significativo é o que soma os quatro bits mais significativos, de ordem superior, do número de 8 bits.

De forma similar, somadores de 4 bits podem ser associados em cascata para operar números de 16 bits conforme mostra a Figura 6–12 (b). Observe que o carry de saída é indicado por C_{16} porque ele é gerado a partir do décimo sexto bit.

EXEMPLO 6–4

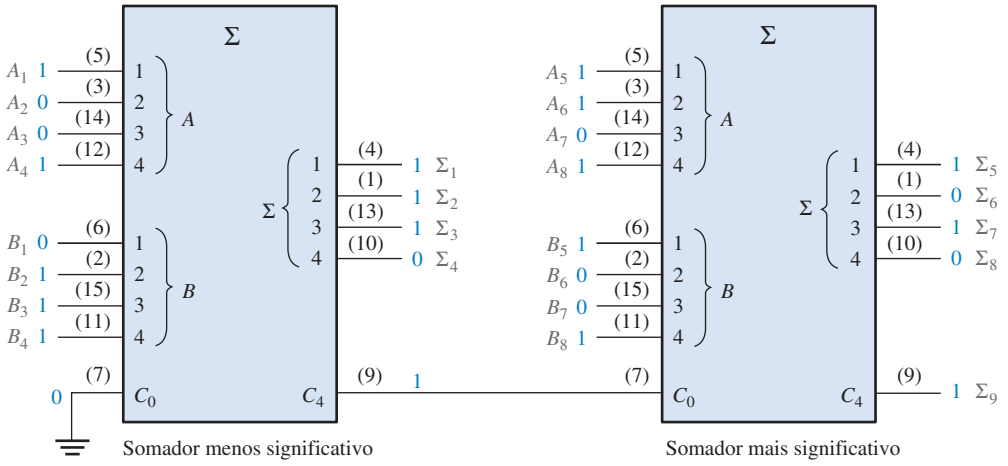
Mostre como dois somadores 74LS283 podem ser conectados para formar um somador paralelo de 8 bits. Mostre os bits de saída para os seguintes números de entrada de 8 bits.

$$A_8A_7A_6A_5A_4A_3A_2A_1 = 10111001 \quad \text{e} \quad B_8B_7B_6B_5B_4B_3B_2B_1 = 10011110$$

Solução Dois somadores paralelos de 4 bits (74LS283) são usados para implementar um somador de 8 bits. A única conexão entre os dois CIs é a saída de carry (pino 9) do somador menos significativo com a entrada de carry (pino 7) do somador mais significativo, como mostra a Figura 6–13. O pino 7 do somador menos significativo é conectado em GND (sem carry de entrada).

A soma dos dois números de 8 bits é

$$\Sigma_9\Sigma_8\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1 = 101010111$$



▲ **FIGURA 6–13**

Dois somadores 74LS283 conectados como um somador paralelo de 8 bits (os números dos pinos estão entre parênteses).

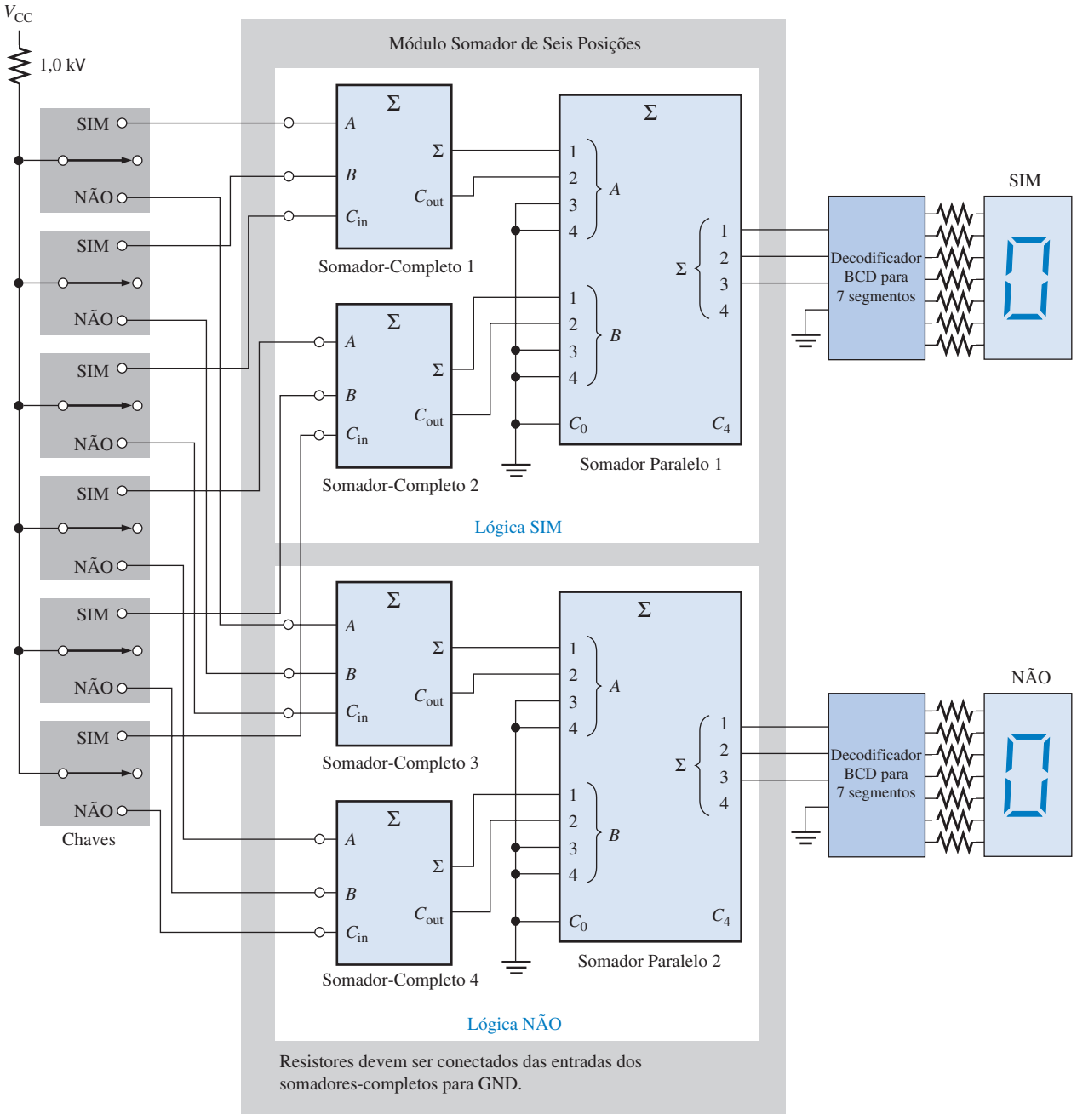
Problema relacionado Use somadores 74LS283 para implementar um somador paralelo de 12 bits.

Uma Aplicação

Um exemplo de aplicação de somador-completo e somador paralelo é um simples sistema de votação que pode ser usado para fornecer simultaneamente o número de votos "sim" e o número de votos "não". Esse tipo de sistema pode ser usado quando um grupo de pessoas é reunido e é necessário determinar imediatamente opiniões (a favor ou contra), tomar decisões ou votar certa questão ou outras matérias.

Em sua forma mais simples, o sistema inclui uma chave para seleção do "sim" ou do "não" em cada um do grupo e um display digital para o número de votos "sim" e um outro para o número de votos "não". O sistema básico é mostrado na Figura 6–14 para uma configuração de 6 posições, porém essa configuração pode ser expandida para qualquer número de posições com módulos de 6 posições adicionais e somadores paralelos e circuitos de displays adicionais.

Na Figura 6–14 cada somador-completo pode produzir a soma de até três votos. A soma e o carry de saída de cada somador-completo passam então para os dois bits menos significativos de um somador paralelo binário. As duas entradas mais significativas do somador paralelo são conectadas em GND (0) porque não existe nenhum caso em que a entrada binária exceda a 0011 (decimal 3). Para esse sistema básico de 6 posições, as saídas do somador paralelo vão para um deco-



▲ **FIGURA 6–14**

Um sistema de votação usando somadores-completos e somadores binários paralelos.

dificador de BCD para 7 segmentos que aciona um display de 7 segmentos. Conforme mencionado, circuitos adicionais têm que ser incluídos quando o sistema é expandido.

Os resistores das entradas de cada somador-completo para GND garantem que cada entrada será nível BAIXO quando a chave estiver na posição neutra (a lógica CMOS é usada). Quando uma chave é comutada para a posição "sim" ou posição "não", um nível ALTO (V_{CC}) é aplicado na entrada do somador-completo associado.

SEÇÃO 6–2 REVISÃO

1. Dois números de 4 bits (1101 e 1011) são aplicados num somador paralelo de 4 bits. O carry de entrada é 1. Determine a soma (Σ) e o carry de saída.
2. Quantos CIs 74LS283 seriam necessários para somar dois números binários em que cada um representa números decimais até 1000_{10}?

6-3 SOMADORES COM CARRY ONDULANTE *VERSUS* SOMADORES COM CARRY ANTECIPADO

Conforme mencionado na última seção, os somadores paralelos podem ser classificados em duas categorias baseado na forma em que os carries internos de um estágio para o outro são operados. As categorias são carry ondulante e carry antecipado. Externamente, os dois tipos de somadores são iguais em termos de entradas e saídas. A diferença é a velocidade na qual eles podem somar números. O somador com carry antecipado é muito mais rápido que o somador com carry ondulante.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir a diferença entre um somador com carry ondulante e um somador com carry antecipado ■ Descrever a vantagem da adição com carry antecipado ■ Definir *geração de carry* e *propagação de carry* explicando a diferença ■ Desenvolver a lógica com carry antecipado ■ Explicar por que CIs 73LS283 conectados em cascata apresentam propriedades de carry ondulante e carry antecipado

Somador com Carry Ondulante

Um somador com **carry ondulante** é aquele no qual a saída de carry de cada somador-completo (FA – *full-adder*) é conectada à entrada de carry do próximo estágio de maior ordem (um estágio corresponde a um somador-completo). A soma e o carry de saída de qualquer estágio não podem ser gerados antes que o carry de entrada seja estabelecido; isso provoca um atraso no processo de adição, conforme ilustrado na Figura 6–15. O atraso de propagação do carry para cada somador-completo é o tempo entre a colocação do carry de entrada e o momento do carry de saída, considerando que as entradas *A* e *B* já estejam estabelecidas.

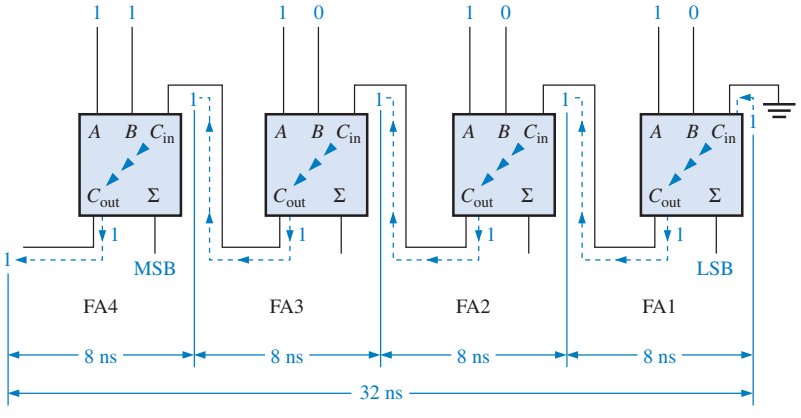


FIGURA 6–15
Um somador paralelo de 4 bits com carry ondulante mostrando atrasos de propagação de carry para o "pior caso".

O somador-completo 1 (FA1) não pode gerar um carry de saída até que um carry de entrada seja aplicado. O somador-completo 2 (FA2) não pode gerar um carry de saída até que o somador-completo 1 gere uma saída de carry. O somador-completo 3 (FA3) não pode gerar um carry de saída até um carry de saída seja gerado pelo FA1 seguido pelo carry de saída do FA2, e assim por diante. Conforme podemos ver na Figura 6–15, o carry de entrada do estágio menos significativo tem que passar (ondular) por todos os somadores antes que a soma final seja gerada. O atraso cumulativo através de todos os estágios somadores é o tempo de adição para o "pior caso". O atraso total pode variar, dependendo do bit de carry gerado por cada somador-completo. Se dois números são somados de forma que não ocorram carries (0) entre os estágios, o tempo de adição é simplesmente o tempo de propagação de um único somador-completo desde a aplicação dos bits de dados nas entradas até o surgimento do resultado na saída de soma.

Somador com Carry Antecipado

A velocidade com a qual uma adição pode ser realizada é limitada pelo tempo necessário para os carries se propagarem (ondulação) através de todos os estágios de um somador paralelo. Um método de aumentar a velocidade do processo de adição que elimina esse atraso do carry ondulante é denominado adição com **carry antecipado**. Esse somador antecipa o carry de saída de cada estágio e, com base nas entradas, produz o carry de saída através da geração ou da propagação de carry.

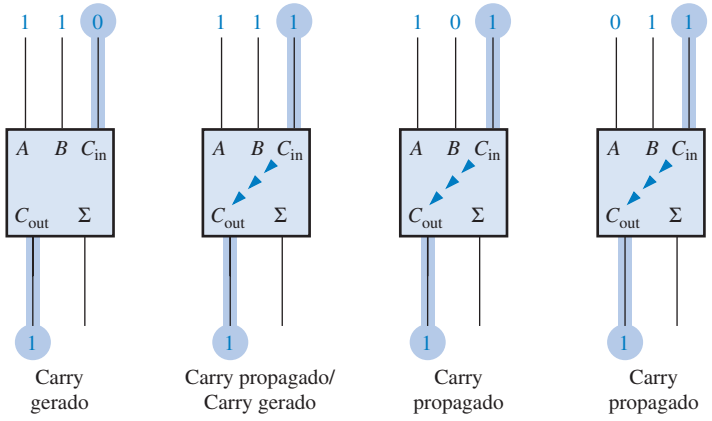
A **geração de carry** ocorre quando um carry de saída é produzido (gerado) internamente pelo somador-completo. Um carry é gerado apenas quando os dois bits de entrada são 1s. O carry gerado, C_g, é expresso como uma função AND dos dois bits de entrada, A e B.

$$C_g = AB$$ **Equação 6–5**

A **propagação de carry** ocorre quando um carry de entrada passa através dos somadores (ondulação) até se tornar um carry de saída. Um carry de entrada pode ser propagado pelo somador-completo quando um ou os dois bits de entrada forem 1s. O carry propagado, C_p, é expresso como a função OR entre os bits de entrada.

$$C_p = A + B$$ **Equação 6–6**

As condições para a geração e a propagação de carry são ilustradas na Figura 6–16. As três pontas de flecha simbolizam a ondulação (propagação).

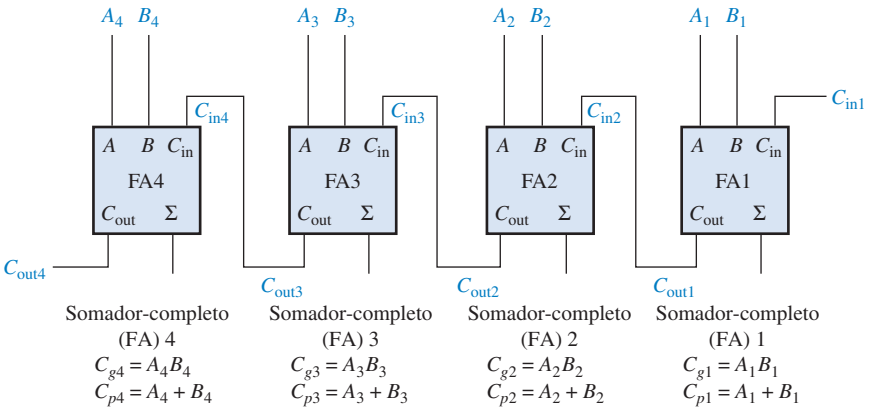


◄ FIGURA 6–16
Ilustração de condições para a geração e propagação de carry.

O carry de saída de um somador-completo pode ser expresso em termos do carry gerado (C_g) e do carry propagado (C_p). O carry de saída (C_{out} – *output carry*) será um 1 se o carry gerado for um 1 OR se o carry propagado for um 1 AND o carry de entrada (C_{in} – *input carry*) for um 1. Em outras palavras, obtemos um carry de saída 1 se ele for gerado pelo somador-completo (A = 1 AND B = 1) ou se o somador propagar o carry de entrada (A = 1 OR B = 1) AND C_{in} = 1. Essa relação é expressa como a seguir:

$$C_{out} = C_g + C_pC_{in}$$ **Equação 6–7**

Agora vejamos como esse conceito pode ser aplicado a um somador paralelo, cujos estágios individuais são mostrados na Figura 6–17 para um exemplo de 4 bits. Para cada somador-completo, o carry de saída é dependente do carry gerado (C_g), do carry propagado (C_p) e de sua entrada de carry (C_{in}). As funções C_g e C_p para cada estágio são disponibilizadas *imediatamente* logo que os bits de entrada A e B e o carry de entrada do somador LSB estiverem presentes porque eles dependem apenas desses bits. O carry de entrada de cada estágio é o carry de saída do estágio anterior.



▶ **FIGURA 6–17**

Geração e propagação de carry em termos dos bits de entrada para um somador de 4 bits.

Com base nessa análise, podemos desenvolver expressões para o carry de saída, C_{out}, de cada estágio somador-completo para o exemplo de 4 bits.

Somador-completo 1:

$$C_{out1} = C_{g1} + C_{p1}C_{in1}$$

Somador-completo 2:

$$\begin{aligned} C_{in2} &= C_{out1} \\ C_{out2} &= C_{g2} + C_{p2}C_{in2} = C_{g2} + C_{p2}C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{in1}) \\ &= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1} \end{aligned}$$

Somador-completo 3:

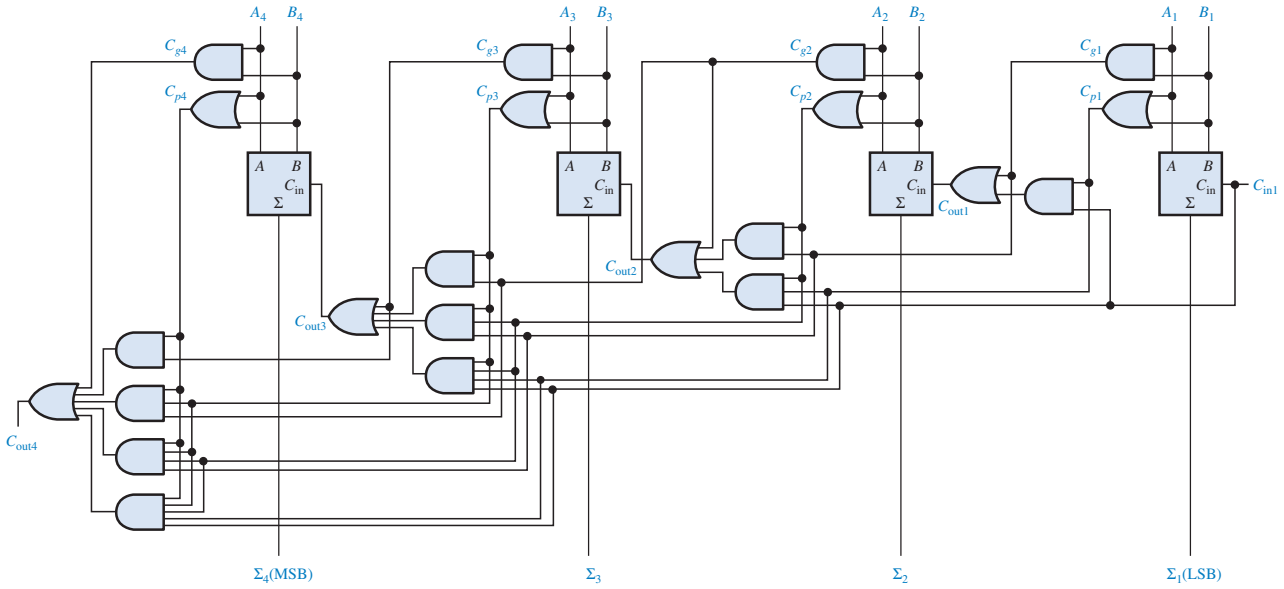
$$\begin{aligned} C_{in3} &= C_{out2} \\ C_{out3} &= C_{g3} + C_{p3}C_{in3} = C_{g3} + C_{p3}C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1}) \\ &= C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1} \end{aligned}$$

Somador-completo 4:

$$\begin{aligned} C_{in4} &= C_{out3} \\ C_{out4} &= C_{g4} + C_{p4}C_{in4} = C_{g4} + C_{p4}C_{out3} \\ &= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1}) \\ &= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{in1} \end{aligned}$$

Observe que em cada uma dessas expressões, o carry de saída para cada estágio somador-completo depende apenas do carry de entrada inicial (C_{in1}), das funções C_g e C_p do estágio, e das funções C_g e C_p dos estágios anteriores. Visto que cada uma das funções C_g e C_p pode ser expressa em termos das entradas A e B dos somadores-completos, todos os carries de saída são disponibilizados imediatamente (exceto pelos atrasos das portas), e não temos que esperar que o carry passe através dos estágios antes que o resultado final seja obtido. Portanto, a técnica de carry antecipado aumenta a velocidade do processo de adição.

As equações para as saídas C_{out} são implementadas com portas lógicas e conectadas aos somadores-completos para criar um somador-completo com carry antecipado, conforme mostra a Figura 6–18.



▲ **FIGURA 6–18**

Diagrama lógico para um somador com carry antecipado de 4 estágios.

Combinação de Somadores com Carry Antecipado e Somadores com Carry Ondulante

O CI 74LS283 (somador de 4 bits) que foi apresentado na Seção 6–2 é um somador com carry antecipado. Quando esses somadores são associados em cascata para expandir sua capacidade de operar números binários com mais de 4 bits, o carry de saída de um somador é conectado à entrada de carry do próximo. Isso gera uma condição de carry ondulante entre os somadores de 4 bits de forma que quando dois ou mais CIs 74LS283 são associados em cascata, o somador resultante é na realidade uma combinação de somador com carry antecipado e carry ondulante. A operação de carry antecipado é interna a cada somador MSI e a característica de carry ondulante se manifesta quando existe um carry de saída de um somador para o seguinte.

SEÇÃO 6–3 REVISÃO

1. Os bits de entrada para um somador-completo são $A = 1$ e $B = 0$. Determine C_g e C_p.
2. Determine o carry de saída de um somador-completo quando $C_{in} = 1$, $C_g = 0$ e $C_p = 1$.

6-4 COMPARADORES

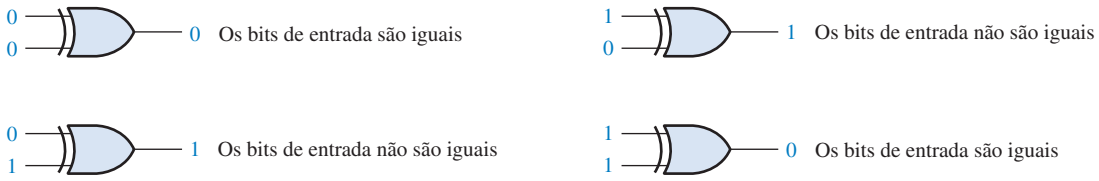
A função básica de um comparador é comparar as magnitudes de dois números binários para determinar a relação comparativa entre eles. Em sua forma mais simples, um circuito comparador determina se dois números são iguais.

Ao final do estudo desta seção você deverá ser capaz de:

■ Usar a porta EX-OR como um comparador básico ■ Analisar a lógica interna de um comparador de magnitude que tem as saídas de igualdade e desigualdade ■ Usar o CI comparador 74HC85 para comparar as magnitudes de dois números de 4 bits ■ Associar em cascata CIs 74HC85 para expandir um comparador para oito bits ou mais

Igualdade

Conforme estudamos no Capítulo 3, a porta EX-OR pode ser usada como um comparador básico porque sua saída é nível 1 se os dois bits de entrada forem diferentes e é 0 se os bits de entrada forem iguais. A Figura 6–19 mostra a porta EX-OR como um comparador de 2 bits.



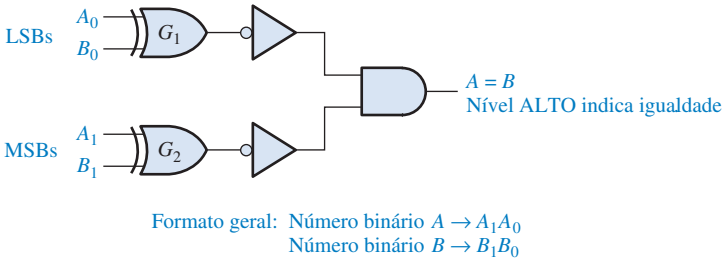
▲ **FIGURA 6–19**

Operação de um comparador básico.

Para comparar números binários com dois bits cada um, é necessário mais uma porta EX-OR. Os dois bits menos significativos (LSBs) dos dois números são comparados pela porta G1 e os dois bits mais significativos (MSBs) são comparados pela porta G2, como mostra a Figura 6–20. Se os dois números forem iguais, os bits correspondentes são iguais, sendo a saída de cada porta EX-OR nível zero. Se o conjunto correspondente de bits não forem iguais, a saída da porta EX-OR é nível 1.

▶ **FIGURA 6–20**

Diagrama lógico para comparação de igualdade de dois números de 2 bits. Abra o arquivo F06-20 para verificar a operação.



Um comparador determina se dois números binários são iguais ou diferentes.

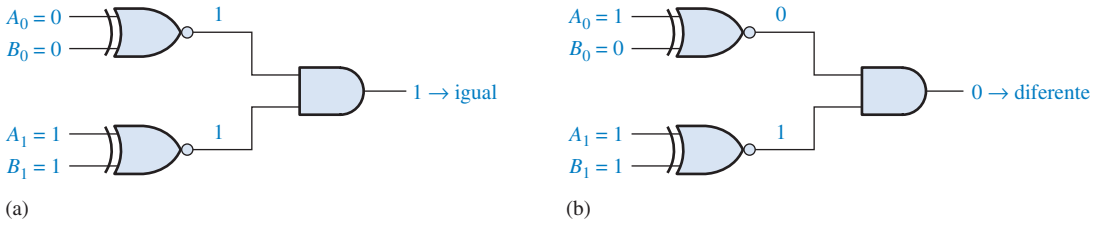
Para gerar uma única saída que indique uma igualdade ou desigualdade entre dois números, pode-se usar dois inversores e uma porta AND, conforme mostra a Figura 6–20. A saída de cada porta EX-OR é invertida e aplicada na entrada da porta AND. Quando os dois bits de entrada para cada porta EX-OR forem iguais, os bits correspondentes dos números são iguais, produzindo um nível 1 nas duas entradas da porta AND e conseqüentemente a saída da AND será nível 1. Quando os dois números não forem iguais, sendo diferente um ou ambos os conjuntos de bits correspondentes, faz aparecer um nível 0 em pelo menos uma das entradas da porta AND produzindo um nível 0 na saída dela. Portanto, a saída da porta AND indica igualdade (1) ou desigualdade (0) entre os dois números.

O Exemplo 6–5 ilustra essa operação para dois casos específicos. A porta EX-OR e o inversor são substituídos pelo símbolo de uma EX-NOR.

EXEMPLO 6–5

Considerando os seguintes conjuntos de números binários e o circuito comparador mostrado na Figura 6–21, determine a saída do circuito para cada conjunto.

(a) 10 e 10 **(b)** 11 e 10



▲ **FIGURA 6–21**

Solução **(a)** A saída é nível **1** para as entradas 10 e 10, conforme mostra a Figura 6–21(a).

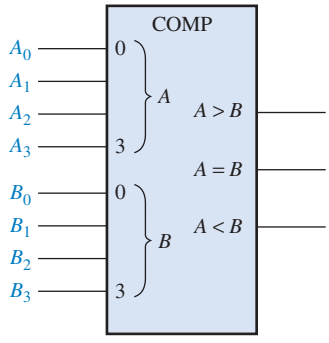
(b) A saída é nível **0** para as entradas 11 e 10, conforme mostra a Figura 6–21(b).

Problema relacionado Repita o processo para as entradas binárias 01 e 10.

Como sabemos do Capítulo 3, o comparador básico pode ser expandido para qualquer número de bits. A porta AND estabelece a condição em que todos os bits correspondentes nos dois números tem que ser iguais se os dos números forem iguais entre si.

Desigualdade

Além da saída de igualdade, muitos CIs comparadores provêem saídas adicionais que indicam qual dos dois números binários comparados é maior. Ou seja, existe uma saída que indica quando o número A é maior que o número B ($A > B$) e uma saída que indica quando o número A é menor que o número B ($A < B$), como mostra o símbolo lógico para um comparador de 4 bits na Figura 6–22.



◀ **FIGURA 6–22**
Símbolo lógico para um comparador de 4 bits com indicação de desigualdade.

NOTA: COMPUTAÇÃO

Num computador, a memória *cache* é uma memória intermediária muito rápida situada entre a unidade central de processamento (CPU) e a memória principal, que é mais lenta. A CPU solicita dados da memória enviando o *endereço* (localização única) dos dados. Parte desse endereço é denominado de *tag* (etiqueta). O comparador de endereço *tag* compara o *tag* proveniente da CPU com o *tag* proveniente do diretório da cache. Se os dois forem iguais, o dado endereçado já está na cache sendo obtido muito rapidamente. Se os *tags* forem diferentes, o dado tem que ser obtido da memória principal a uma taxa muito menor.

Para determinar uma desigualdade dos números binários A e B, temos que examinar primeiro o bit mais significativo de cada número. As seguintes condições são possíveis:

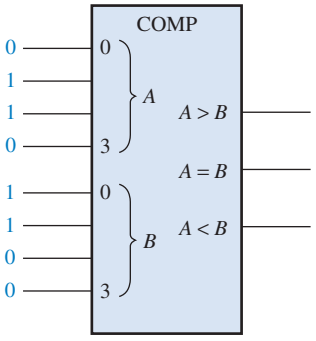
1. Se $A_3 = 1$ e $B_3 = 0$, o número A é maior que o número B.
2. Se $A_3 = 0$ e $B_3 = 1$, o número A é menor que o número B.
3. Se $A_3 = B_3$, então temos que examinar a desigualdade do próximo bit da posição mais inferior.

Essas três operações são válidas para a posição de cada bit nos números. O procedimento geral usado num comparador é verificar uma desigualdade numa posição de bit, começando pelos

bits mais significativos (MSBs). Quando tal desigualdade é identificada, a relação de dois números é estabelecida, sendo que qualquer outra desigualdade nas posições menos significativas tem que ser ignorada porque é possível que uma indicação oposta ocorra; *a indicação do mais significativo tem precedência.*

EXEMPLO 6–6

Determine as saídas $A = B$, $A > B$ e $A < B$ para os números de entradas mostrados no comparador visto na Figura 6–23.



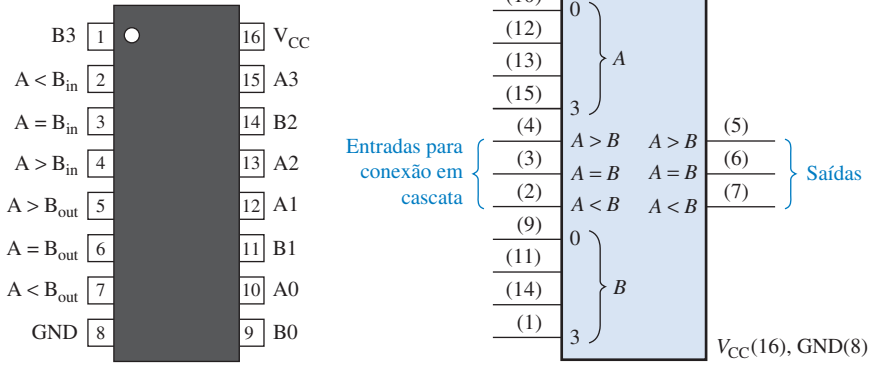
► FIGURA 6–23

Solução O número nas entradas A é 0110 e o número nas entradas B é 0011. A **saída $A > B$ é nível ALTO e as outras saídas são nível BAIXO**.

Problema relacionado Quais são as saídas do comparador quando $A_3A_2A_1A_0 = 1001$ e $B_3B_2B_1B_0 = 1010$?

UM COMPARADOR DE MAGNITUDE DE 4 BITS (74HC85)

O CI 74HC85 é um comparador que também é comercializado em outras famílias de CIs. O diagrama de pinos e o símbolo lógico são mostrados na Figura 6–24. Observe que esse dispositivo tem todas as entradas e as saídas do comparador generalizado discutido anteriormente e, além disso, tem três entradas para conexão em cascata: $A < B$, $A = B$, $A > B$. Essas entradas permitem que diversos comparadores sejam conectados em cascata para



► FIGURA 6–24

Diagrama de pinos e símbolo lógico para o CI 74HC85, um comparador de magnitude de 4 bits (os números dos pinos estão entre parênteses).

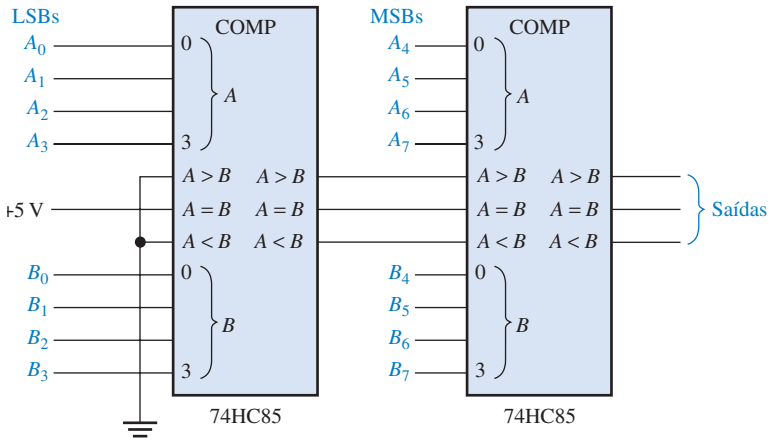
comparação de números com qualquer quantidade de bits maior que quatro. Para expandir o comparador, as saídas $A < B$, $A = B$ e $A > B$ do comparador menos significativo são conectadas às entradas de conexão em cascata correspondentes do próximo comparador de ordem maior. O comparador menos significativo tem que ter um nível ALTO na entrada $A = B$ e um nível BAIXO nas entradas $A < B$ e $A > B$. Esse dispositivo pode ser comercializado em outras famílias TTL e CMOS. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.

EXEMPLO 6–7

Use comparadores 74HC85 para comparar as magnitudes de dois números de 8 bits. Mostre o diagrama dos comparadores com as interconexões próprias.

Solução Dois CIs 74HC85 são necessários para comparar dois números de 8 bits. Eles são conectados num arranjo em cascata conforme mostra a Figura 6–25.

Problema relacionado Faça a expansão do circuito mostrado na Figura 6–25 para um comparador de 16 bits.



▶ **FIGURA 6–25**
Um comparador de magnitude de 8 bits usando dois CIs 74HC85.

Problema relacionado Faça a expansão do circuito mostrado na Figura 6–25 para um comparador de 16 bits.

SEÇÃO 6–4 REVISÃO

1. Os números binários $A = 1011$ e $B = 1010$ são aplicados nas entradas de um 74HC85. Determine as saídas.
2. Os números binários $A = 11001011$ e $B = 11010100$ são aplicados ao comparador de 8 bits mostrado na Figura 6–25. Determine os estados das saídas nos pinos 5, 6 e 7 em cada 74HC85.

DICA PRÁTICA

A maioria dos dispositivos CMOS contém um circuito de proteção contra danos a partir de tensões estáticas ou campos elétricos altos. Entretanto, precauções têm que ser tomadas para evitar a aplicação de tensões maiores que as tensões especificadas máximas. Para uma operação adequada, as tensões de entrada e saída devem estar entre V_{CC} e GND. Além disso, lembre-se que as entradas não usadas têm que ser conectadas sempre a um nível lógico apropriado (GND ou V_{CC}). As saídas não usadas podem ser deixadas em aberto.

6-5 DECODIFICADORES

Um **decodificador** é um circuito digital que detecta a presença de uma combinação específica de bits (código) em suas entradas indicando a presença desse código através de um nível de saída especificado. Em sua forma geral, um decodificador tem n linhas de entrada para manipular n bits e de uma a 2^n linhas de saída para indicar a presença de uma ou mais combinações de n bits. Nessa seção, diversos decodificadores são apresentados. Os princípios básicos podem ser estendidos para outros tipos de decodificadores.

Ao final do estudo desta seção você deverá ser capaz de:

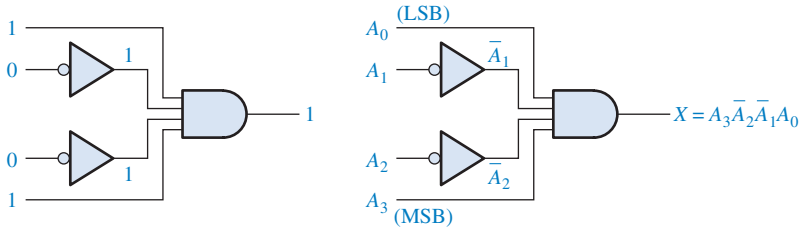
■ Definir *decodificador* ■ Projetar um circuito lógico para decodificar qualquer combinação de bits ■ Descrever o decodificador de binário para BCD (74HC154) ■ Expandir decodificadores para operar códigos com um grande número de bits ■ Descrever o decodificador de BCD para 7 segmentos (74LS47) ■ Discutir a supressão de zeros em displays de 7 segmentos ■ Usar decodificadores em aplicações específicas

NOTA: COMPUTAÇÃO

Uma *instrução* "diz" ao computador qual operação realizar. As instruções são códigos de máquina (1s e 0s) e, para o computador executar uma instrução, a instrução tem que ser decodificada. A decodificação de instrução é um dos passos da instrução *pipelining*, que são os seguintes: A instrução é lida na memória (busca da instrução), a instrução é decodificada, o(s) operando(s) é(são) lido(s) na memória (busca do operando), a instrução é executada e o resultado é escrito de volta na memória. Basicamente, o pipelining permite que o processamento da próxima instrução seja iniciado antes que o da atual seja completado.

Decodificador Binário Básico

Suponha que precisamos determinar quando um binário 1001 ocorre nas entradas de um circuito digital. Uma porta AND pode ser usada como o elemento de decodificação básico porque ela produz um nível ALTO na saída apenas quando todas as suas entradas estão em nível ALTO. Portanto, temos que ter certeza que todas as entradas da porta AND são nível ALTO quando ocorrer o número binário 1001; isso pode ser feito invertendo os dois bits do meio (os 0s), conforme mostra a Figura 6–26.



▲ **FIGURA 6–26**

Lógica de decodificação para o código binário 1001 com uma saída ativa em nível ALTO.

A equação lógica para o decodificador visto na Figura 6–26(a) é desenvolvida como ilustra a Figura 6–26(b). Devemos verificar que a saída é 0 exceto quando $A_0 = 1, A_1 = 0, A_2 = 0$ e $A_3 = 1$ forem aplicados nas entradas. A_0 é o LSB e A_3 é o MSB. *Na representação de um número binário ou outro código ponderado nesse livro, o LSB é o bit mais à direita numa representação horizontal e o bit mais alto numa representação vertical, a menos que seja especificado algo em contrário.*

Se uma porta NAND for usada no lugar da porta AND no circuito da Figura 6–26, uma saída de nível BAIXO indicará a presença do código binário próprio, que neste caso é 1001.

EXEMPLO 6–8

Determine a lógica necessária para decodificar o número binário 1011 produzindo um nível ALTO na saída.

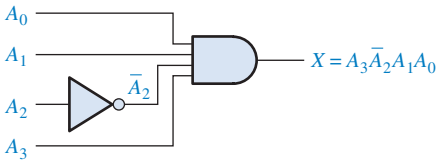
Solução A função de decodificação pode ser obtida complementando apenas as variáveis que aparecem como 0 no número binário desejado, como a seguir:

$$X = A_3\overline{A}_2A_1A_0 \qquad (1011)$$

Essa função pode ser implementada conectando as variáveis verdadeiras (não-complementadas) A_0, A_1 e A_3 diretamente nas entradas de uma porta AND e invertendo a variável A_2 antes de aplicá-la na entrada da porta AND. A lógica de decodificação é mostrada na Figura 6–27.

▶ FIGURA 6–27

Lógica de decodificação para gerar uma saída de nível ALTO quando 1011 estiver nas entradas.



Problema relacionado Desenvolva a lógica necessária para detectar o código binário 10010 e produzir uma saída ativa em nível BAIXO.

Decodificador de 4 Bits

Para decodificar todas as combinações possíveis de quatro bits, são necessárias dezesseis portas de decodificação ($2^4 = 16$). Esse tipo de decodificador é normalmente denominado *decodificador de 4 linhas para 16 linhas* porque existem quatro entradas e dezesseis saídas ou *decodificador 1 de 16* porque para um dado código nas entradas, uma das dezesseis saídas é ativada. A Tabela 6–4 mostra uma lista de dezesseis códigos binários e suas correspondentes funções de decodificação.

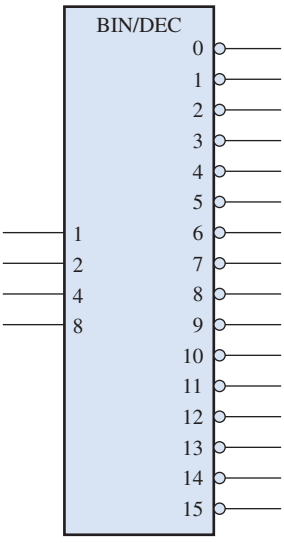
Se uma saída ativa em nível BAIXO for necessária para cada número decodificado, o decodificador completo pode ser implementado com portas NAND e inversores. Para decodificar cada um dos dezesseis códigos binários, são necessárias dezesseis portas NAND (portas AND podem ser usadas para produzir saídas ativas em nível ALTO).

▼ TABELA 6–4

Funções de decodificação e tabela-verdade para um decodificador de 4 linhas para 16 linhas (1 de 16) com saídas ativas em nível BAIXO

DÍGITO DECIMAL	ENTRADAS BINÁRIAS				FUNÇÃO DE DECODIFICAÇÃO	SAÍDAS														
	A_3	A_2	A_1	A_0		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
10	1	0	1	0	$A_3\overline{A}_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
11	1	0	1	1	$A_3\overline{A}_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
12	1	1	0	0	$A_3A_2\overline{A}_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
13	1	1	0	1	$A_3A_2\overline{A}_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
14	1	1	1	0	$A_3A_2A_1\overline{A}_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
15	1	1	1	1	$A_3A_2A_1A_0$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Um símbolo lógico para um decodificador de 4 linhas para 16 linhas (1 de 16) com saídas ativas em nível BAIXO é mostrado na Figura 6–28. A denominação BIN/DEC indica que uma entrada binária ativa a correspondente saída decimal. As denominações de entrada 8, 4, 2 e 1 representam os pesos binários dos bits de entrada ($2^3 2^2 2^1 2^0$).

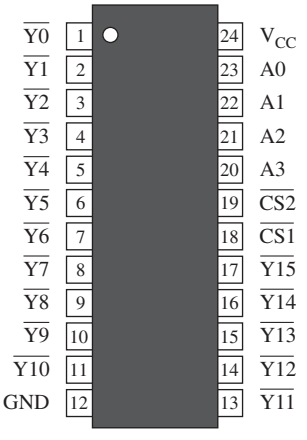


▶ **FIGURA 6–28**

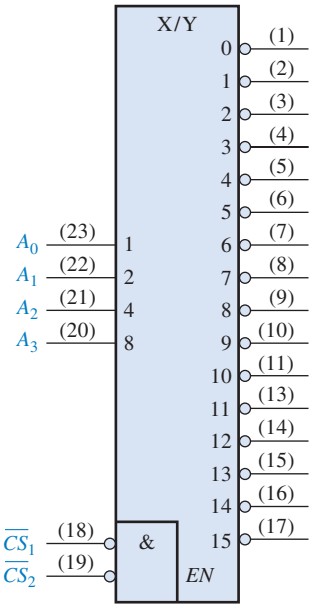
Funções de decodificação e tabela-verdade para um decodificador de 4 linhas para 16 linhas (1 de 16) com saídas ativas em nível BAIXO.

UM DECODIFICADOR 1 DE 16 (74HC154)

O CI 74HC154 é um bom exemplo de um decodificador. O símbolo lógico é mostrado na Figura 6–29. Existe uma função de habilitação (*EN*) fornecida nesse dispositivo, a qual é implementada com uma porta NOR usada com uma AND negativa. Um nível BAIXO em cada entrada de seleção de chip, $\overline{CS}_1$ e $\overline{CS}_2$, é necessário para tornar nível ALTO a saída da porta de habilitação (*EN*). A saída da porta de habilitação é conectada na entrada de cada porta NAND no decodificador, assim ela tem que ser nível ALTO para as portas NAND serem habilitadas. Se a porta de habilitação não for ativada por um nível BAIXO nas duas



(a) Diagrama de pinos



(b) Símbolo lógico

▶ **FIGURA 6–29**

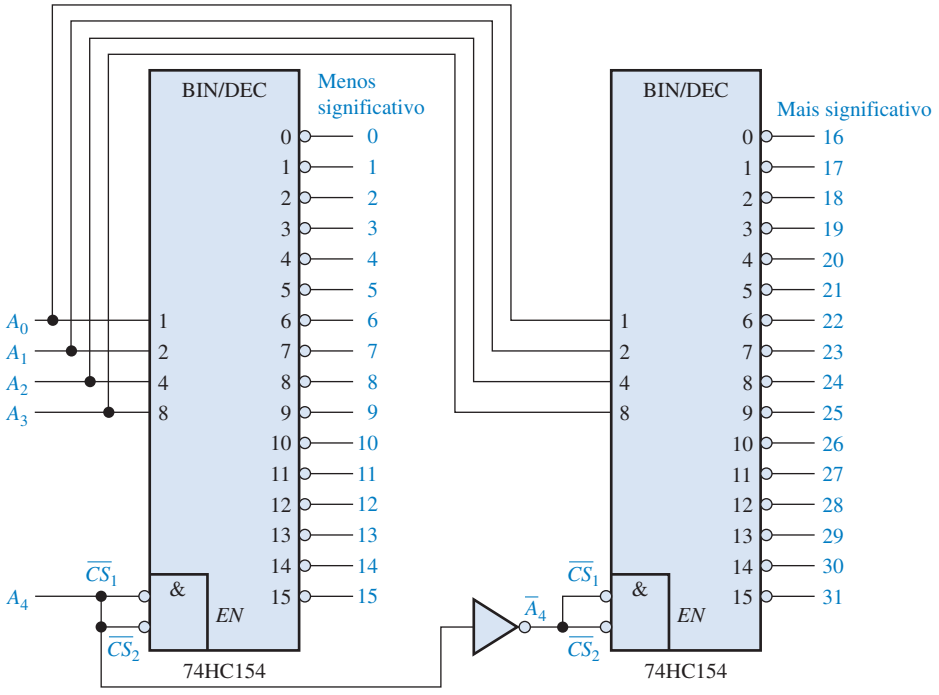
Diagrama de pino e símbolo lógico para o decodificador 1 de 16 (74HC154).

entradas, então todas as dezesseis saídas do decodificador (Y) estarão em nível ALTO independente dos estados das quatro variáveis de entrada (A_0, A_1, A_2 e A_3). Esse dispositivo pode ser comercializado em outras famílias CMOS ou TTL. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.

EXEMPLO 6–9

Certa aplicação necessita que um número de 5 bits seja decodificado. Use CIs decodificadores para implementar a lógica. O número binário é representado pelo formato $A_4A_3A_2A_1A_0$.

Solução Como o CI 74HC154 pode operar apenas quatro bits, temos que usar dois decodificadores para decodificar 5 bits. O quinto bit (A_4) é conectado às entradas de seleção de chip $\overline{CS}_1$ e $\overline{CS}_2$ de um decodificador, e $\overline{A}_4$ é conectado às entradas $\overline{CS}_1$ e $\overline{CS}_2$ do outro decodificador, como mostra a Figura 6–30. Quando o número decimal for 15 ou menor, $A_4 = 0$, o decodificador menos significativo é habilitado e o decodificador mais significativo é desabilitado. Quando o número decimal for maior que 15, $A_4 = 1$ sendo $\overline{A}_4 = 0$, o decodificador mais significativo é habilitado e o decodificador menos significativo é desabilitado.



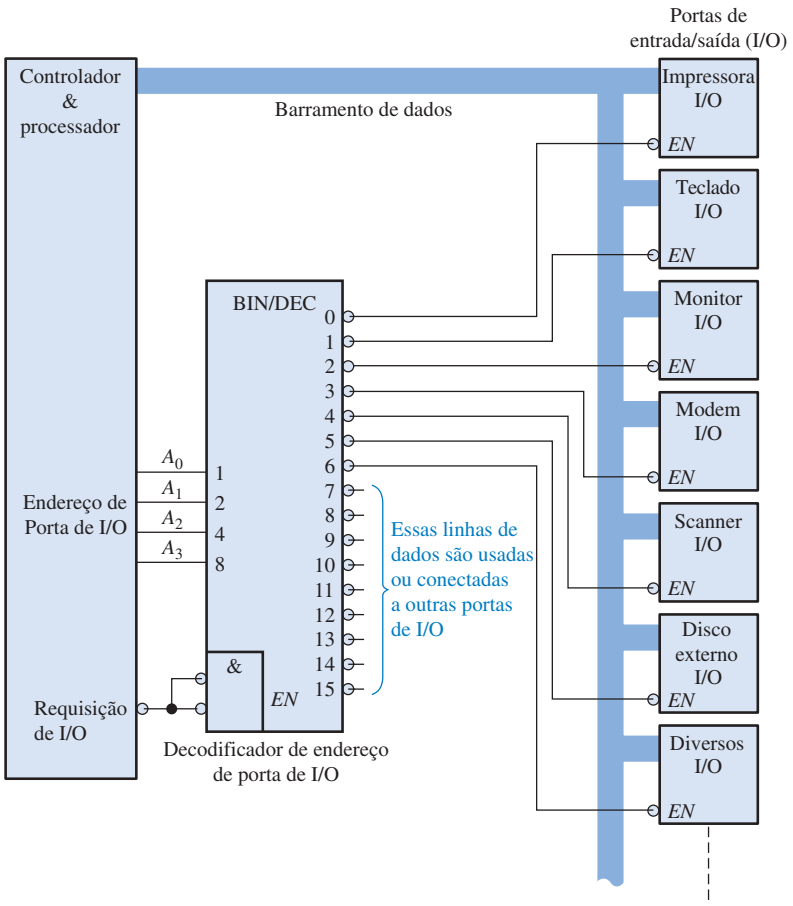
▶ **FIGURA 6–30**
Um decodificador de 5 bits usando CIs 74HC154.

Problema relacionado Determine a saída do circuito da Figura 6–30 que é ativada para a entrada binária 10110.

Uma Aplicação

Decodificadores são usados em muitos tipos de aplicações. Um exemplo é usado em computadores para seleção de entrada/saída conforme ilustrado no diagrama geral da Figura 6–31.

Os computadores têm que se comunicar com uma variedade de dispositivos externos denominados *periféricos* enviando e/ou recebendo dados através do que é conhecido como portas de entrada/saída (I/O). Esses dispositivos externos incluem impressoras, modems, scanners, acionadores de disco externos, teclados, monitores de vídeo e outros computadores. Conforme indicado na



► FIGURA 6–31

Um sistema simplificado de porta de I/O de computador com um decodificador de endereço de porta de I/O com apenas quatro linhas de endereço sendo mostradas.

Figura 6–31, um decodificador é usado para selecionar a porta de I/O conforme determinado pelo computador de forma que os dados podem ser enviados ou recebidos de um dispositivo externo específico.

Cada porta de I/O tem um número, denominado endereço, o qual o identifica unicamente. Quando o computador "quer" se comunicar com um dispositivo em particular, ele emite o código de endereço apropriado para a porta de I/O na qual o dispositivo em particular está conectado. Esse endereço binário de porta de I/O é decodificado e a saída apropriada do decodificador é ativada para habilitar a porta de I/O.

Conforme mostra a Figura 6–31, os dados binários são transferidos internamente ao computador através do barramento de dados, o qual é constituído de um conjunto de linhas paralelas. Por exemplo, um barramento de 8 bits consiste de oito linhas em paralelo que transportam um byte de dados de cada vez. O barramento de dados alcança todos as portas de I/O, porém todos os dados que entram ou saem passam através da porta de I/O que está habilitada pelo decodificador de endereço de porta de I/O.

Decodificador de BCD para Decimal

O decodificador de BCD para decimal converte cada código BCD (código 8421) em uma das dez indicações decimais possíveis. Ele é freqüentemente referido como um *decodificador de 4 linhas para 10 linhas* ou um *decodificador 1 de 10*.

O método de implementação é o mesmo que para o decodificador 1 de 16 discutido anteriormente, exceto que são necessárias apenas dez portas de decodificação porque o código BCD representa apenas os dígitos decimais de 0 a 9. A Tabela 6–5 apresenta uma lista de dez códigos BCD e suas correspondentes funções de decodificação. Cada uma dessas funções de decodificação é implementada com portas NAND para prover saídas ativas em nível BAIXO. Se for necessário uma saída ativa em nível ALTO, são usadas portas AND para decodificação. A lógica de decodificação é idêntica às dez primeiras portas de decodificação do decodificador 1 de 16 (veja a Tabela 6–4).

TABELA 6–5

Funções de decodificação BCD

DÍGITO DECIMAL	CÓDIGO BCD A_3	A_2	A_1	A_0	FUNÇÃO DE DECODIFICAÇÃO
0	0	0	0	0	$\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$
1	0	0	0	1	$\overline{A}_3\overline{A}_2\overline{A}_1A_0$
2	0	0	1	0	$\overline{A}_3\overline{A}_2A_1\overline{A}_0$
3	0	0	1	1	$\overline{A}_3\overline{A}_2A_1A_0$
4	0	1	0	0	$\overline{A}_3A_2\overline{A}_1\overline{A}_0$
5	0	1	0	1	$\overline{A}_3A_2\overline{A}_1A_0$
6	0	1	1	0	$\overline{A}_3A_2A_1\overline{A}_0$
7	0	1	1	1	$\overline{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\overline{A}_2\overline{A}_1\overline{A}_0$
9	1	0	0	1	$A_3\overline{A}_2\overline{A}_1A_0$

EXEMPLO 6–10

O CI 74HC42 é um decodificador de BCD para decimal. O símbolo lógico é mostrado na Figura 6–32. Se as formas de onda de entrada vistas na Figura 6–33(a) são aplicadas nas entradas do CI 74HC42, mostre as formas de onda de saída.

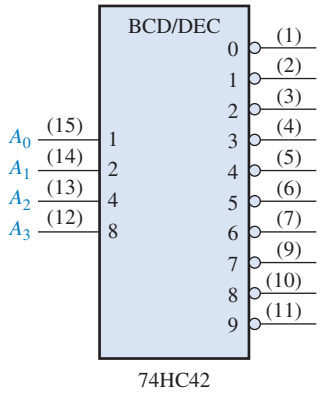


FIGURA 6–32

Decodificador de BCD para decimal (74HC42).

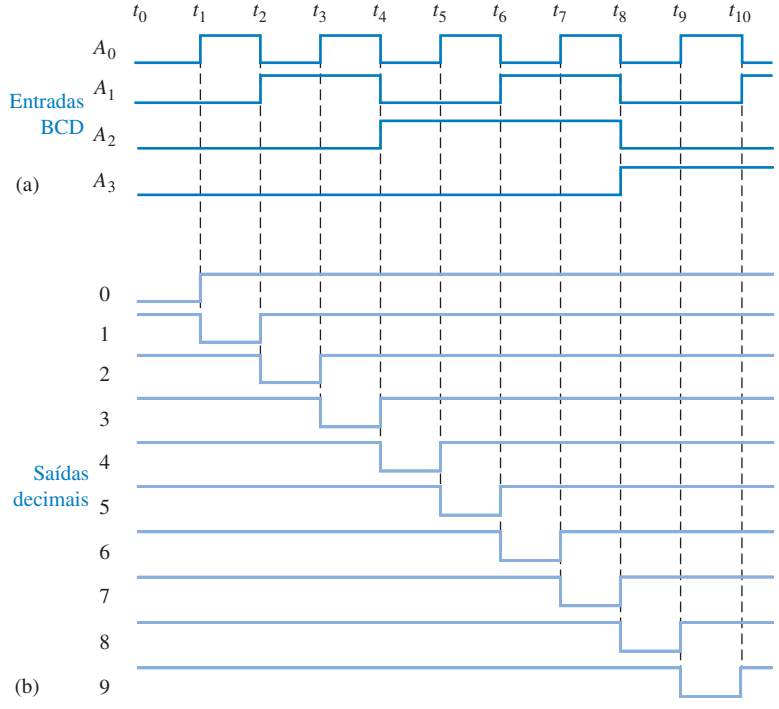


FIGURA 6–33

Solução As formas de onda de saída são mostradas na Figura 6–33(b). Como podemos ver, as entradas são uma seqüência BCD para os dígitos de 0 a 9. As formas de onda de saída no diagrama de temporização indicam essa seqüência BCD nas saídas de valores decimais.

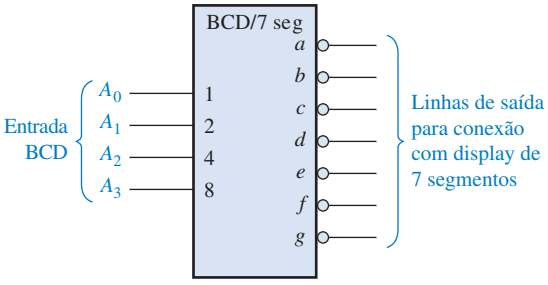
Problema relacionado Construa um diagrama de temporização mostrando as formas de onda de entrada e saída para o caso em que a seqüência de números decimais através das entradas BCD é a seguinte: 0, 2, 4, 6, 8, 1, 3, 5 e 9.

Decodificador de BCD para 7 Segmentos

O decodificador de BCD para 7 segmentos aceita o código BCD em suas entradas e fornece saídas para acionar displays de 7 segmentos para produzir uma leitura decimal. O diagrama lógico para um decodificador de 7 segmentos básico é mostrado na Figura 6–34.

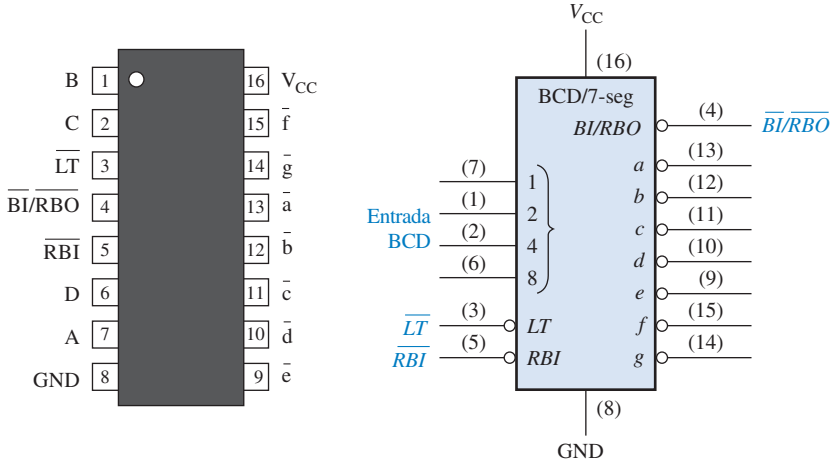
▶ FIGURA 6–34

Símbolo lógico para um decodificador/driver de BCD para 7 segmentos com saídas ativas em nível BAIXO. Abra o arquivo F06-34 para verificar a operação.



UM DECODIFICADOR/DRIVER DE BCD PARA 7 SEGMENTOS (74LS47)

O CI 74LS47 é um exemplo de um CI que decodifica uma entrada BCD e aciona um display de 7 segmentos. Além dessa capacidade de decodificação e acionamento de segmento, o CI 64LS47 tem algumas características adicionais conforme indicado pelas funções $\overline{LT}$, $\overline{RBI}$, $\overline{BI}/\overline{RBO}$ no símbolo lógico visto na Figura 6–35. Conforme indicado pelos pequenos círculos no símbolo lógico, todas as saídas (de *a* a *g*) são ativas em nível baixo como são as funções $\overline{LT}$ (teste de lâmpada), *RBI* (entrada de apagamento) e $\overline{BI}/\overline{RBO}$ (entrada de apagamento/saída de apagamento). As saídas podem acionar diretamente um display de 7 segmentos do tipo anodo comum. Lembre-se que os displays de 7 segmentos foram discutidos no Capítulo 4. Além de decodificar uma entrada BCD e produzir as saídas apropriadas de 7 segmentos, o CI 74LS47 tem capacidade de teste de lâmpada e supressão de zero. Esse dispositivo pode ser comercializado em outras famílias TTL ou CMOS. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha este livro.



▶ FIGURA 6–35

Diagrama de pinos e símbolo lógico para o CI 74LS47 (decodificador/driver de BCD para 7 segmentos).

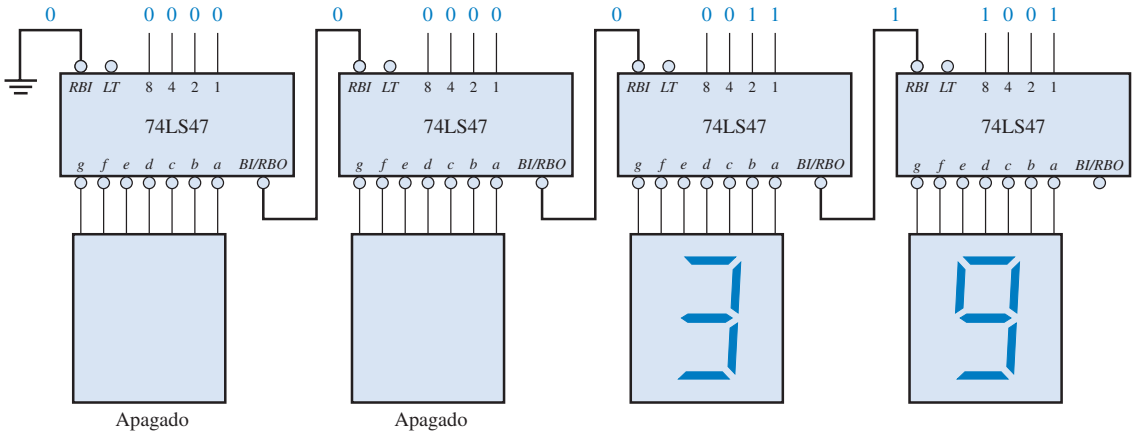
Teste de Lâmpada Quando um nível BAIXO é aplicado na entrada $\overline{LT}$ e $\overline{BI}/\overline{RBO}$ for nível ALTO, todos os 7 segmentos do display são ligados. O teste de lâmpada é usado para verificar se algum segmento está queimado.

Supressão de Zero A **supressão de zero** é uma característica usada por displays de múltiplos dígitos para apagar os zeros não necessários. Por exemplo, num display de 6 dígitos o número 6,4 pode ser mostrado como 006,400 se os zeros não forem apagados. O apagamento dos zeros no início do número é denominado de *supressão de zeros mais significativos* e o apagamento de zeros no final do número é denominado de *supressão de zeros menos significativos*. Tenha em mente que apenas os zeros não necessários são apagados. Com a supressão de zeros o número 030,080 será mostrado como 30,08 (os zeros essenciais são mantidos).

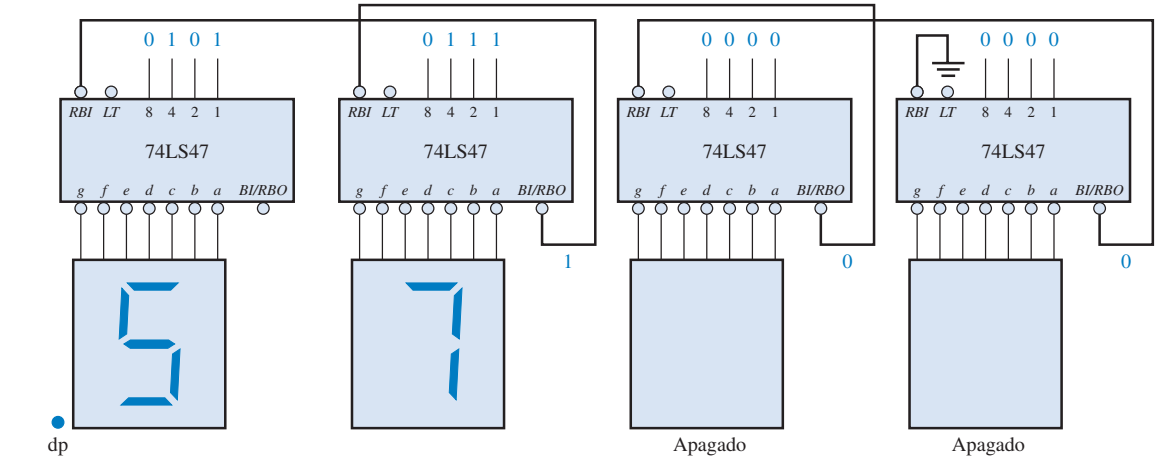
A supressão de zeros resulta em zeros mais significativos e menos significativos num número não mostrado no display.

A supressão de zero no CI 74LS47 é realizada usando as funções $\overline{RBI}$ e $\overline{BI}/\overline{RBO}$. $\overline{RBI}$ é a entrada de apagamento e $\overline{RBO}$ é a saída de apagamento no 74LS47; essas são usadas para supressão de zeros. $\overline{BI}$ é a entrada de apagamento que compartilha o mesmo pino com $\overline{RBO}$; em outras palavras, o pino $\overline{BI}/\overline{RBO}$ pode ser usado como uma entrada ou uma saída. Quando usado como $\overline{BI}$ (entrada de apagamento), todas as saídas de segmentos são nível ALTO (desativadas) quando $\overline{BI}$ for nível BAIXO, o que anula todas as outras entradas. A função $\overline{BI}$ não faz parte da capacidade de supressão de zeros do dispositivo.

Todas as saídas de segmentos estarão desativadas (nível ALTO) se um código zero (0000) for colocado nas entradas BCD e se sua entrada $\overline{RBI}$ estiver em nível BAIXO. Isso faz com que o display apague e produza um nível BAIXO em $\overline{RBO}$.



(a) Ilustração da supressão de zeros mais significativos.



(b) Ilustração da supressão de zeros menos significativos.

▲ **FIGURA 6–36**

Exemplo de supressão de zeros usando um decodificador/driver de BCD para 7 segmentos (74LS47).

O diagrama lógico na Figura 6–36(a) ilustra a supressão de zeros mais significativos para um número inteiro. A posição do dígito mais significativo (mais à esquerda) estará sempre apagada se um código zero estiver nas entradas BCD porque a entrada $\overline{RBI}$ do decodificador mais significativo é colocada em nível BAIXO pela conexão em GND. A saída $\overline{RBO}$ de cada decodificador é conectada à entrada $\overline{RBI}$ do próximo decodificador de menor ordem de forma que todos os zeros à esquerda do primeiro dígito diferente de zero sejam apagados. Por exemplo, na parte (a) da figura os dois dígitos mais significativos são zeros e, portanto, estão apagados. Os dois dígitos restantes, 3 e 9, são mostrados.

O diagrama lógico visto na Figura 6–36(b) ilustra a supressão de zeros menos significativos para um número fracionário. O dígito de menor ordem (mais à direita) é sempre apagado se o código do zero estiver nas entradas BCD porque a entrada $\overline{RBI}$ está conectada em GND. A saída $\overline{RBO}$ de cada decodificador está conectada na entrada $\overline{RBI}$ do próximo decodificador de ordem maior de forma que todos os zeros à direita do primeiro dígito diferente de zero são apagados. Na parte (b) da figura, os dois dígitos de menor ordem são zeros e, portanto, são apagados. Os dois dígitos restantes, 5 e 7, são mostrados. Para combinar a supressão de zeros mais e menos significativos em um display e ter a capacidade de indicação de ponto (vírgula) decimal, é necessária uma lógica adicional.

SEÇÃO 6–5 REVISÃO

1. Um decodificador de 3 linhas para 8 linhas pode ser usado para a decodificação de octal para decimal. Quando um binário 101 for colocado nas entradas, qual linha de saída é ativada?
2. Quantos CIs 74HC154 (decodificador 1 de 16) são necessários para decodificar um número binário de 6 bits?
3. Você escolheria um decodificador/driver com saídas ativas em nível ALTO ou BAIXO para acionar um display de LEDs do tipo catodo comum?

6-6 CODIFICADORES

Um **codificador** é um circuito lógico que realiza essencialmente a função "inversa" do decodificador. Um codificador aceita um nível ativo em uma de suas entradas representando um dígito, tal como um dígito decimal ou octal, e o converte em uma saída codificada, tal como binário ou BCD. Codificadores também podem ser implementados para codificar vários símbolos e caracteres alfabéticos. O processo de conversão de símbolos familiares ou números para um formato codificado é denominado de *codificação*.

Ao final do estudo desta seção você deverá ser capaz de:

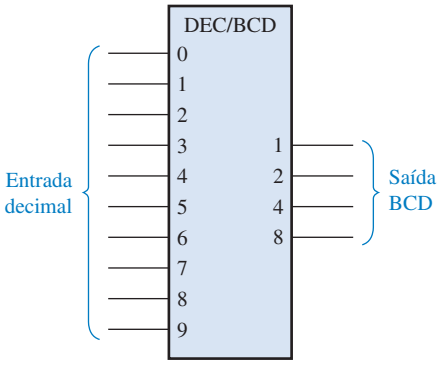
■ Determinar a lógica para um codificador decimal ■ Explicar a finalidade da característica de prioridade em codificadores ■ Descrever um codificador de prioridade de decimal para BCD (74HC154) ■ Descrever um codificador de prioridade de octal para binário (74LS148) ■ Expandir um codificador ■ Usar um codificador numa aplicação específica

Codificador de Decimal para BCD

Este tipo de codificador tem dez entradas – uma para cada dígito decimal – e quatro saídas correspondentes ao código BCD, conforme mostra a Figura 6–37. Esse é um codificador básico de 10 linhas para 4 linhas.

O código BCD (8421) é mostrado na Tabela 6–6. A partir dessa tabela podemos determinar a relação entre cada bit BCD e os dígitos decimais em ordem para analisar a lógica. Por exemplo, o bit mais significativo do código BCD, A_3, é sempre nível 1 para o dígito decimal 8 ou 9. Portanto, pode-se escrever uma expressão OR para o bit A_3 em termos dos dígitos decimais como

$$A_3 = 8 + 9$$



◄ FIGURA 6–37

Símbolo lógico para um codificador de decimal para BCD.

◄ TABELA 6–6

DÍGITO DECIMAL	CÓDIGO BCD A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

O bit A_2 é sempre nível 1 para o dígito decimal 4, 5, 6 ou 7 e pode ser expresso como uma função OR conforme a seguir:

$$A_2 = 4 + 5 + 6 + 7$$

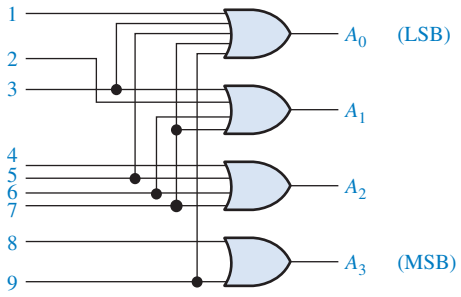
O bit A_1 é sempre nível 1 para o dígito decimal 2, 3, 6 ou 7 e pode ser expresso como

$$A_1 = 2 + 3 + 6 + 7$$

Finalmente, A_0 é sempre nível 1 para o dígito decimal 1, 3, 5, 7 ou 9. A expressão para A_0 é

$$A_0 = 1 + 3 + 5 + 7$$

Agora vamos implementar o circuito lógico necessário para a codificação de cada dígito decimal para o código BCD usando as expressões lógicas desenvolvidas. Para formar cada saída BCD basta simplesmente realizar uma operação OR entre as linhas de entrada dos dígitos decimais apropriados. A lógica do codificador básico resultante dessas expressões é mostrada na Figura 6–38.



◄ FIGURA 6–38

Diagrama lógico básico de um codificador de decimal para BCD. Uma entrada de dígito 0 não é necessária porque as saídas BCD são todas nível BAIXO quando não existirem entradas em nível ALTO.

NOTA: COMPUTAÇÃO

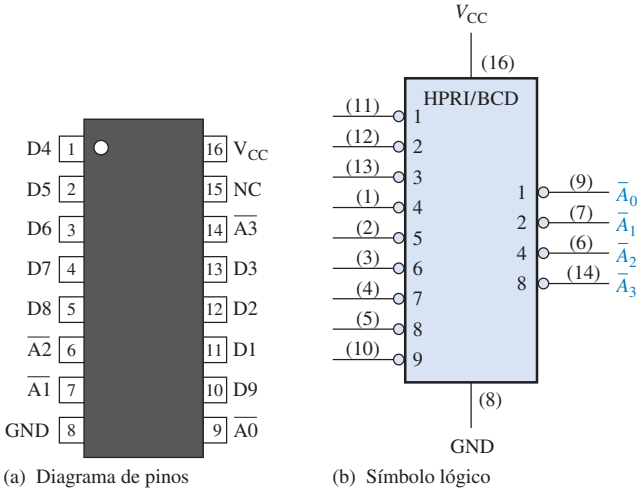
Um *assemblador* pode ser idealizado como um software codificador porque ele interpreta as instruções mnemônicas com as quais um programa é escrito e executa a codificação aplicável para converter cada mnemônico para uma instrução em código de máquina (uma série de 1s e 0s) a qual o computador pode entender. Como exemplos de mnemônicos temos ADD, MOV (mover dados), MUL (multiplicar), XOR, JMP (jump) e OUT (saída para uma porta de I/O).

A operação básica do circuito visto na Figura 6–38 é a seguinte: quando um nível ALTO aparece em uma das linhas de entrada de dígito decimal, os níveis apropriados aparecem nas quatro linhas de saída BCD. Por exemplo, se a linha de entrada 9 for nível ALTO (considerando que todas as outras linhas de entrada estejam em nível BAIXO), essa condição produzirá um nível ALTO nas saídas A_0 e A_3 e um nível BAIXO nas saídas A_1 e A_2, que é o código BCD (1001) para o decimal 9.

Codificador de Prioridade de Decimal para BCD Esse tipo de codificador realiza a mesma função de codificação básica discutida anteriormente. Um **codificador de prioridade** oferece também uma flexibilidade adicional na qual ele pode ser usado em aplicações que requerem detecção de prioridade. A função de prioridade significa que o codificador produzirá uma saída BCD correspondente à entrada do *dígito decimal mais significativo* que estiver ativado ignorando qualquer outra entrada ativa menos significativa. Por exemplo, se as entradas 6 e 3 estiverem ativas, a saída BCD será 0110 (que representa o decimal 6).

UM CODIFICADOR DE DECIMAL PARA BCD (74HC147)

O CI 74HC147 é um codificador de prioridade com entradas ativas em nível BAIXO (0) para os dígitos decimais de 1 a 9 e saídas BCD ativas em nível BAIXO conforme indicado no símbolo lógico mostrado na Figura 6–39. Uma saída BCD zero é representada quando nenhuma das entradas estiver ativa. Os números dos pinos do dispositivo estão entre parênteses. Esse dispositivo pode ser comercializado em outras famílias CMOS ou TTL. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.

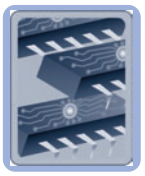


(a) Diagrama de pinos (b) Símbolo lógico

► **FIGURA 6–39**

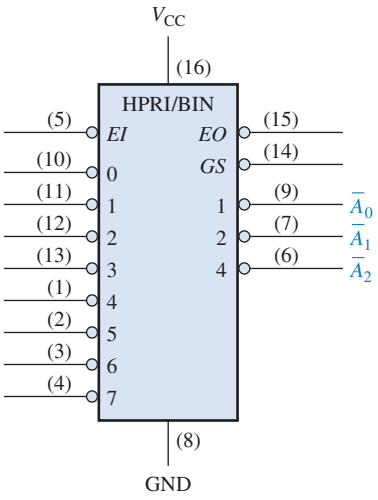
Diagrama de pinos e símbolo lógico para o codificador de prioridade de decimal para BCD 74HC147 (HPRI significa *highest value input has priority* – a entrada de valor mais alto tem prioridade).

UM CODIFICADOR DE 8 LINHAS PARA 3 LINHAS (74LS148)



O CI 74LS148 é um codificador de prioridade que tem oito entradas ativas em nível baixo e três saídas binárias ativas em nível ALTO, conforme mostra a Figura 6–40. Esse dispositivo pode ser usado para converter entradas octal (lembre-se que os dígitos octais são de 0 a 7) para um código binário de 3 bits. Para habilitar o dispositivo, a entrada *EI* (entrada de habilitação) tem que ser nível BAIXO. Ele também tem a saída *EO* (saída de habilitação) e a saída *GS* para fins de expansão. A saída *EO* é nível BAIXO quando a entrada *EI* for nível BAIXO e nenhuma das entradas (de 0 a 7) estiver ativa. A saída *GS* é nível BAIXO quando a entrada *EI* for nível BAIXO e qualquer uma das entradas estiver ativa. Esse dispositivo pode

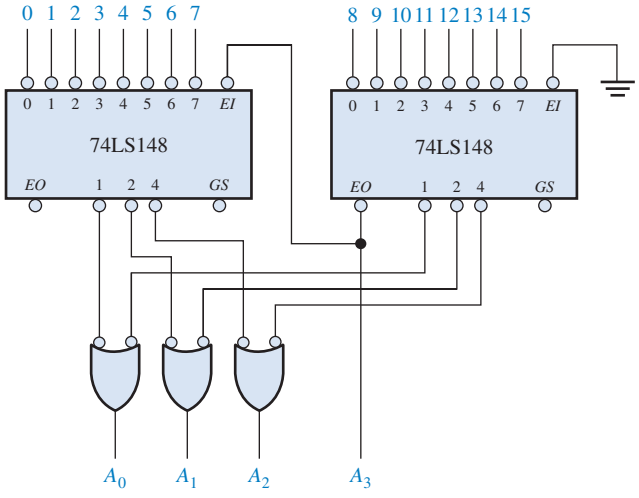
ser comercializado em outras famílias TTL ou CMOS. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.



◀ FIGURA 6–40

Símbolo lógico para o codificador de 8 para 3 linhas 74LS148.

O CI 74LS148 pode ser expandido para um codificador de 16 linhas para 4 linhas conectando a saída *EO* do codificador mais significativo na entrada *EI* do codificador menos significativo e fazendo uma operação OR negativa entre as correspondentes saídas binárias como mostra a Figura 6–41. A saída *EO* é usada como o quarto bit (MSB). Essa configuração particular produz saídas ativas em nível ALTO para o número binário de 4 bits.



◀ FIGURA 6–41

Um codificador de 16 linhas para 4 linhas usando CIs 74LS148 e lógica externa.

EXEMPLO 6–11

Se aparecerem níveis BAIXOs nos pinos 1, 4 e 13 do CI 74HC147 mostrado na Figura 6–39, indique o estado das quatro saídas. Todas as outras entradas estão em nível ALTO.

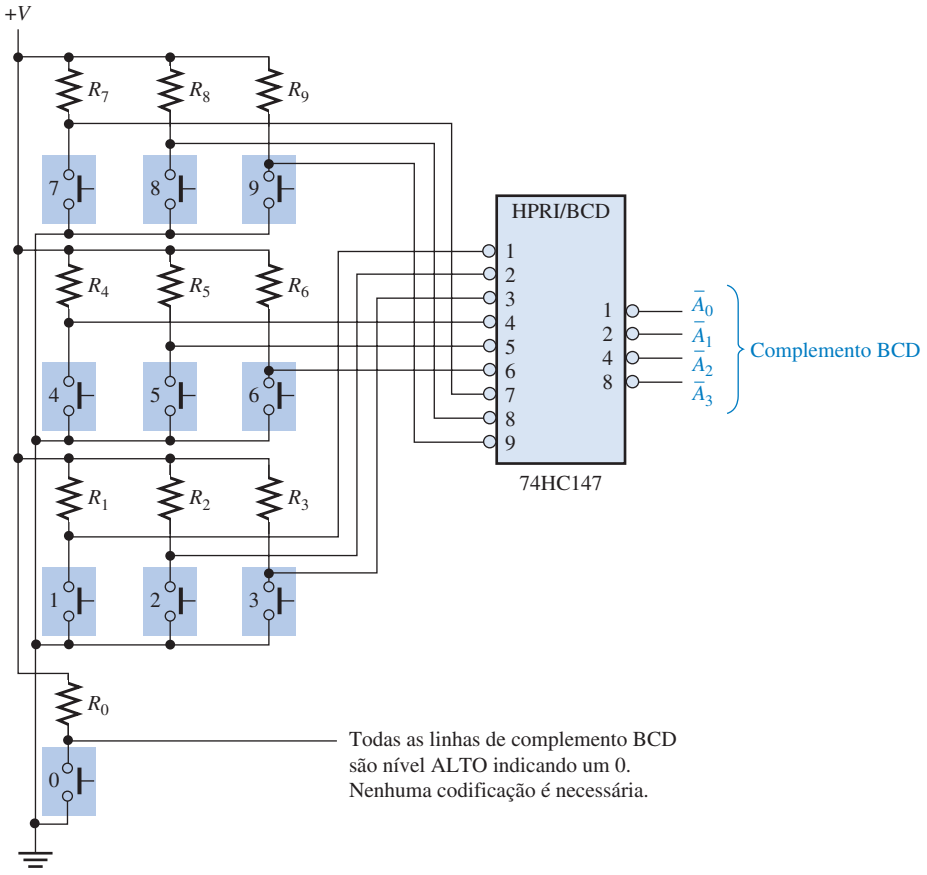
Solução O pino 4 é a entrada de dígito decimal mais significativo que tem um nível BAIXO e representa o decimal 7. Portanto, os níveis de saída indicam o código BCD para o decimal 7 onde $\overline{A}_0$ é o LSB e $\overline{A}_3$ é o MSB. A saída $\overline{A}_0$ é nível BAIXO, $\overline{A}_1$ é nível BAIXO, $\overline{A}_2$ é nível BAIXO e $\overline{A}_3$ é nível ALTO.

Problema relacionado Quais são as saídas do CI 74HC147 se todas as entradas estiverem em nível BAIXO? E se todas as entradas estiverem em nível ALTO?

Uma Aplicação

Um exemplo de aplicação clássica é um codificador de teclado. Os dez dígitos decimais no teclado de um computador, por exemplo, tem que ser codificado para ser processado pelo circuito lógico. Quando uma das teclas é pressionada, o dígito decimal é codificado para o código BCD correspondente. A Figura 6–42 mostra a configuração de um codificador de teclado simples usando um CI codificador de prioridade 74HC147. As teclas são representadas por dez chaves push-button, cada uma com um **resistor de pull-up** (elevador) para +V. O resistor de pull-up garante que a linha será nível ALTO quando a tecla não estiver pressionada. Quando a tecla for pressionada, a linha é conectada em GND, sendo que um nível BAIXO é aplicado na entrada correspondente do codificador. A tecla zero não é conectada porque a saída BCD representa zero quando nenhuma das teclas é pressionada.

A saída BCD complementada do codificador vai para um dispositivo de armazenamento, sendo que cada código BCD sucessivo é armazenado até que o número completo tenha sido digitado. Os métodos para armazenamento de números BCD e dados binários são abordados em capítulos posteriores.



▶ **FIGURA 6–42**
Um codificador de teclado simplificado.

SEÇÃO 6–6 REVISÃO

1. Suponha que sejam aplicados níveis ALTOs nas entradas 2 e 9 do circuito visto na Figura 6–38.
- **(a)** Quais são os estados das linhas de saída?
- **(b)** Elas representam um código BCD válido?
- **(c)** Qual é a restrição na lógica de codificação mostrada na Figura 6–38?

2. (a) Quais são os estados das saídas $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ quando são aplicados níveis BAIXOs nos pinos 1 e 5 do CI 74HC147 visto na Figura 6–39?
- **(b)** O que essa saída representa?

6-7 CONVERSORES DE CÓDIGOS

Nesta seção, analisaremos alguns métodos de utilização de circuitos lógicos combinacionais para converter de um código para outro.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar o processo da conversão de binário para BCD ■ Usar portas EX-OR para conversões entre binário e códigos Gray

Coversão de BCD para Binário

Um método de conversão de código BCD para binário usa circuitos somadores. O processo básico de conversão é o seguinte:

1. O valor, ou o peso, de cada bit do número BCD é representado por um número binário.
2. Todas as representações binárias dos pesos dos bits que são 1s no número BCD são somadas.
3. O resultado dessa adição é o equivalente binário do número BCD.

Uma descrição mais concisa dessa operação é:

Os números binários que representam os pesos dos bits BCD são somados para produzir o número binário total.

Vamos examinar um código BCD de 8 bits (que representa um número decimal de 2 dígitos) para entender as relações entre BCD e binário. Por exemplo, já sabemos que o número decimal 87 pode ser expresso em BCD como

$$\underbrace{1000}_{8}\ \ \underbrace{0111}_{7}$$

O grupo de 4 bits mais à esquerda representa 80 e o grupo de 4 bits mais à direita representa 7. Ou seja, o grupo mais à esquerda tem um peso de 10 e o grupo mais à direita tem um peso de 1. Dentro de cada grupo, o peso binário de cada bit é o seguinte:

	Dezenas				**Unidades**			
Peso:	80	40	20	10	8	4	2	1
Indicação de bit:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

O equivalente binário de cada bit BCD é um número binário que representa o peso desse bit dentro do número BCD total. Essa representação é dada na Tabela 6–7.

◀ **TABELA 6–7**
Representações em binário de pesos de bits BCD

BIT BCD	PESO EM BCD	(MSB) REPRESENTAÇÃO BINÁRIA (LSB)						
		64	32	16	8	4	2	1
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

Se as representações binárias para os pesos de todos os 1s no número BCD são somadas, o resultado é o número binário que corresponde ao número BCD. O Exemplo 6–12 ilustra isso.

EXEMPLO 6–12

Converta os números BCD 00100111 (decimal 27) e 100110000 (decimal 98) para binário.

Solução Escreva as representações binárias dos pesos de todos os 1s que aparecem nos números para, em seguida, somá-los.

80	40	20	10	8	4	2	1		
0	0	1	0	0	1	1	1		
								0000001	1
								0000010	2
								0000100	4
								+ 0010100	20
								0011011	Número binário para o decimal 27

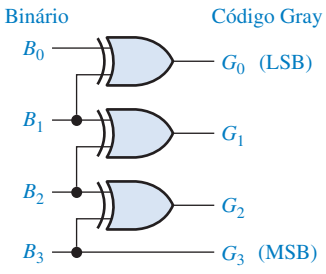
80	40	20	10	8	4	2	1		
1	0	0	1	1	0	0	0		
								0001000	8
								0001010	10
								+ 1010000	80
								1100010	Número binário para o decimal 98

Problema relacionado Mostre o processo de conversão do número BCD 01000001 para binário.

Com esse procedimento básico em mente, vamos ver como o processo pode ser implementado com circuitos lógicos. Uma vez que a representação binária para cada 1 no número BCD seja determinada, circuitos somadores podem ser usados para somar os 1s em cada coluna da representação binária. Os 1s ocorrem numa dada coluna apenas quando o bit BCD correspondente for 1. A ocorrência de um BCD 1 pode portanto ser usada para gerar o binário 1 próprio na coluna apropriada na estrutura do somador. Para operar com um código BCD de dois dígitos decimais (duas décadas), são necessárias oito linhas de entrada BCD e sete saídas binárias. (São usados sete bits para representar números binários até noventa e nove).

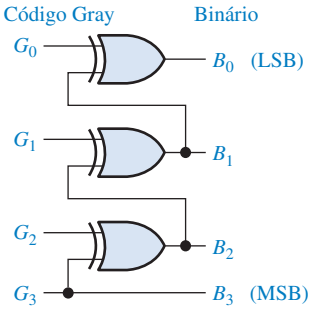
Conversão de Binário para Código Gray e Vice-Versa

O processo básico para conversões Gray-binário foi abordado no Capítulo 2. Para essas conversões podem ser usadas portas EX-OR. Podem ser usados também dispositivos de lógica programável (PLDs) para essas conversões de código. A Figura 6–43 mostra um conversor de código binário para Gray de 4 bits e a Figura 6–44 ilustra um conversor de Gray para binário de 4 bits.



▶ **FIGURA 6–43**

Lógica de conversão de binário para Gray de 4 bits. Abra o arquivo F06-43 para verificar a operação.



◀ FIGURA 6–44

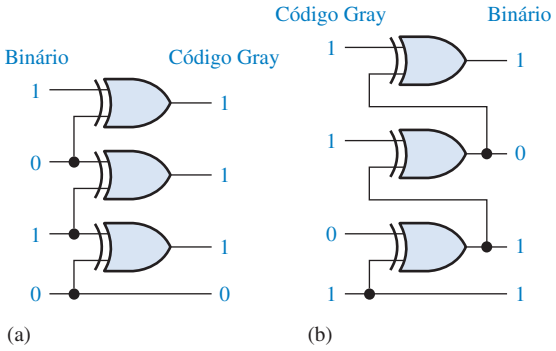
Lógica de conversão de Gray para binário de 4 bits. Abra o arquivo F06-44 para verificar a operação.

EXEMPLO 6–13

(a) Converta o número binário 0101 para código Gray usando portas EX-OR.
(b) Converta o código Gray 1011 para binário usando portas EX-OR.

Solução

(a) 0101_2 é 0111 em código Gray. Veja a Figura 6–45(a).
(b) 1011 em código Gray é 1101_2. Veja a Figura 6–45(b).



▶ FIGURA 6–45

Problema relacionado Quantas portas EX-OR são necessárias para converter um binário de 8 bits em código Gray?

SEÇÃO 6–7 REVISÃO

1. Converta o número BCD 10000101 para binário.
2. Desenhe o diagrama lógico para a conversão de um número binário de 8 bits para o código Gray.

6-8 MULTIPLEXADORES (SELETORES DE DADOS)

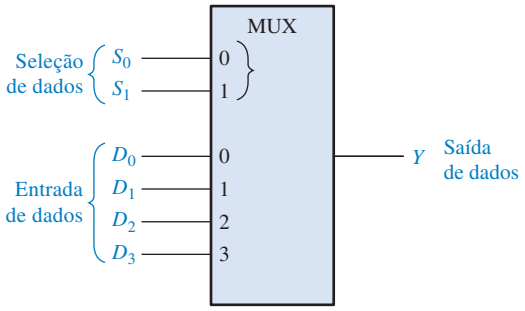
Um **multiplexador (MUX)** é um dispositivo que permite que informações digitais de diversas fontes sejam encaminhadas para uma única linha para serem transmitidas nessa linha para um destino comum. Um multiplexador básico tem várias linhas de entrada de dados e uma única linha de saída. Ele também possui entradas de seleção de dados, as quais permitem que os dados digitais de quaisquer entradas sejam comutados para a linha de saída. Os multiplexadores também são conhecidos como seletores de dados.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a operação básica de um multiplexador ■ Descrever os CIs multiplexadores 74LS151 e 74HC157 ■ Expandir um multiplexador para poder operar com mais entradas de dados ■ Usar um multiplexador como um gerador de expressões lógicas

Em um multiplexador, os dados passam das diversas linhas para uma linha.

A Figura 6–46 mostra o símbolo lógico para um multiplexador (MUX) de 4 bits. Observe que existem duas linhas de seleção de dados porque com dois bits de seleção, qualquer uma das quatro linhas de entrada de dados pode ser selecionada.



▶ **FIGURA 6–46**
Símbolo lógico para um seletor/multiplexador de dados 1 de 4.

Na Figura 6–46, um código de 2 bits nas entradas de seleção de dados (S) permitem que o dado na entrada selecionada passe para a saída de dados. Se um binário 0 ($S_1 = 0$ e $S_0 = 0$) for aplicado nas linhas de seleção de dados, o dado na entrada D_0 aparece na linha de saída de dados. Se um binário 1 ($S_1 = 0$ e $S_0 = 1$) for aplicado nas linhas de seleção de dados, o dado na entrada D_1 aparece na linha de saída de dados. Se um binário 2 ($S_1 = 1$ e $S_0 = 0$) for aplicado, o dado na entrada D_2 aparece na linha de saída de dados. Se um binário 3 ($S_1 = 1$ e $S_0 = 1$) for aplicado, o dado na entrada D_3 é comutado para a linha de saída de dados. A Tabela 6–8 mostra um resumo dessa operação.

▶ **TABELA 6–8**
Seleção de dados para um multiplexador 1 de 4

ENTRADAS DE SELEÇÃO DE DADOS S_1	S_0	ENTRADA SELECIONADA
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

NOTA: COMPUTAÇÃO

Um barramento é uma via ao longo da qual sinais elétricos são enviados de um ponto para outro no computador. Numa rede de computadores, um *barramento compartilhado* é aquele que está conectado a todos os microprocessadores do sistema com a finalidade de troca de dados. Um barramento compartilhado pode conter dispositivos de memória e de entrada/saída (in/out) os quais podem ser acessados por todos os microprocessadores do sistema. O acesso ao barramento compartilhado é controlado pelo árbitro de barramento (um multiplexador de classificações) o qual permite que apenas um microprocessador de cada vez use o barramento compartilhado do sistema.

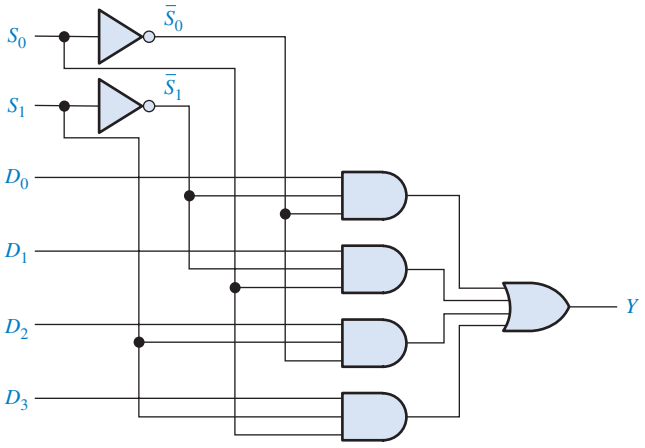
Agora vamos pensar no circuito lógico necessário para realizar essa operação de multiplexação. A saída de dados é igual ao estado da entrada selecionada. Portanto, podemos deduzir uma expressão para a saída em termos da entrada de dados e das entradas de seleção.

A saída de dados é igual a D_0 apenas se $S_1 = 0$ e $S_0 = 0$: $D_0\overline{S}_1\overline{S}_0$.
A saída de dados é igual a D_1 apenas se $S_1 = 0$ e $S_0 = 1$: $D_1\overline{S}_1S_0$.
A saída de dados é igual a D_2 apenas se $S_1 = 1$ e $S_0 = 0$: $D_2S_1\overline{S}_0$.
A saída de dados é igual a D_3 apenas se $S_1 = 1$ e $S_0 = 1$: $D_3S_1S_0$.

Quando esses termos são relacionados por uma operação OR, a expressão total para a saída de dados é

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

A implementação dessa equação requer quatro portas AND de 3 entradas, uma porta OR de 4 entradas e dois inversores para gerar o complemento de S_1 e S_0, conforme mostra a Figura 6–47. Como os dados podem ser selecionados a partir de qualquer uma das linhas de entrada, esse circuito também é denominado de **seletor de dados**.

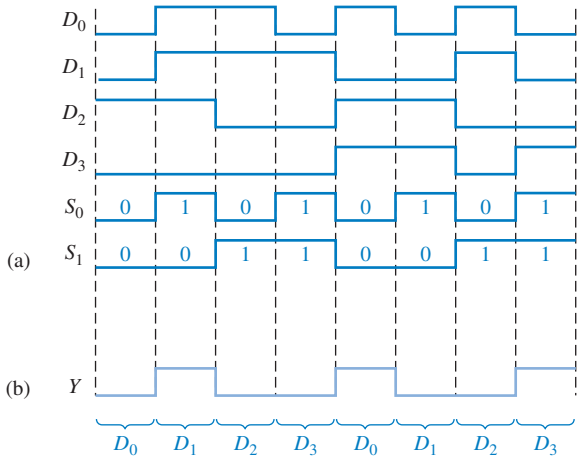


◀ **FIGURA 6–47**

Diagrama lógico para um multiplexador de 4 entradas. Abra o arquivo F06-47 para verificar a operação.

EXEMPLO 6–14

As formas de onda da entrada de dados e das entradas de seleção de dados vistas na Figura 6–48(a) são aplicadas no multiplexador mostrado na Figura 6–47. Determine a forma de onda de saída em relação às entradas.



▶ **FIGURA 6–48**

Solução Os estados binários das entradas de seleção de dados durante cada intervalo determina qual dado de entrada é selecionado. Observe que as entradas de seleção de dados passam pela seqüência binária repetitiva: 00, 01, 10, 11, 00, 01, 10, 11 e assim por diante. A forma de onda de saída resultante é mostrada na Figura 6–48(b).

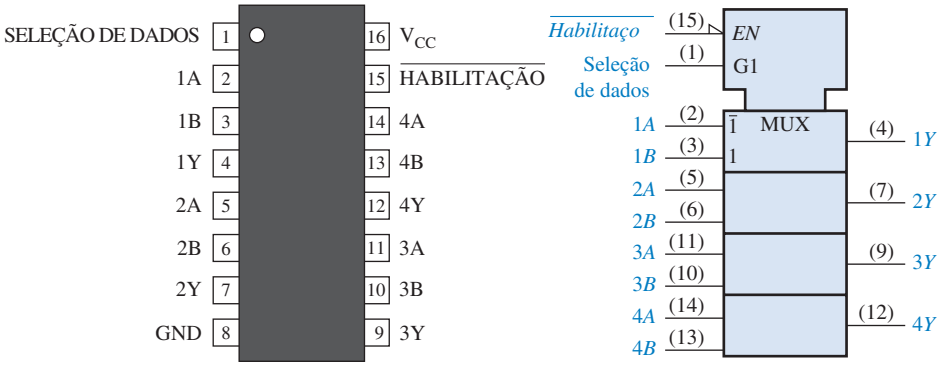
Problema relacionado Construa um diagrama de temporização mostrando todas as entradas e a saída se as formas de onda de S_0 e S_1, vistas na Figura 6–48(b), forem trocadas entre si.

UM SELETOR/MULTIPLEXADOR DE DADOS QUÁDRUPLO DE 2 ENTRADAS (74HC157)

O CI 74HC157, bem como a sua versão LS, consiste em quatro multiplexadores de 2 entradas separados. Cada um dos quatro multiplexadores compartilha a linha de seleção de dados e a entrada de habilitação (*EN*). Como existem apenas duas entradas a serem selecionadas em cada multiplexador, uma única entrada de seleção de dados é suficiente.

Um nível BAIXO na entrada de habilitação permite que o dado da entrada selecionada passe para a saída. Um nível ALTO na entrada evita a passagem do dado para a saída; ou seja, com a entrada nesse estado os multiplexadores estão desabilitados. Esse dispositivo pode ser comercializado em outras famílias CMOS e TTL. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.

O Símbolo Lógico ANSI/IEEE O diagrama de pinos para o CI 74HC157 é mostrado na Figura 6–49(a). O símbolo lógico ANSI/IEEE para o CI 74HC157 é mostrado na Figura 6–49(b). Observe que os quatro multiplexadores são indicados por um contorno retangular e que as entradas comuns aos quatro multiplexadores são indicadas como entradas de um bloco com entalhes na parte superior, o qual é denominado de *bloco de controle comum*. Todas as identificações dentro do bloco MUX superior se aplicam aos outros blocos abaixo dele.

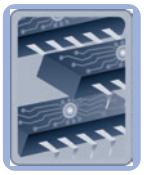


(a) Diagrama de pinos (b) Símbolo lógico

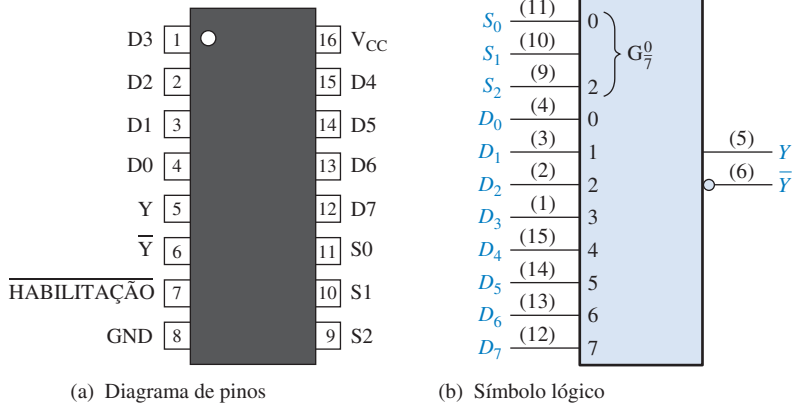
FIGURA 6–49
Diagrama de pinos e símbolo lógico para o CI 74HC157 (quatro seletores/multiplexadores de dados de 2 entradas).

Observe as identificações 1 e $\overline{1}$ nos blocos MUX e a identificação G1 no bloco de controle comum. Essas identificações são um exemplo do sistema de **notação de dependência** especificado no padrão 91-1984 da ANSI/IEEE. Nesse caso, G1 indica uma relação AND entre a entrada de seleção de dados e as entradas de dados com indicações 1 ou $\overline{1}$. (O $\overline{1}$ significa que a relação AND se aplica ao complemento da entrada G1). Em outras palavras, quando a entrada de seleção de dados for nível ALTO, as entradas *B* dos multiplexadores são selecionadas; e quando a entrada de seleção de dados for nível BAIXO, as entradas *A* são selecionadas. Um "G" sempre é usado para indicar uma dependência AND. Outros aspectos da notação de dependência são apresentados em momentos apropriados ao longo desse livro.

UM SELETOR/MULTIPLEXADOR DE DADOS DE 8 ENTRADAS (74LS151)



O CI 74LS151 tem oito entradas de dados (D_0–D_7) e, portanto, três entradas de seleção de dados, ou endereço, (S_0–S_2). Três bits são necessários para selecionar qualquer uma das oito entradas de dados ($2^3 = 8$). Um nível BAIXO na entrada de $\overline{habilitação}$ permite que a entrada de dados selecionada passe para a saída. Observe que a saída de dados e o seu complemento estão disponíveis. A Figura 6–50(a) mostra o diagrama de pinos e a parte (b) mostra o símbolo lógico. Nesse caso não existe um bloco de controle comum porque existe apenas um multiplexador a ser controlado, e não quatro como no CI 74HC157. A indicação $G\frac{0}{7}$ dentro do símbolo lógico representa a relação AND entre as entradas de seleção de dados e cada uma das entradas de dados (de 0 a 7). Esse dispositivo pode ser comercializado em outras famílias TTL e CMOS. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha este livro.



(a) Diagrama de pinos (b) Símbolo lógico

◄ FIGURA 6–50

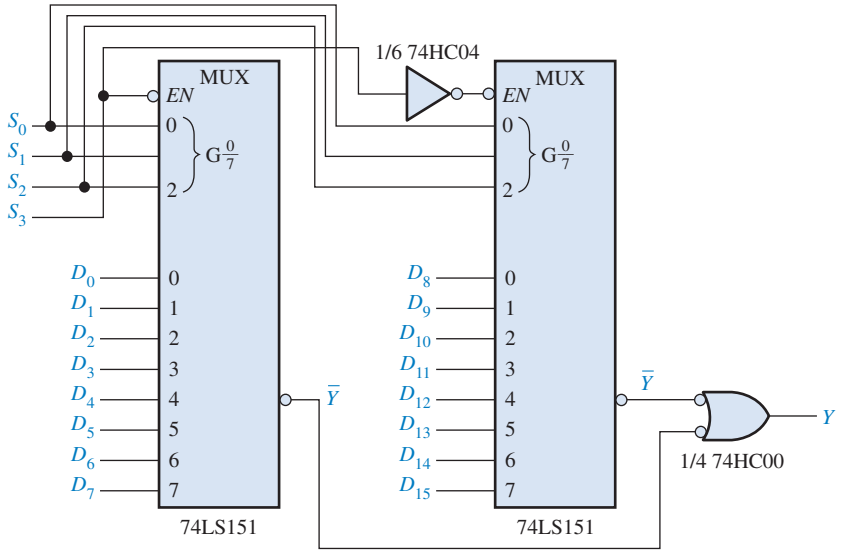
Diagrama de pinos e símbolo lógico para o CI 74LS151 (seletor/multiplexador de dados de 8 entradas).

EXEMPLO 6–15

Use CIs 74LS151 e qualquer outra lógica necessária para multiplexar 16 linhas de dados em uma única linha de dados de saída.

Solução

A Figura 6–51 mostra uma implementação desse sistema. São necessários quatro bits para selecionar uma das 16 entradas de dados ($2^4 = 16$). Nessa aplicação a entrada de $\overline{habilitação}$ é usada como o bit de seleção de dados mais significativo. Quando o MSB no código de seleção de dados for nível BAIXO, o CI 74LS151 à esquerda será habilitado, sendo que um dos dados de entrada (D_0 a D_7) será selecionado pelos outros três bits de seleção de dados. Quando o MSB da seleção de dados for nível ALTO, o CI 74LS151 à direita será habilitado, sendo que uma das entradas de dados (D_8 a D_{15}) será selecionada. O dado da entrada selecionada passa então pela porta OR negativa saindo pela única linha de saída.



► FIGURA 6–51

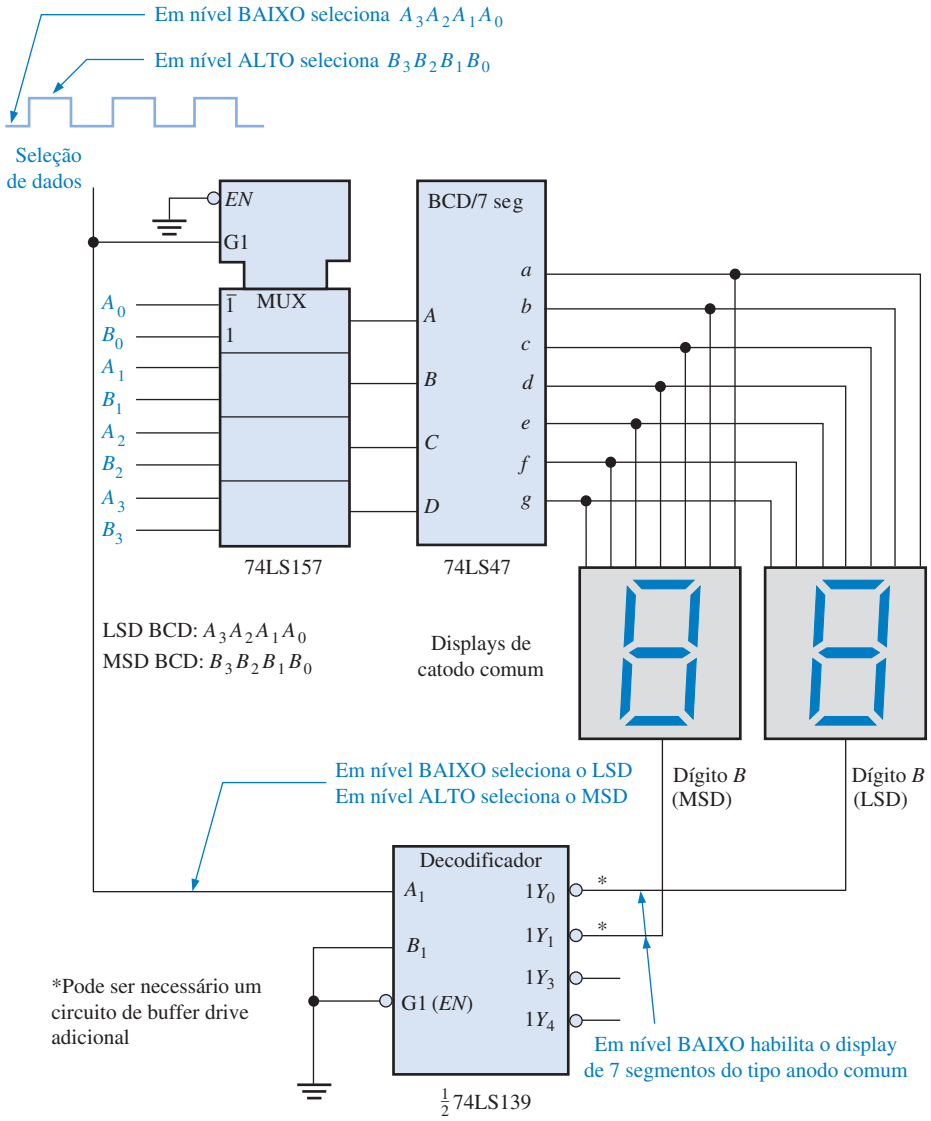
Um multiplexador de 16 entradas.

Problema relacionado

Determine os códigos nas entradas de seleção necessários para selecionar cada uma das seguintes entradas de dados: D_0, D_4, D_8 e D_{13}.

Aplicações

Multiplexador para Display de 7 Segmentos A Figura 6–52 mostra um método simplificado de multiplexação de números BCD para displays de 7 segmentos. Nesse exemplo são mostrados números de 2 dígitos em displays de 7 segmentos usando um único decodificador de BCD para 7 segmentos. Esse método básico de multiplexação de displays pode ser estendido para sistemas com qualquer número de dígitos.



▶ **FIGURA 6–52**
Lógica simplificada de multiplexação de displays de 7 segmentos.

A operação básica é descrita logo a seguir. Os dois dígitos BCD ($A_3A_2A_1A_0$ e $B_3B_2B_1B_0$) são aplicados nas entradas do multiplexador. Uma onda quadrada é aplicada na linha de seleção de dados, sendo que quando essa linha for nível BAIXO, os bits A ($A_3A_2A_1A_0$) passam para as entradas do decodificador de BCD para 7 segmentos (74LS47). Um nível BAIXO na entrada de seleção de dados é aplicado também na entrada A_1 do decodificador de 2 linhas para 4 linhas (74LS139), ativando então a saída 0 deste CI e habilitando o display do dígito *A* conectando efetivamente o seu terminal comum em GND. Desta forma, o dígito *A* estará *ligado* (*on*) e o dígito *B* *desligado* (*off*).

Quando a linha de seleção de dados for para nível ALTO, os bits B ($B_3B_2B_1B_0$) passam para as entradas do decodificador de BCD para 7 segmentos. Além disso, a saída 1 do CI decodificador 74LS139 é ativada, habilitando então o display do dígito B. Agora o dígito B estará *ligado* (*on*) e o dígito A *desligado* (*off*). O ciclo se repete na freqüência da onda quadrada na entrada de seleção de dados. Essa freqüência tem que ser alta o suficiente (cerca de 30 Hz) para evitar cintilações (*flickers*) conforme os displays dos dígitos são multiplexados.

Gerador de Funções Lógicas Uma aplicação útil para um seletor/multiplexador de dados é na geração de funções lógicas combinacionais na forma de soma-de-produtos. Quando dessa forma, o dispositivo pode substituir portas discretas, podendo freqüentemente diminuir bastante o número de CIs, e pode tornar muito fáceis as alterações de projetos.

Para ilustrar, podemos usar um seletor/multiplexador de dados de 8 entradas (74LS151) para implementar qualquer função lógica especificada de 3 variáveis sendo as variáveis conectadas nas entradas de seleção de dados e cada entrada de dado submetida ao nível lógico necessário de acordo com a tabela-verdade para a função em questão. Por exemplo, se a função for nível 1 quando a combinação de variáveis for $\overline{A}_2A_1\overline{A}_0$, a entrada 2 (selecionada por 010) deverá estar conectada ao nível ALTO. Esse nível ALTO passará para a saída quando essa combinação particular de variáveis ocorrer nas linhas de seleção de dados. Um exemplo ajudará a esclarecer essa aplicação.

EXEMPLO 6–16

Implemente a função lógica especificada na Tabela 6–9 usando um CI 74LS151 (seletor/multiplexador de dados de 8 entradas). Compare esse método com uma implementação que usa portas lógicas discretas.

▼ **TABELA 6–9**

ENTRADAS			SAÍDA
A_2	A_1	A_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Solução Observe a partir da tabela-verdade que Y é nível 1 para as seguintes combinações das variáveis de entrada: 001, 011, 101 e 110. Para todas as outras combinações, Y é nível 0. Para essa função ser implementada com o seletor de dados definido, a entrada de dados selecionada para cada uma das combinações apresentadas tem que ser conectada ao nível ALTO (5 V). Todas as outras entradas de dados têm que ser conectadas ao nível BAIXO (GND), conforme mostra a Figura 6–53.

A implementação dessa função com portas lógicas necessitaria de quatro portas AND de 3 entradas, uma porta OR de 4 entradas e três inversores, a menos que essa expressão possa ser simplificada.

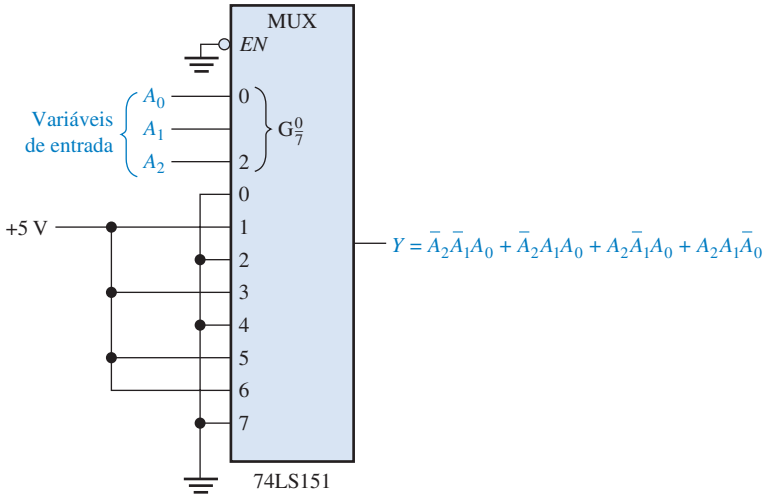


FIGURA 6–53

Seletor/multiplexador de dados conectado como um gerador de funções lógicas de 3 variáveis.

Problema relacionado Use o CI 74LS151 para implentar a seguinte expressão:

$$Y = \overline{A}_2\overline{A}_1\overline{A}_0 + A_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1\overline{A}_0$$

O Exemplo 6–16 ilustra como um seletor de dados de 8 entradas pode ser usado como um gerador de função lógica para três variáveis. Na realidade, esse dispositivo também pode ser usado como um gerador de função lógica de 4 variáveis fazendo uso de um dos bits (A_0) em conjunto com as entradas de dados.

Uma tabela-verdade de 4 variáveis tem 16 combinações das variáveis de entrada. Quando um seletor de dados de 8 bits é usado, cada entrada é selecionada duas vezes: a primeira vez quando A_0 é nível 0 e a segunda vez quando A_0 é nível 1. Com isso em mente, as regras a seguir podem ser aplicadas (Y é a saída e A_0 é o bit menos significativo):

1. Se $Y = 0$ nas duas vezes em que uma dada entrada for selecionada por uma certa combinação de variáveis de entrada, $A_3A_2A_1$, conecte essa entrada de dados em GND (0).
2. Se $Y = 1$ nas duas vezes em que uma dada entrada for selecionada por uma certa combinação de variáveis de entrada, $A_3A_2A_1$, conecte essa entrada de dados em $+V$ (1).
3. Se Y for diferente nas duas vezes em que uma dada entrada de dados for selecionada por uma certa combinação de variáveis de entrada, $A_3A_2A_1$, e se $Y = A_0$, conecte essa entrada de dados em A_0.
4. Se Y for diferente nas duas vezes em que uma dada entrada de dados for selecionada por uma certa combinação de variáveis de entrada, $A_3A_2A_1$, e se $Y = A_0$, conecte essa entrada de dados em $\overline{A}_0$.

EXEMPLO 6–17

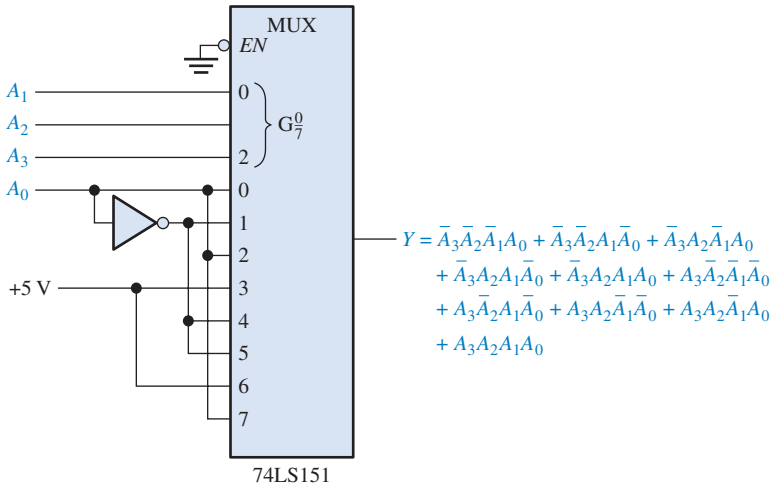
Implemente a função lógica dada pela Tabela 6–10 usando um CI 74LS151 (seletor/multiplexador de dados de 8 entradas). Compare esse método com uma implementação que usa portas lógicas discretas.

Solução As entradas de seleção de dados são $A_3A_2A_1$. Na primeira linha da tabela, $A_3A_2A_1 = 000$ e $Y = A_0$. Na segunda linha, onde $A_3A_2A_1$ é novamente 000, $Y = \overline{A}_0$. Portanto, A_0 é conectada na entrada 0. Na terceira linha da tabela, $A_3A_2A_1 = 001$ e $Y = A_0$. Além disso, na quarta linha, quando $A_3A_2A_1$ também é 001, $Y = \overline{A}_0$. Portanto, $\overline{A}_0$ é invertida e conectada na entra-

▶ **TABELA 6–10**

DÍGITO DECIMAL	ENTRADAS A_3	A_2	A_1	A_0	SAÍDA Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

da 1. Essa análise prossegue até que cada entrada seja conectada adequadamente de acordo com as regras especificadas. A implementação é mostrada na Figura 6–54.



▶ **FIGURA 6–54**
Seletor/multiplexador de dados conectado como um gerador de função lógica de 4 variáveis.

Caso fosse implementado com portas lógicas, a função necessitaria de dez portas AND de 4 entradas, uma porta OR de 10 entradas e quatro inversores, embora uma possível simplificação reduzisse essas especificações.

Problema relacionado Na Tabela 6–10, se $Y = 0$ quando as entradas estão todas em zero e alternadamente for 1 e 0 para as linhas restantes da tabela, use um CI 74LS151 para implementar a função lógica resultante.

SEÇÃO 6–8 REVISÃO

1. Na Figura 6–47, $D_0 = 0$, $D_1 = 0$, $D_2 = 1$, $D_3 = 0$, $S_0 = 1$ e $S_1 = 0$. Qual é o nível lógico da saída?
2. Identifique cada dispositivo a seguir.
   (a) 74LS157 **(b)** 74LS151
3. Um CI 74LS151 tem alternadamente níveis BAIXO e ALTO em suas entradas de dados começando por D_0. As linhas de seleção de dados são recebem uma seqüência de contagem binária (000, 001, 010 e assim por diante) numa freqüência de 1 kHz. A entrada de habilitação é nível BAIXO. Descreva a forma de onda na saída de dados.
4. Descreva resumidamente a finalidade de cada um dos seguintes dispositivos vistos na Figura 6–52.
   (a) 74LS157 **(b)** 74LS47 **(c)** 74LS139

6-9 DEMULTIPLEXADORES

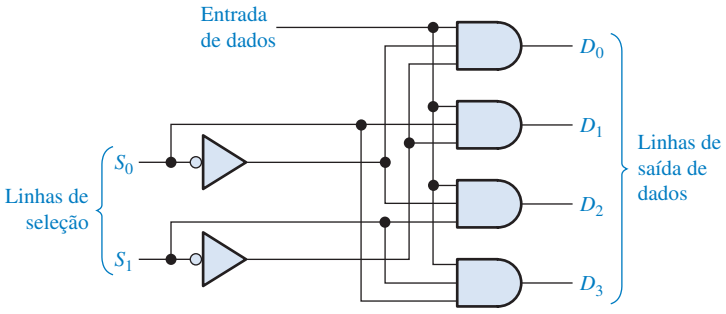
Um **demultiplexador (DEMUX)** basicamente inverte a função da multiplexação. Ele recebe informações digitais a partir de uma linha e as distribui para um determinado número de linhas de saída. Por essa razão, o demultiplexador também é conhecido como distribuidor de dados. Conforme estudaremos, os decodificadores também podem ser usados como demultiplexadores.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a operação básica de um demultiplexador ■ Descrever como o CI decodificador de 4 para 16 linhas 74HC154 pode ser usado como um demultiplexador ■ Desenvolver o diagrama de temporização para um demultiplexador com dados e entradas de seleção de dados especificadas

Em um demultiplexador, os dados são transferidos de uma linha para diversas linhas.

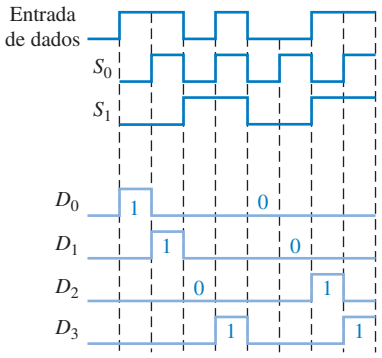
A Figura 6–55 mostra o circuito de um demultiplexador (DEMUX) de 1 linha para 4 linhas. A linha de entrada de dados está conectada em todas as portas AND. As duas linhas de seleção de dados habilitam uma porta de cada vez, e os dados que aparecem na linha de entrada de dados passam, através da porta selecionada, para a linha de saída de dados associada.



▶ **FIGURA 6–55**
Demultiplexador de 1 linha para 4 linhas.

EXEMPLO 6–18

A forma de onda de entrada de dados em série e as entradas de seleção de dados (S_0 e S_1) são mostradas na Figura 6–56. Determine as formas de onda da saída de dados D_0 a D_3 para o demultiplexador visto na Figura 6–55.



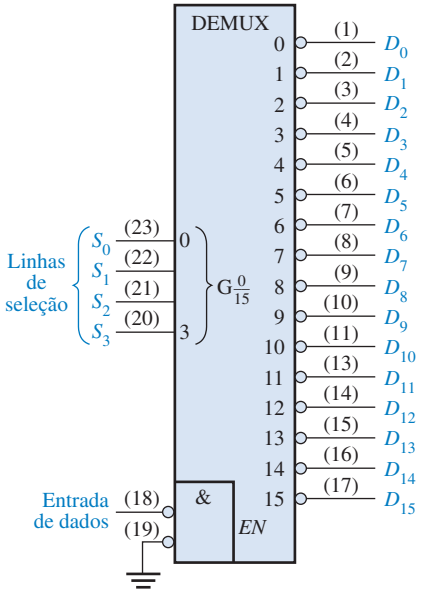
▶ **FIGURA 6–56**

Solução Observe que as linhas de seleção seguem uma seqüência binária de forma que cada bit sucessivo de entrada é direcionado para D_0, D_1, D_2 e D_3 na seqüência, conforme mostra as formas de onda vistas na Figura 6–56.

Problema relacionado Desenvolva o diagrama de temporização para o demultiplexador se as formas de onda de S_0 e S_1 forem invertidas.

O CI DEMULTIPLEXADOR 74HC154

Já discutimos o CI decodificador 74HC154 sendo usado numa aplicação como um decodificador de 4 linhas para 16 linhas (Seção 6–5). Esse dispositivo e outros decodificadores também podem ser usados em aplicações de demultiplexação. O símbolo lógico para esse dispositivo quando usado como um demultiplexador é mostrado na Figura 6–57. Em aplicações como demultiplexador, as linhas de entrada são usadas como linhas de dados. Uma das entradas de seleção de chip é usada como linha de entrada de dados, enquanto a outra entrada de seleção de chip é mantida em nível BAIXO para habilitar a porta AND negativa interna na parte inferior do diagrama. Esse dispositivo pode ser comercializado em outras famílias CMOS ou TTL. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha esse livro.



◀ **FIGURA 6–57**

O CI decodificador 74HC154 usado como um demultiplexador.

SEÇÃO 6–9 REVISÃO

1. Em geral, como um decodificador pode ser usado como um demultiplexador?
2. O CI demultiplexador 74HC154 mostrado na Figura 6–57 tem um código binário de 1010 nas linhas de seleção de dados e a linha de entrada de dados é nível BAIXO. Quais são os estados das linhas de saída?

6-10 GERADORES/VERIFICADORES DE PARIDADE

À medida que códigos digitais são transferidos de um ponto para outro dentro de um sistema digital ou enquanto códigos são transmitidos de um sistema para outro, podem ocorrer erros. Os erros nessas transferências constituem alterações indesejadas nos bits que constituem a informação codificada; ou seja, um nível 1 pode mudar para 0, ou um 0 mudar para 1, devido ao mau funcionamento do componente ou em função de ruído elétrico. Na maioria dos sistemas digitais a probabilidade de ocorrer erro num único bit é muito pequena, e a probabilidade de ocorrer erro em mais de um bit é ainda menor. Contudo, quando ocorre um erro não detectado, ele pode causar sérios problemas num sistema digital.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar o conceito de paridade ■ Implementar um circuito de paridade básico com portas EX-OR ■ Descrever a operação básica da geração e da verificação de paridade ■ Discutir a operação do CI gerador/verificador de paridade de 9 bits 74LS280 ■ Discutir como a detecção de erro pode ser implementada numa transmissão de dados

O método da paridade de detecção de erro no qual um **bit de paridade** é anexado ao grupo de bits de informação para tornar o número total de 1s par ou ímpar (depende do sistema) foi abordada no Capítulo 2. Em adição aos bits de paridade, vários códigos específicos também proporcionam detecção de erro inerente.

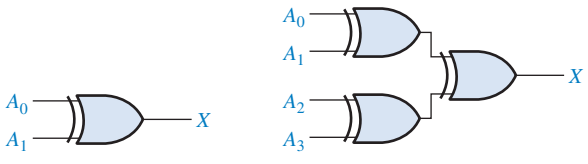
Lógica Básica de Paridade

Um bit de paridade indica se o número de 1s em um código é par ou ímpar com a finalidade de detecção de erro.

Para verificar ou gerar o bit de paridade adequado num dado código, pode ser usado um princípio básico:

A soma (desconsiderando os carries) de um número par de 1s é sempre 0, e a soma de um número ímpar de 1s é sempre 1.

Portanto, para determinar se um dado código tem **paridade par** ou **paridade ímpar**, todos os bits no código são somados. Como sabemos, a soma de dois bits pode ser gerada por uma porta EX-OR, como mostra a Figura 6–58(a); a soma de quatro bits pode ser feita por três portas EX-OR conectadas como mostra a Figura 6–58(b); e assim por diante. Quando o número de 1s na entrada for par, a saída *X* será 0 (nível BAIXO). Quando o número de 1s for ímpar, a saída *X* será 1 (nível ALTO).

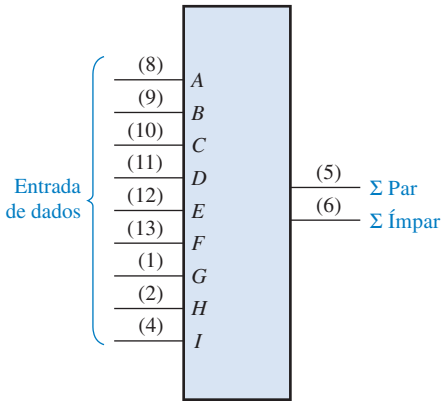


(a) Soma de dois bits (b) Soma de quatro bits

▶ **FIGURA 6–58**

GERADOR/VERIFICADOR DE PARIDADE DE 9 BITS (74LS280)

A Figura 6–59 mostra o símbolo lógico e a tabela de funções para o CI 74LS280. Esse dispositivo em particular pode ser usado para verificar paridade par ou ímpar num código de 9 bits (oito bits de dados e um bit de paridade) ou ainda pode ser usado para gerar um bit de paridade para um código binário de até nove bits. As entradas são identificadas de *A* a *I*; quando existe um número par de 1s nas entradas, a saída ∑ Par é nível ALTO e a saída ∑ Ímpar é nível BAIXO. Esse dispositivo pode ser comercializado em outras famílias TTL e CMOS. Verifique o site da Texas Instruments (www.ti.com) ou o CD-ROM da Texas Instruments que acompanha este livro.



(a) Símbolo lógico tradicional

Número de entradas de A a I que são nível ALTO	Saídas	
	Σ Par	Σ Ímpar
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

(b) Tabela de funções

▲ **FIGURA 6–59**

O CI gerador/verificador de paridade de 9 bits 74LS280.

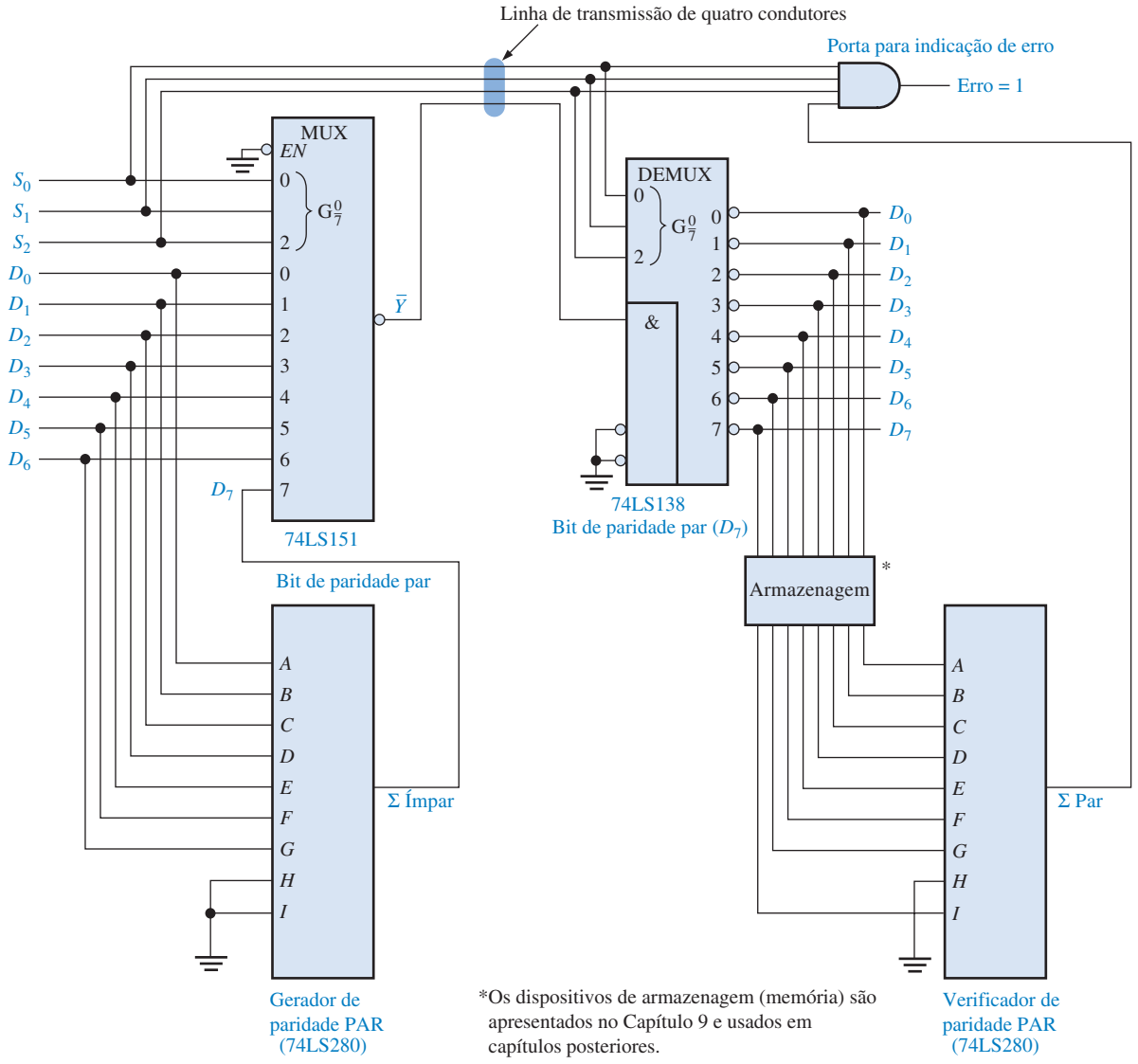
Verificador de Paridade Quando esse dispositivo é usado como um verificador de paridade par, o número de bits de entrada deve ser sempre par; e quando ocorrer um erro de paridade, a saída ∑ Par vai para nível BAIXO e a saída ∑ Ímpar vai para nível ALTO. Quando ele é usado como verificador de paridade ímpar, o número de bits de entrada tem que ser sempre ímpar; e quando um erro de paridade ocorrer, a saída ∑ Ímpar vai para nível BAIXO e a saída ∑ Par vai para nível ALTO.

Gerador de Paridade Se esse dispositivo é usado como um gerador de paridade par, o bit de paridade é obtido da saída ∑ Ímpar porque essa saída é nível 0 se houver um número par de bits de entrada em nível 1 e será nível 1 se houver um número ímpar de 1s. Quando usado como um gerador de paridade ímpar, o bit de paridade é obtido da saída ∑ Par porque ele é nível 0 quando o número de bits 1s de entrada for ímpar.

Sistema de Transmissão de Dados com Detecção de Erros

Um sistema de transmissão de dados simplificado é mostrado na Figura 6–60 para ilustrar uma aplicação de geradores/verificadores de paridade, bem como multiplexadores e demultiplexadores, e para ilustrar a necessidade do armazenamento de dados em algumas aplicações.

Nessa aplicação, os dados digitais provenientes de sete fontes são multiplexados numa única linha para serem transmitidos para um ponto distante. Os sete bits de dados (D_0 a D_6) são aplicados nas entradas de dados do multiplexador e, ao mesmo tempo, nas entradas dos gerador de paridade par. A saída ∑ Ímpar do gerador de paridade é usada como bit de paridade par. Esse bit será 0 se o número de 1s nas entradas de *A* a *I* for par e será 1 se o número de 1s nas entradas de *A* a *I* for ímpar. Esse é o bit D_7 do código transmitido.



▲ **FIGURA 6–60**

Sistema de transmissão de dados simplificado com detecção de erro.

NOTA: COMPUTAÇÃO

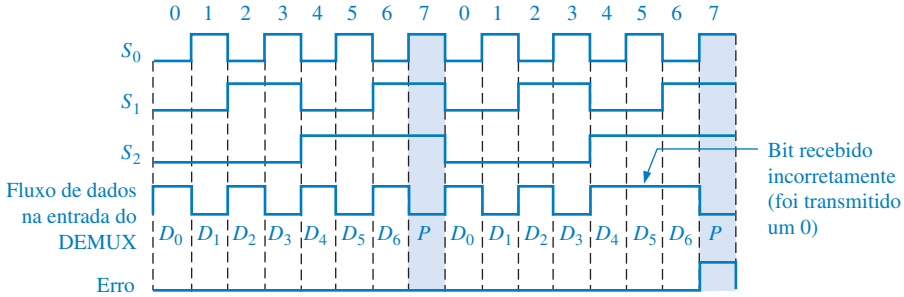
O microprocessador Pentium realiza verificações internas de paridade, bem como verifica a paridade das informações externas nos barramentos de dados e endereço. Numa operação de leitura, o sistema externo pode transferir a informação de paridade juntamente com os bytes de dados. O Pentium verifica se a paridade resultante é par e envia o sinal correspondente. Quando envia um código de endereço, o Pentium não faz a verificação de paridade do endereço, mas gera um bit de paridade par para o endereço.

As entradas de seleção de dados recebem uma seqüência binária repetida ciclicamente, e cada bit de dado, começando com D_0, passa de forma serial para a linha de transmissão $(\overline{Y})$. Nesse exemplo, a linha de transmissão consiste de quatro condutores: um transporta os dados em série e três transportam os sinais de temporização (seleção de dados). Existem formas mais sofisticadas de enviar a informação de temporização, mas estamos usando esse método direto para ilustrar o princípio básico.

Os sinais de seleção de dados e os dados seriais são aplicados no demultiplexador no final do sistema. Os bits de dados são distribuídos pelo demultiplexador nas linhas de saída na ordem em que estavam nas entradas do multiplexador. Ou seja, D_0 vai para a saída D_0, D_1 vai para a saída D_1, e assim por diante. O bit de paridade vai para a saída D_7. Esses oito bits são temporariamente armazenados e aplicados no verificador de paridade par. Nem todos os bits estarão presentes nas entradas do verificador de paridade até que o bit de paridade D_7 saia e seja armazenado. Nesse momento, a porta de erro é habilitada pelo código de seleção de dados 111. Se a paridade estivar correta, um nível 0 aparece na saída Σ Par mantendo a saída Erro em nível 0. Se a paridade não estiver correta, todos os 1s aparecem nas entradas da porta de erro, resultando num nível 1 na saída Erro.

Essa aplicação particular demonstrou a necessidade do armazenamento de dados de forma que o leitor estará mais bem preparado para compreender a utilidade dos dispositivos de armazenamento que serão apresentados no Capítulo 7 e usados em capítulos posteriores.

O diagrama de temporização visto na Figura 6–61 ilustra um caso específico no qual duas palavras de 8 bits são transmitidas, uma com a paridade correta e a outra com um erro.



◀ FIGURA 6–61

Exemplo de uma transmissão de dados com e sem erro para o sistema mostrado na Figura 6–60.

SEÇÃO 6–10 REVISÃO

1. Acrescente um bit de paridade par em cada um dos seguintes códigos:
(a) 110100 **(b)** 01100011 **(c)** 500

2. Acrescente um bit de paridade ímpar em cada um dos seguintes códigos:
(a) 1010101 **(b)** 1000001

3. Verifique cada um dos códigos de paridade par para saber se existe erro.
(a) 100010101 **(b)** 1110111001

6-11 ANÁLISE DE DEFEITO

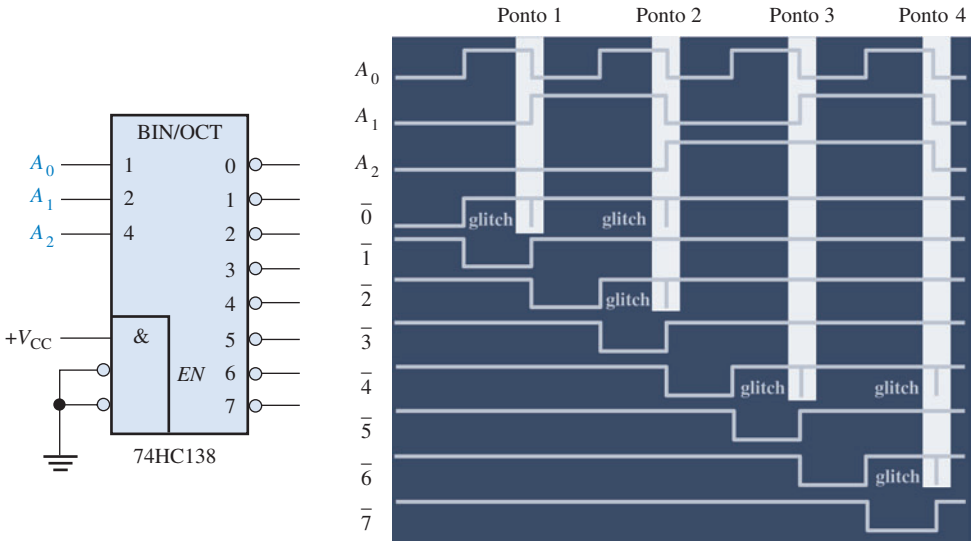
Nesta seção os problemas dos glitches de decodificador são apresentados e examinados do ponto de vista da análise de defeito. Um **glitch** é qualquer spike (pulso) de tensão ou corrente de duração muito curta. Um glitch pode ser interpretado como um sinal válido por um circuito lógico podendo provocar uma operação inadequada.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar o que é um glitch ■ Determinar a causa de glitches na aplicação de um decodificador ■ Usar o método de habilitação (*strobe*) de saída para eliminar glitches

O CI 74LS138 foi usado como um DEMUX no sistema de transmissão de dados visto na Figura 6–60. Agora o CI 74HC138 é usado como um decodificador de 3 linhas para 8 linhas (de binário para octal), mostrado na Figura 6–62, para ilustrar como glitches ocorrem e como identificar a causa deles. As entradas $A_2A_1A_0$ do decodificador recebem uma seqüência binária de um contador e as formas de onda resultantes das entradas e saída podem ser mostradas na tela de um analisador lógico, como mostra a Figura 6–62. As transições de A_2 estão atrasadas em relação às transições de A_1 e as transições de A_1 estão atrasadas em relação às transições de A_0. Isso normalmente ocorre quando formas de onda são geradas por um contador binário, como aprenderemos no Capítulo 8.

As formas de onda de saída estão corretas exceto pelos glitches que ocorrem em alguns dos sinais de saída. Um analisador lógico ou um osciloscópio pode ser usado para mostrar glitches, os quais normalmente são difíceis de visualizar. Geralmente, o analisador lógico é preferido, especialmente para baixas taxas de repetição (menores que 10 Hz) e/ou ocorrências irregulares porque a maioria dos analisadores lógicos tem uma capacidade de *captura de glitch*. Os osciloscópios po-

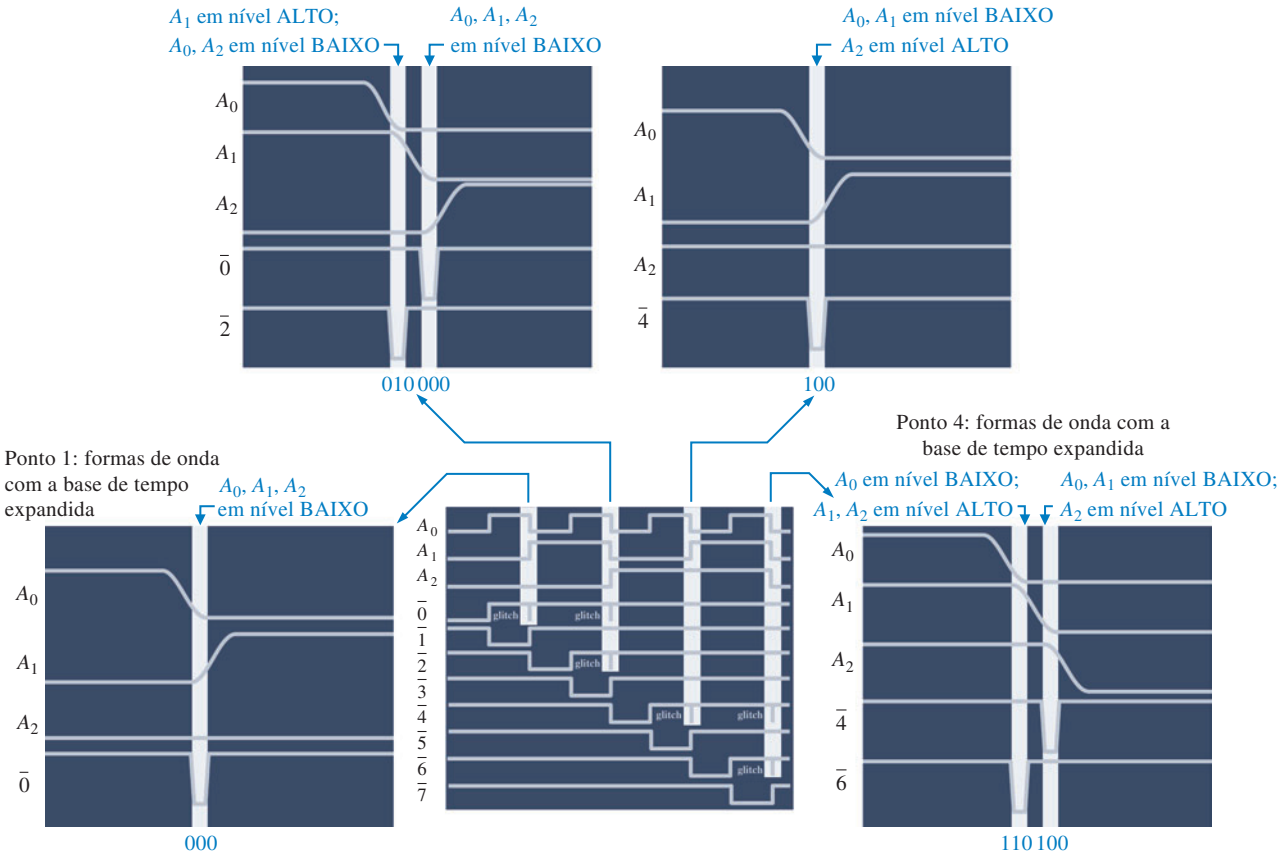


▶ **FIGURA 6–62**

Formas de onda de um decodificador com glitches de saída.

dem ser usados para observar glitches com um sucesso razoável particularmente se os glitches ocorrerem em intervalos regulares e numa taxa de repetição alta (maior que 10 kHz).

Os pontos de interesse indicados pelas áreas destacadas nas formas de onda de entrada mostradas na Figura 6–62 são explorados na Figura 6–63. No ponto 1 existe um estado de transição de

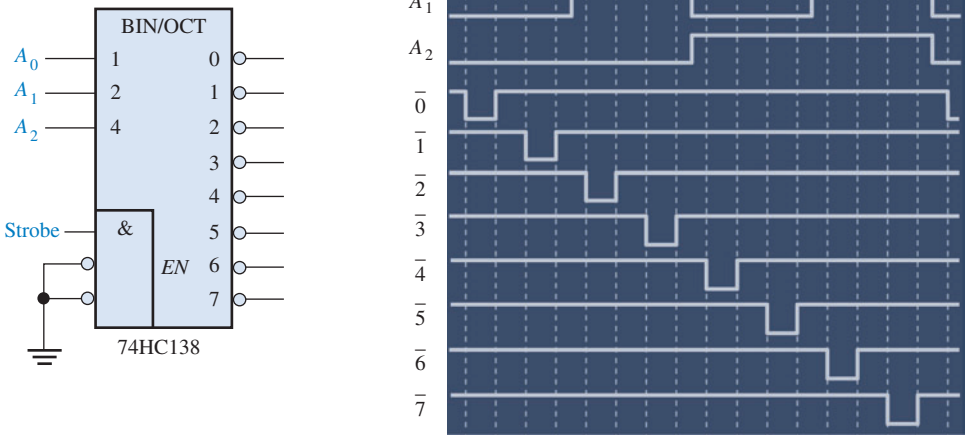


▲ **FIGURA 6–63**

Forma de onda de decodificador mostrando como as transições dos estados de entrada produzem glitches nas formas de onda de saída.

000 devido às diferenças de atraso nas formas de onda. Isso provoca o primeiro glitch na saída $\overline{0}$ do decodificador. No ponto 2 existem dois estados de transição, 010 e 000. Isso provoca o glitch na saída $\overline{2}$ do decodificador e o segundo glitch na saída $\overline{0}$, respectivamente. No ponto 3, o estado de transição é 100, o qual provoca o primeiro glitch na saída $\overline{4}$ do decodificador. No ponto 4, os dois estados de transição, 110 e 100, resultam em um primeiro glitch na saída $\overline{6}$ e um segundo glitch na saída $\overline{4}$, respectivamente.

Uma forma de eliminar o problema de glitch é um método denominado de **strobing** (habilitação), no qual o decodificador é habilitado por um pulso de strobe apenas durante os momentos em que as formas de onda não estão em transição. Esse método é ilustrado na Figura 6–64.



◀ **FIGURA 6–64**
Aplicação de uma forma de onda strobe para eliminar glitches nas saídas do decodificador.

SEÇÃO 6–11 REVISÃO

1. Defina o termo *glitch*.
2. Explique a causa básica de glitches na lógica de um decodificador.
3. Defina o termo *strobe*.

Os problemas de análise de defeito existentes no CD-ROM estão disponíveis na Seção "Práticas de Análise de Defeito Usando o Multisim" no final dos problemas do capítulo.

DICA PRÁTICA

Além dos glitches que são resultado dos atrasos de propagação, assim como vimos no caso do decodificador, outros tipos de spikes de ruído indesejado também podem ser um problema. Spikes de tensão e corrente nas linhas de V_{CC} e GND são provocados por formas de onda de comutação rápida em circuitos digitais. Esse problema pode ser minimizado através de um leiaute adequado na placa de circuito impresso. Spikes de comutação podem ser absorvidos desacoplando a placa de circuito com um capacitor de 1 μF de V_{CC} para GND. Além disso, capacitores de desacoplamento menores (0,022μF a 0,1 μF) devem ser distribuídos nos diversos pontos entre V_{CC} e GND ao longo da placa de circuito. O desacoplamento deve ser feito em especial próximo dos dispositivos que estão comutando em altas freqüências e acionando mais cargas como osciladores, contadores, buffers e drivers de barramento.

APLICAÇÕES EM SISTEMAS DIGITAIS

Nesta seção de aplicações em sistemas digitais, começamos a trabalhar com um sistema de controle de semáforo de trânsito. Estabelecemos aqui os requisitos do sistema, desenvolvemos um diagrama em bloco e criamos um diagrama de estados para definir a seqüência de operação. Faremos o projeto da parte do sistema que envolve lógica combinacional e consideraremos os métodos de teste. A temporização e as partes seqüenciais do sistema serão tratadas nos Capítulos 7 e 8.

Requisitos Gerais de Sistema

Um controlador digital é necessário para controlar um semáforo de trânsito na interseção de uma via principal e uma via secundária. A via principal terá um sinal verde de pelo menos 25 s ou continuará verde enquanto não houver veículos na via secundária. A via secundária terá um sinal verde enquanto não existir veículos na via principal ou por um máximo de 25 s. Teremos um sinal amarelo de atenção durante 4 s entre a mudança do verde para o vermelho nas vias principal e secundária. Esses requisitos estão ilustrados no diagrama ilustrado na Figura 6–65.

Desenvolvimento de um Diagrama em Bloco do Sistema

A partir dos requisitos podemos desenvolver um diagrama em bloco do sistema. Primeiro, sabemos que o sistema tem que controlar seis diferentes pares de luz. Essas luzes são vermelho, amarelo e verde para as duas direções da via principal e vermelho, amarelo e verde nas duas direções da via secundária. Além disso, sabemos que existe uma entrada externa (além da alimentação) a partir do sensor de veículos na via secundária. A Figura 6–66 mostra um diagrama em bloco mínimo com esses requisitos.

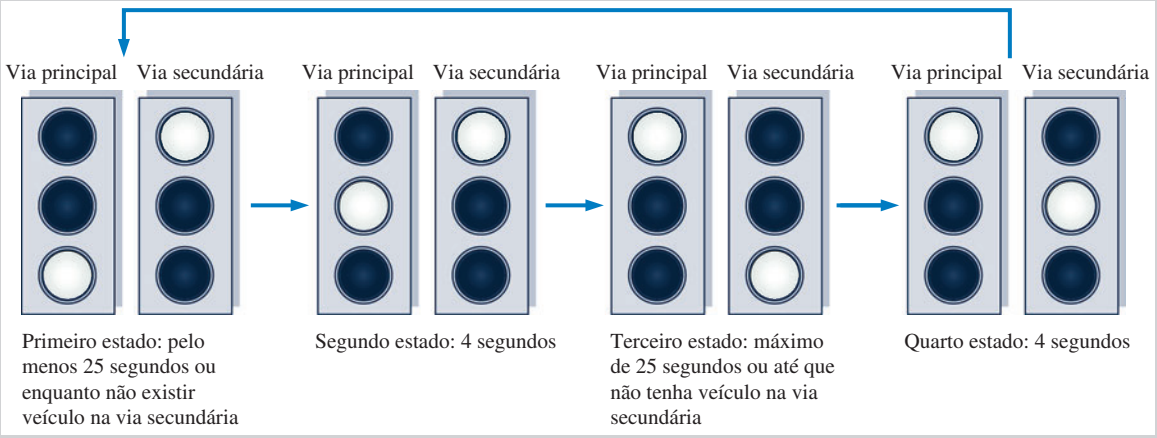
Usando o diagrama em bloco mínimo do sistema, podemos começar a detalhá-lo. O sistema tem quatro estados, conforme indicado na Figura 6–65, assim é necessário um circuito lógico para controlar a seqüência de estados (lógica seqüencial). Além disso, são necessários circuitos para gerar os intervalos de tempo adequados de 25 s e 4 s que são necessários no sistema e para gerar um sinal de clock para a operação cíclica do sistema (circuitos de temporização). Os intervalos de tempo (longo e curto) e o sensor de veículo são entradas para a lógica seqüencial porque o seqüenciamento dos estados é uma função dessas variáveis. Os circuitos lógicos também são necessários para determinar em qual dos quatro estados o sistema está em qualquer momento especificado, para gerar as saídas adequadas para as luzes (decodificador de estados e circuito lógico de acionamento das luzes) e para iniciar os intervalos de tempo longo e curto. Os circuitos de interface são incluídos no semáforo e a unidade de interface para converter os níveis de saída do circuito de acionamento das luzes para as tensões e correntes necessárias para ligar cada uma das luzes. A Figura 6–67 mostra um diagrama em bloco mais detalhado mostrando esses elementos essenciais.

Diagrama de Estados

Um diagrama de estados mostra graficamente a seqüência de estados num sistema e as condições para cada estado e para as transições de um estado para o próximo. Na realidade, a Figura 6–65 é uma forma de diagrama de estados porque mostra a seqüência de estados e as condições.

Definição de Variáveis Antes que um diagrama de estados tradicional possa ser desenvolvido, as variáveis é que determinam como as seqüências do sistema através dos estados tem que ser definidas. Essas variáveis e os seus símbolos são apresentadas a seguir:

- Presença de veículo na via secundária $= V_s$
- Temporizador de 25 s (temporizador longo) é *ligado* $= T_L$

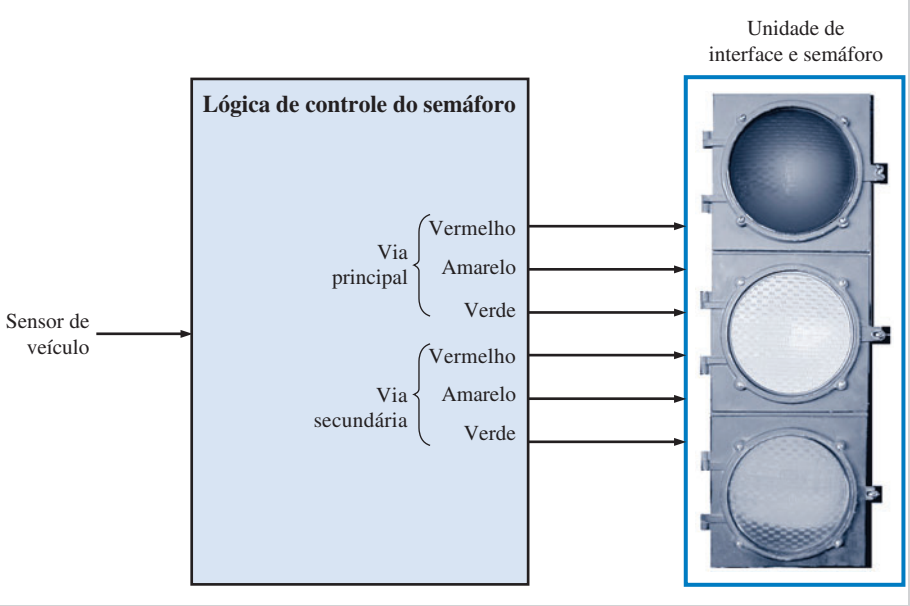


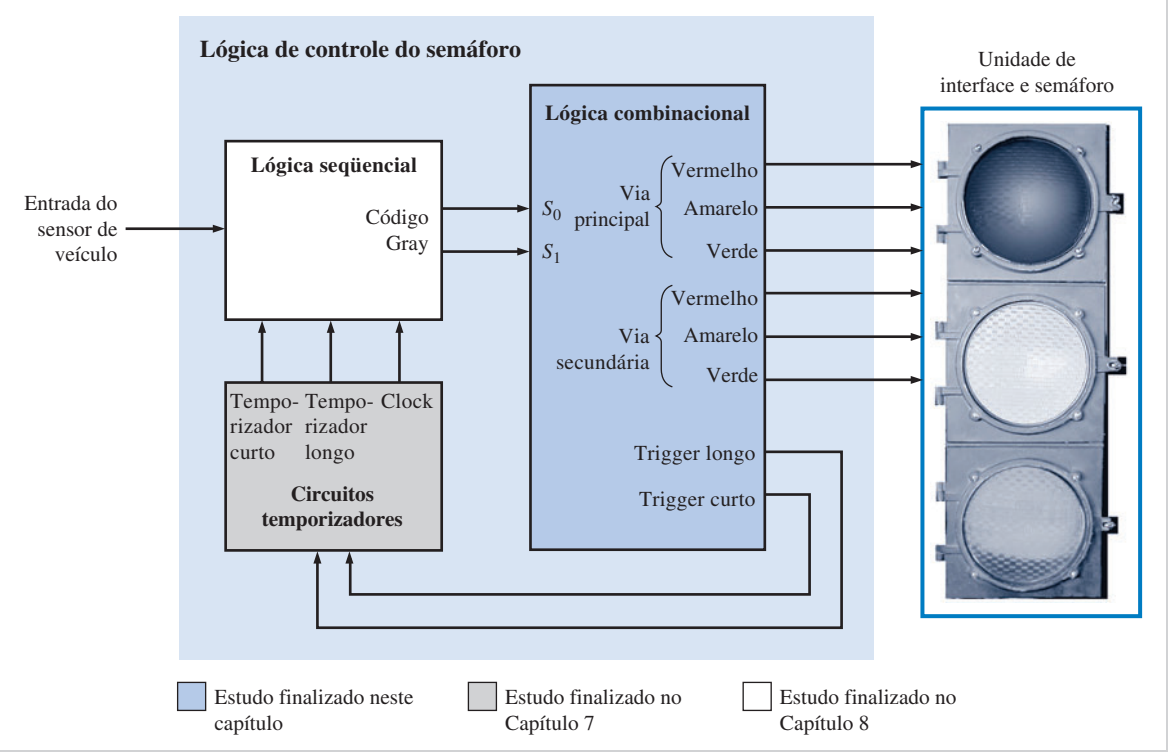
▲ **FIGURA 6–65**

Requisito para a seqüência do semáforo de trânsito.

▶ FIGURA 6–66

Um diagrama em bloco do sistema mínimo.





▲ FIGURA 6–67

Diagrama em bloco mostrando os elementos essenciais.

■ Temporizador de 4 s (temporizador curto) é ligado = T_S

O uso de variáveis complementadas indica as condições opostas. Por exemplo, $\overline{V}_S$ indica que não existe veículo na via secundária, $\overline{T}_L$ indica que o temporizador longo está desligado (*off*), $\overline{T}_S$ indica que o temporizador curto está desligado (*off*).

Descrição do Diagrama de Estados Um diagrama de estados é mostrado na Figura 6–68. Cada um dos quatro estados é indicado com uma seqüência de código

Gray de 2 bits, conforme indicado pelos círculos. A seta em loop em cada estado indica que o sistema permanece no estado sob a condição definida pela variável associada ou expressão. Cada uma das setas de um estado para o próximo indica uma transição de estado sob a condição definida pela variável associada ou expressão.

Primeiro estado O código Gray para esse estado é 00. A luz da via principal é verde e a luz da via secundária é vermelha. O sistema permanece nesse estado por pelo menos 25 s quando o temporizador longo é ligado (*on*) ou enquanto não existir veículo na via secundária ($T_L + \overline{V}_S$).. O sistema vai para o próximo estado quando o temporizador de 25 s estiver desligado e houver um veículo na via secundária ($\overline{T}_L V_S$).

Segundo estado O código Gray para esse estado é 01. A luz na via principal é amarela (atenção) e a luz na via secundária é vermelha. O sistema permanece nesse estado por 4 s quando o temporizador curto é ligado (T_S) e passa para o próximo estado quando o temporizador curto for desligado ($\overline{T}_S$).

Terceiro estado O código Gray para esse estado é 11. A luz na via principal é vermelha e a luz na via secundária é verde. O sistema permanece nesse estado quando o temporizador longo é ligado e existir um veículo na via secundária ($T_L V_s$). O sistema vai para o próximo estado quando decorrer os 25 s ou quando não houver veículo na via secundária, qualquer que ocorrer primeiro ($\overline{T}_L + \overline{V}_S$).

Quarto estado O código Gray para esse estado é 10. A luz na via principal é vermelha e a luz na via secundária é amarela. O sistema permanece nesse estado por 4 s quando o temporizador curto é ligado (T_S) e retorna ao primeiro estado quando o temporizador curto desligar ($\overline{T}_S$).

A Lógica Combinacional

O foco dessa aplicação nesse capítulo é a parte relativa à lógica combinacional do diagrama em bloco mostrado na Figura 6–67. Os circuitos de temporização e de lógica seqüencial serão objetivos das seções de aplicações de sistemas nos Capítulos 7 e 8.

Um diagrama em bloco para a lógica combinacional do sistema é desenvolvido como sendo o primeiro passo do projeto. As três funções que essa lógica tem que realizar são definidas a seguir, sendo que o diagrama resultante, com um bloco para cada uma das três funções, é mostrado na Figura 6–69:

- **Decodificador de estado** Decodifica os 2 bits do código Gray a partir da lógica seqüencial para determinar em qual dos quatro estados o sistema se encontra.
- **Circuito lógico de acionamento das luzes** Usa o estado decodificado para ativar as luzes apropriadas do semáforo para as unidades das vias principal e secundária.
- **Circuito lógico de trigger** Usa os estados decodificados para produzir sinais adequados para iniciar os temporizadores longo e curto.

Implementação da lógica Combinacional

Implementação da lógica do decodificador A parte relativa ao decodificador de estado tem duas entradas (código Gray de 2 bits) e uma saída para cada

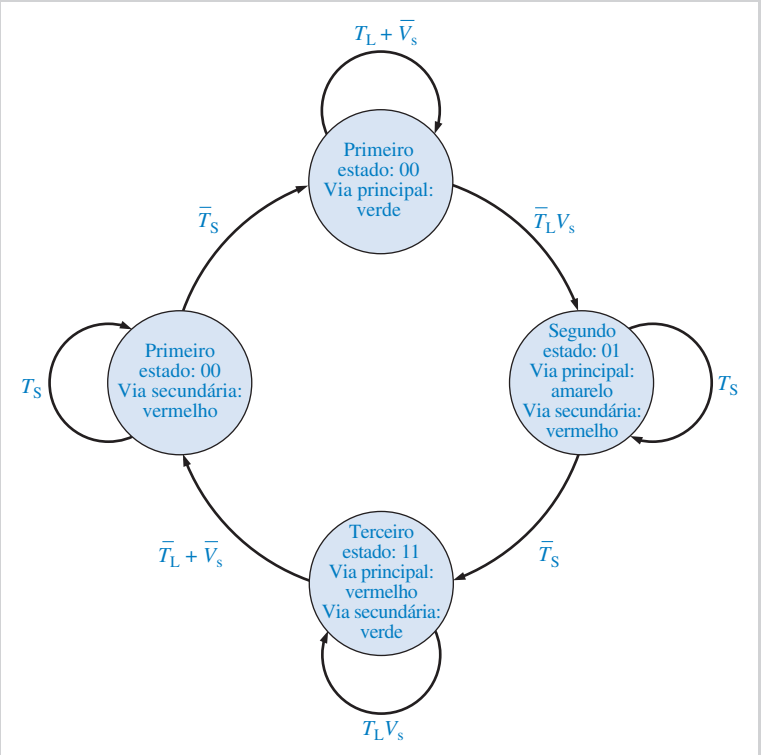
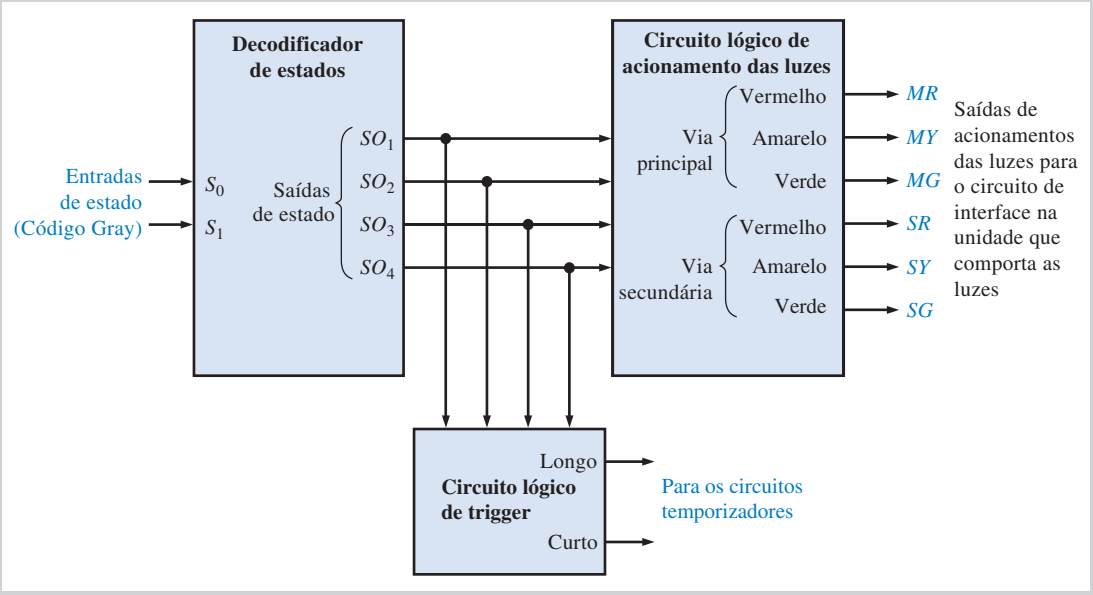
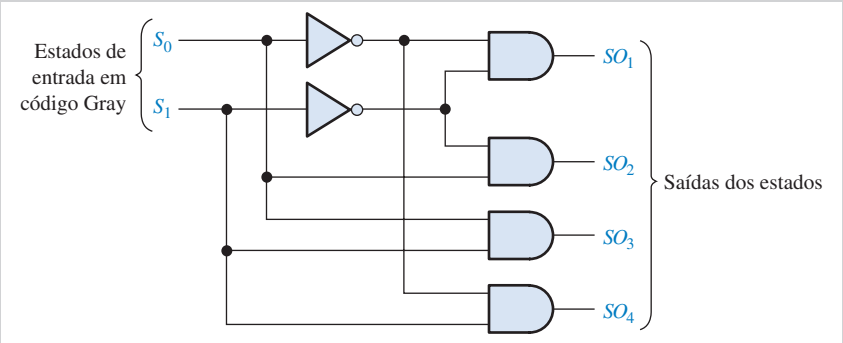


FIGURA 6–68

Diagrama de estados para o sistema de controle de semáforo mostrando a seqüência de código Gray.



▲ **FIGURA 6–69**
Diagrama em bloco da lógica combinacional.



▶ **FIGURA 6–70**
A lógica do decodificador de estado.

um dos quatro estados, conforme mostra a Figura 6–70. As duas entradas de código Gray são indicadas por S_0 e S_1 e as quatro saídas de estado são indicadas por SO_1, SO_2, SO_3 e SO_4. As expressões Booleanas para as saídas de estado são as seguintes:

$$SO_1 = \overline{S}_1\overline{S}_0$$
$$SO_2 = \overline{S}_1 S_0$$
$$SO_3 = S_1 S_0$$
$$SO_4 = S_1\overline{S}_0$$

A tabela-verdade para essa lógica do decodificador de estado é mostrada na Tabela 6–11.

Implementação da Lógica de Acionamento das Luzes

A lógica de acionamento das luzes recebe as quatro saídas de estado e produz seis saídas para ativação das luzes do semáforo. Essas saídas são indicadas por *MR*, *MY*, *MG* (para o vermelho (*red*), amarelo (*yellow*) e verde (*green*) da via principal (*main*)) e *SR*, *SY*, *SG* (para o vermelho, amarelo e verde da via secundária (*side*)). Consultando a tabela-verdade na Tabela 6–11, podemos ver que as saídas do semáforo podem ser expressas como

$$MR = SO_3 + SO_4$$
$$MY = SO_2$$
$$MG = SO_1$$
$$SR = SO_1 + SO_2$$
$$SY = SO_4$$
$$SG = SO_3$$

A lógica de acionamento das luzes pode ser implementada como mostra a Figura 6–71.

Implementação do Circuito Lógico de Trigger O circuito lógico de trigger produz duas saídas. A saída *longo* é uma transição de BAIXO para ALTO que dispara o circuito de temporização de 25 s quando o sistema entra no primeiro (00) ou no terceiro (11) estados. A saída *curto* é uma transição de BAIXO para ALTO que dis-

TABELA 6–11

Tabela-verdade para a lógica combinacional

ENTRADAS DE ESTADO		SAÍDAS DE ESTADO				SAÍDAS LUMINOSAS						SAÍDAS DE TRIGGER	
S_1	S_0	SO_1	SO_2	SO_3	SO_4	*MR*	*MY*	*MG*	*SR*	*SY*	*SG*	LONGO	CURTO
0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	1	0	1	0	0	0	1	0	1	0	0	0	1
1	1	0	0	1	0	1	0	0	0	0	1	1	0
1	0	0	0	0	1	1	0	0	0	1	0	0	1

As saídas de estado são ativas em nível ALTO e as saídas luminosas são ativas em nível ALTO. *MR* significa rua principal em vermelho (red), *SG* significa rua secundária em verde (green), etc.

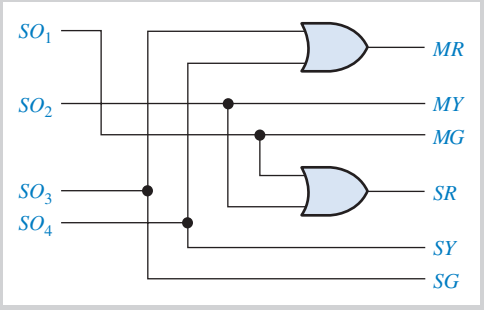


FIGURA 6–71

Circuito lógico de acionamento das luzes.

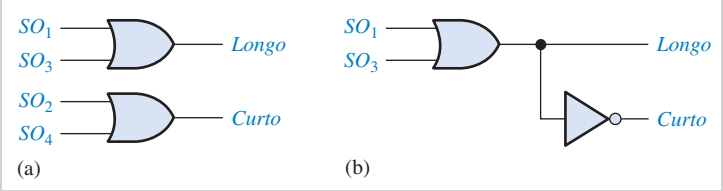


FIGURA 6–72

Circuito lógico de trigger.

para o circuito de temporização de 4 s quando o sistema passa para o segundo (01) ou quarto (10) estados. As saídas de disparo (trigger) são mostradas na Tabela 6–11 e na forma de equações como a seguir:

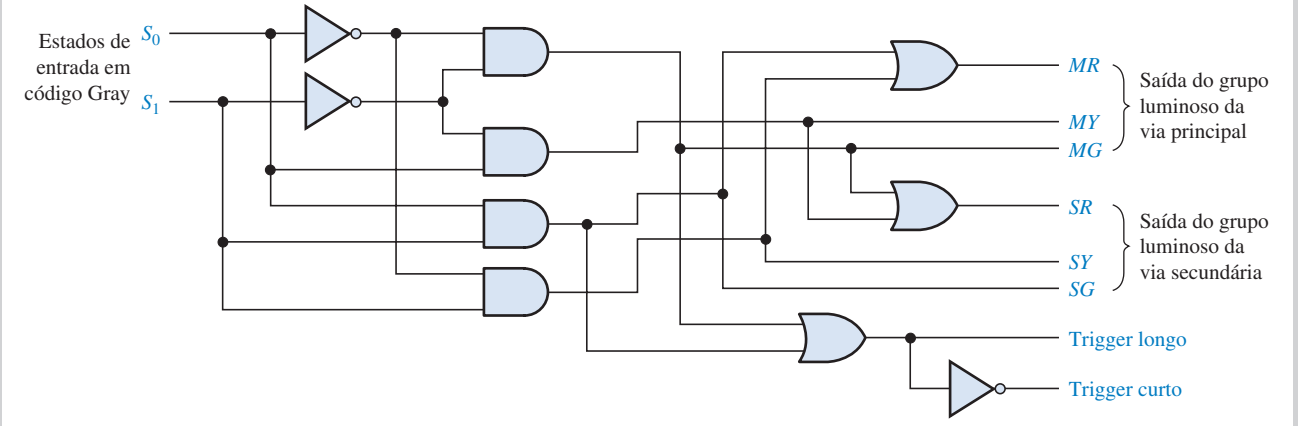
$$\textit{Trigger longo} = SO_1 + SO_3$$
$$\textit{Trigger curto} = SO_2 + SO_4$$

O circuito lógico de trigger é mostrado na Figura 6–72(a). A Tabela 6–11 mostra também que a saída Longo e a saída Curto são complementares, assim a lógica pode ser implementada com uma porta OR e um inversor, como mostra a parte (b) da figura.

A Figura 6–73 mostra a lógica combinacional completa que combina o decodificador de estados, o circuito lógico de acionamento das luzes e o circuito lógico de trigger.

Atribuições do Sistema

- *Atividade 1* Aplicar as formas de onda para o código Gray de 2 bits nas entradas S_0 e S_1 da lógica combinacional e desenvolver todas as formas de onda de saída.
- *Atividade 2* Mostre como você implementaria a lógica combinacional com funções 74XX.
- *Atividade opcional* Escreva um programa VHDL descrevendo a lógica combinacional.

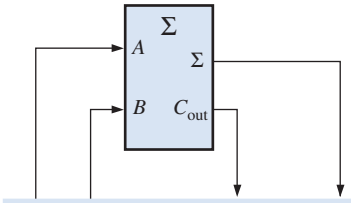


▲ **FIGURA 6–73**
Circuito lógico combinacional completo.

RESUMO

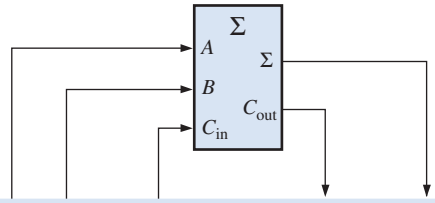
- As operações de meio-somador e somador-completo são resumidas na Figura 6–74.
- Os símbolos lógicos com a numeração de pinos para os CIs usados nesse capítulo são mostrados na Figura 6–75. As indicações dos pinos podem diferir em algumas folhas de dados de fabricantes.
- As funções lógicas padrão das séries 74XX estão disponíveis para uso em projetos de lógica programável.

Meio-somador



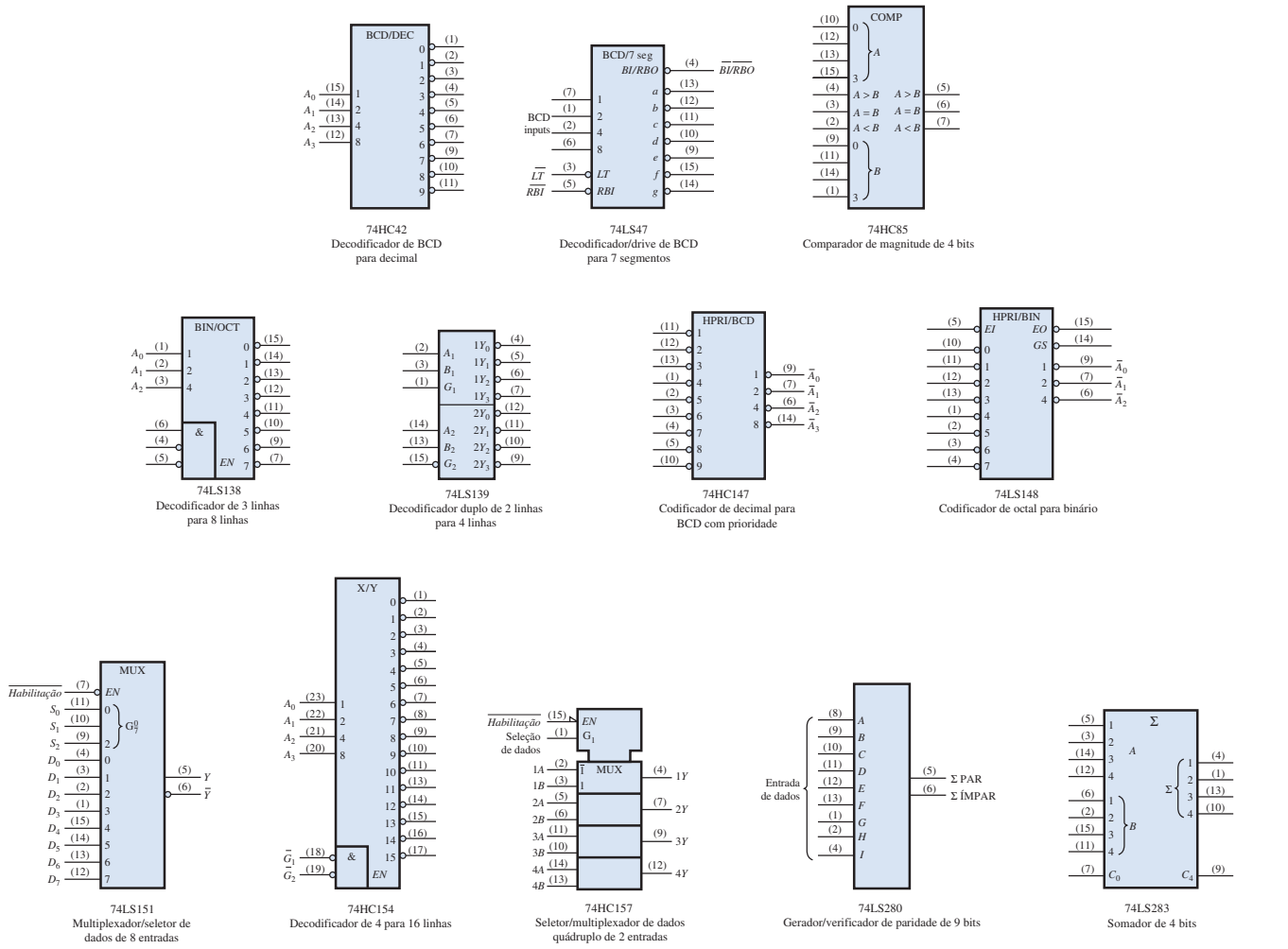
Entradas A	B	Saída de Carry C_{out}	Soma Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Somador-completo



Entradas A	B	Entrada de Carry C_{in}	Saída de Carry C_{out}	Soma Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

▶ **FIGURA 6–74**



▲ **FIGURA 6–75**

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Bit de paridade Um bit anexado em cada grupo de bits de informação para tornar o número total de 1s ímpar ou par para cada grupo de bits.

Carry antecipado Um método de adição binária por meio do qual os carries dos estágios somadores anteriores são antecipados, eliminando assim os atrasos de propagação do carry.

Carry ondulante Um método de adição binária no qual o carry de saída de cada somador se torna o carry de entrada do próximo somador de ordem maior.

Cascata Conexão da saída de um dispositivo para a entrada de um dispositivo similar, possibilitando que um dispositivo acione o outro para expandir a capacidade operacional.

Codificador Um circuito digital que converte informação para uma forma codificada.

Decodificador Um circuito digital que converte informação codificada numa forma familiar ou não-codificada.

Decodificador de prioridade Um codificador no qual apenas o dígito de entrada de valor maior é codificado e qualquer outra entrada ativa é ignorada.

Demultiplexador (DEMUX) Um circuito que comuta dados digitais de uma linha de entrada para diversas linhas de saída numa seqüência temporal especificada.

Glitch Um spike de tensão ou corrente de curta duração, geralmente produzido não-intencionalmente e indesejável.

Meio-somador Um circuito digital que soma dois bits e produz uma soma e um carry de saída. Ele não tem a capacidade de operar com carry de entrada.

Multiplexador (MUX) Um circuito que comuta dados digitais de diversas linhas de entrada para uma única linha de saída numa seqüência temporal especificada.

Somador-completo Um circuito digital que soma dois bits e um carry de entrada para produzir uma soma e um carry de saída.

AUTOTESTE

As respostas estão no final do capítulo.

1. Um meio-somador é caracterizado por
   (a) duas entradas e duas saídas
   (b) três entradas e duas saídas
   (c) duas entradas e três saídas
   (d) duas entradas e uma saída
2. Um somador-completo é caracterizado por
   (a) duas entradas e duas saídas
   (b) três entradas e duas saídas
   (c) duas entradas e três saídas
   (d) duas entradas e uma saída
3. As entradas para um somador-completo são $A = 1$, $B = 1$, $C_{in} = 0$. As saídas são
   (a) $\sum = 1, C_{out} = 1$ **(b)** $\sum = 1, C_{out} = 0$
   (c) $\sum = 0, C_{out} = 1$ **(a)** $\sum = 0, C_{out} = 0$
4. Um somador paralelo de 4 bits pode somar
   (a) dois números binários de 4 bits
   (b) dois números binários de 2 bits
   (c) quatro bits de cada vez
   (d) quatro bits em seqüência
5. Para expandir um somador paralelo de 4 bits para um somador paralelo de 8 bits, temos que
   (a) usar quatro somadores de 4 bits sem interconexões
   (b) usar dois somadores de 4 bits conectando as saídas soma de um nas entradas de bit do outro
   (c) usar oito somadores de 4 bits sem interconexões
   (d) usar dois somadores de 4 bits com a saída de carry de um conectada na entrada de carry do outro
6. Se um CI comparador de magnitude 74HC85 tem $A = 1011$ e $B = 1001$ em suas entradas, as saídas são
   (a) $A > B = 0, A < B = 1, A = B = 0$ **(b)** $A > B, A < B = 0, A = B = 0$
   (c) $A > B = 1, A < B = 1, A = B = 0$ **(d)** $A > B, A < B = 0, A = B = 1$
7. Se um decodificador 1 de 16 com saídas ativas em nível BAIXO apresenta um nível BAIXO na saída decimal 12, quais são os bits nas entradas?
   (a) $A_3A_2A_1A_0 = 1010$ **(b)** $A_3A_2A_1A_0 = 1110$
   (b) $A_3A_2A_1A_0 = 1100$ **(c)** $A_3A_2A_1A_0 = 0100$
8. Um decodificador de BCD para 7 segmentos tem 0100 em suas entradas. As saídas ativas são
   (a) a, c, f, g **(b)** b, c, f, g **(c)** b, c, e, f **(d)** b, d, e, g
9. Se um codificador de prioridade de octal para binário tem as suas entradas 0, 2, 5 e 6 no nível ativo, a saída binária ativa em nível ALTO é
   (a) 110 **(b)** 010 **(c)** 101 **(d)** 000
10. Em geral, um multiplexador tem
    (a) uma entrada de dados, diversas saídas de dados e entradas de seleção
    (b) uma entrada de dados, uma saída de dados e uma entrada de seleção
    (c) diversas entradas de dados, diversas saídas de dados e entradas de seleção
    (d) diversas entradas de dados, uma saída de dados e entradas de seleção
11. Seletores de dados são basicamente o mesmo que
    (a) decodificadores **(b)** demultiplexadores **(c)** multiplexadores **(d)** codificadores
12. Qual dos seguintes códigos apresenta paridade par?
    (a) 10011000 **(b)** 01111000 **(c)** 11111111
    (d) 11010101 **(e)** todos **(f)** as alternativas (b) e (c) estão corretas

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 6–1 Somadores Básicos

1. Para o somador-completo visto na Figura 6–4, determine o estado lógico (1 ou 0) da saída de cada porta para as entradas a seguir:

(a) $A = 1, B = 1, C_{in} = 1$ **(b)** $A = 0, B = 1, C_{in} = 1$ **(c)** $A = 0, B = 1, C_{in} = 0$

2. Quais são os níveis nas entradas de um somador-completo que produz as seguintes saídas:

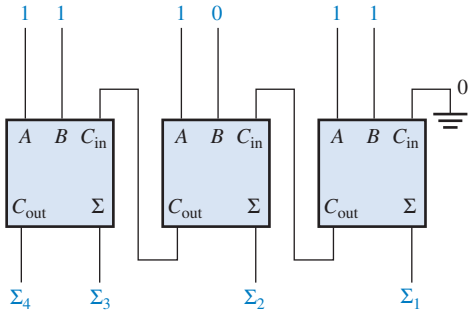
(a) $\sum = 0, C_{out} = 0$ **(b)** $\sum = 1, C_{out} = 0$
(c) $\sum = 1, C_{out} = 1$ **(d)** $\sum = 0, C_{out} = 1$

3. Determine as saídas de um somador-completo para cada uma das seguintes entradas:

(a) $A = 1, B = 0, C_{in} = 0$ **(b)** $A = 0, B = 0, C_{in} = 1$
(c) $A = 0, B = 1, C_{in} = 1$ **(d)** $A = 1, B = 1, C_{in} = 1$

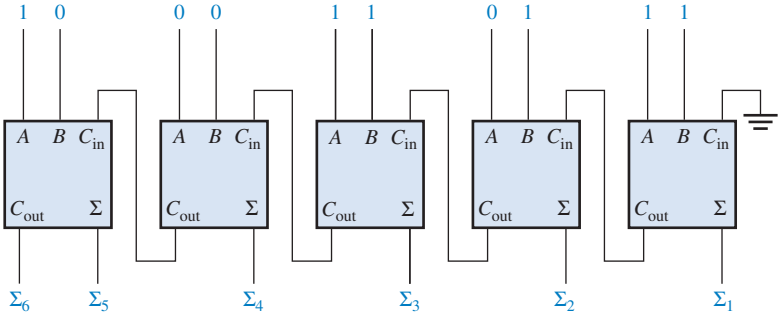
SEÇÃO 6–2 Somadores Binários Paralelos

4. Para o somador paralelo mostrado na Figura 6–76, determine a soma completa analisando a operação lógica do circuito. Verifique o seu resultado fazendo a soma manualmente com os dois números de entrada.



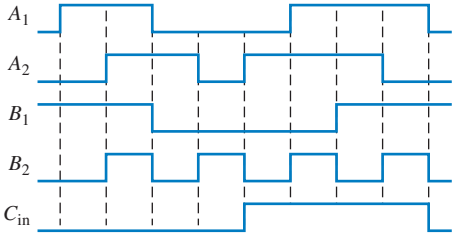
▶ FIGURA 6–76

5. Repita o Problema 4 para o circuito e as condições de entrada vistos na Figura 6–77.



▶ FIGURA 6–77

6. As formas de onda de entrada mostradas na Figura 6–78 são aplicadas num somador de 2 bits. Determine as formas de onda para a soma e o carry de saída em relação às entradas construindo o diagrama de temporização.



▶ FIGURA 6–78

7. As seqüências de bits a seguir (o primeiro bit é o mais à direita) aparece nas entradas de um somador paralelo de 4 bits. Determine a seqüência resultante dos bits em cada saída soma.

A_1	1001
A_2	1110
A_3	0000
A_4	1011
B_1	1111
B_2	1100
B_3	1010
B_4	0010

8. No processo de verificação do CI somador paralelo de 4 bits 74LS283 são observados os seguintes níveis de tensão nos seu pinos: 1-ALTO, 2-ALTO, 3-ALTO, 4-ALTO, 5-BAIXO, 6-BAIXO, 7-BAIXO, 9-ALTO, 10-BAIXO, 11-ALTO, 12-BAIXO, 13-ALTO, 14-ALTO E 15-ALTO. Determine se o CI está funcionando adequadamente.

SEÇÃO 6–3 Somadores com Carry Ondulante *Versus* Somadores com Carry Antecipado

9. Cada um dos oito somadores-completos em um somador paralelo de 8 bits com carry ondulante apresenta os seguintes atrasos de propagação:

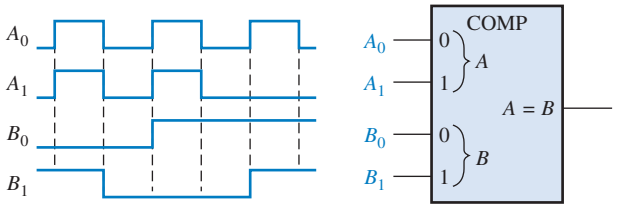
A para Σ e C_{out}:	40 ns
B para Σ e C_{out}:	40 ns
C_{in} para Σ:	35 ns
C_{in} para C_{out}:	25 ns

Determine o tempo total máximo para a adição de dois números de 8 bits.

10. Mostre o circuito lógico adicional necessário para tornar um somador de 4 bits com carry antecipado, visto na Figura 6–18, num somador de 5 bits.

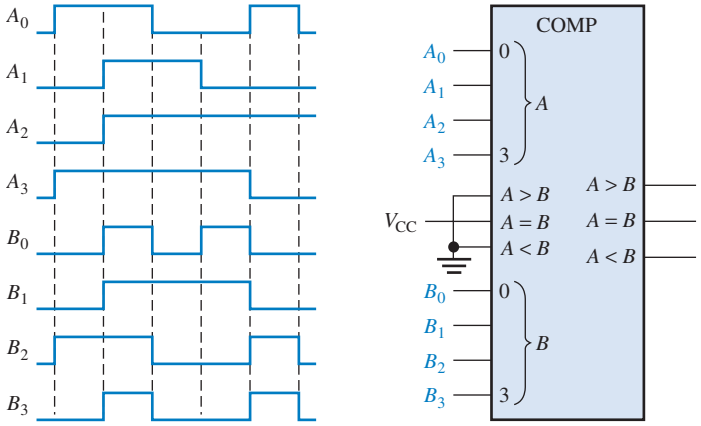
SEÇÃO 6–4 Comparadores

11. As formas de onda mostradas na Figura 6–79 são aplicadas ao comparador como mostra a figura. Determine a forma de onda de saída ($A = B$).



▶ FIGURA 6–79

12. Para o comparador de 4 bits visto na Figura 6–80, desenhe cada forma de onda de saída para as entradas mostradas.



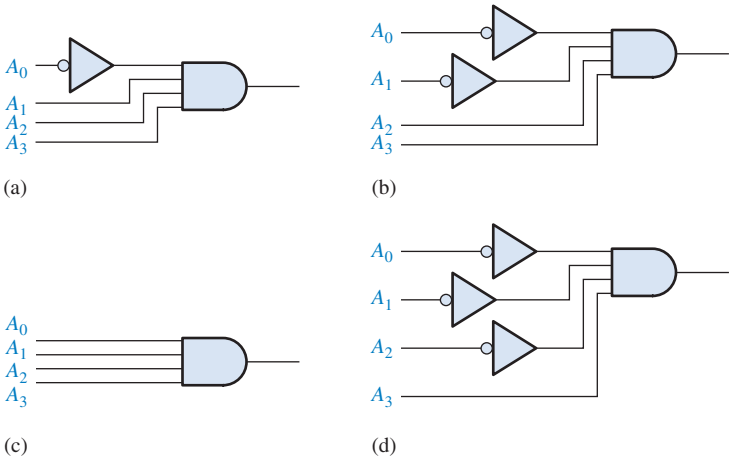
▶ FIGURA 6–80

13. Para cada conjunto de números binários a seguir, determine os estados da saída para o comparador mostrado na Figura 6–22.

(a) $A_3A_2A_1A_0 = 1100$ $B_3B_2B_1B_0 = 1001$ **(b)** $A_3A_2A_1A_0 = 1000$ $B_3B_2B_1B_0 = 1011$ **(c)** $A_3A_2A_1A_0 = 0100$ $B_3B_2B_1B_0 = 0100$

SEÇÃO 6–5 Decodificadores

14. Quando um nível ALTO estiver presente em cada uma das portas de decodificação vistas na Figura 6–81, qual é o código binário que aparece nas entradas? A_3 é o MSB.



▶ FIGURA 6–81

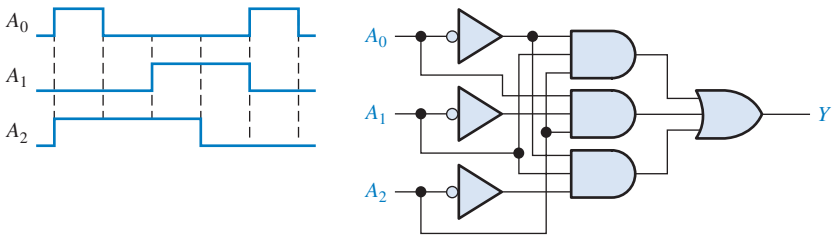
15. Mostre a lógica de decodificação para cada um dos seguintes códigos se uma saída ativa em nível ALTO for necessária:

(a) 1101 **(b)** 1000 **(c)** 11011 **(d)** 11100
(e) 101010 **(f)** 111110 **(g)** 000101 **(h)** 1110110

16. Resolva o Problema 15 sendo que uma saída ativa em nível BAIXO é necessária:

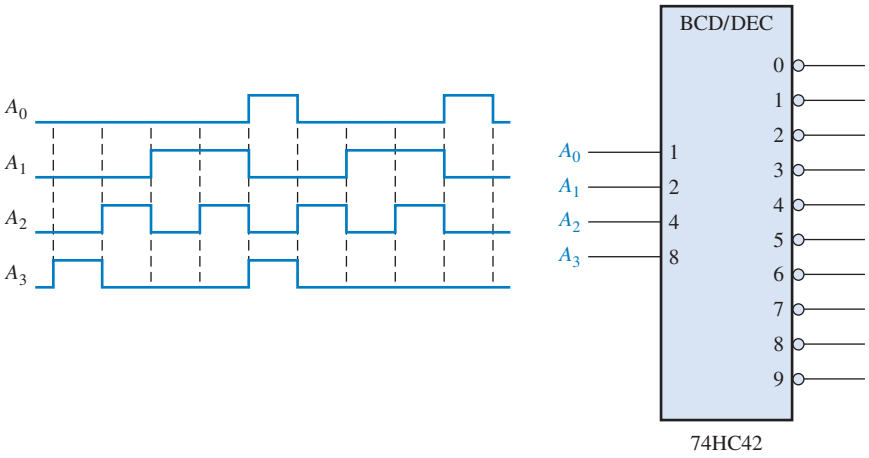
17. Você deseja apenas detectar a presença dos códigos 1010, 1100, 0001 e 1011. Uma saída ativa em nível ALTO é necessária para indicar a presença desses códigos. Desenvolva uma lógica de decodificação mínima com uma única saída que indique quando qualquer um desses códigos estiver nas entradas. Para qualquer outro código a saída tem que ser nível BAIXO.

18. Se as formas de onda de entrada são aplicadas na lógica de decodificação vista na Figura 6–82, determine a forma de onda de saída relacionando com as entradas.



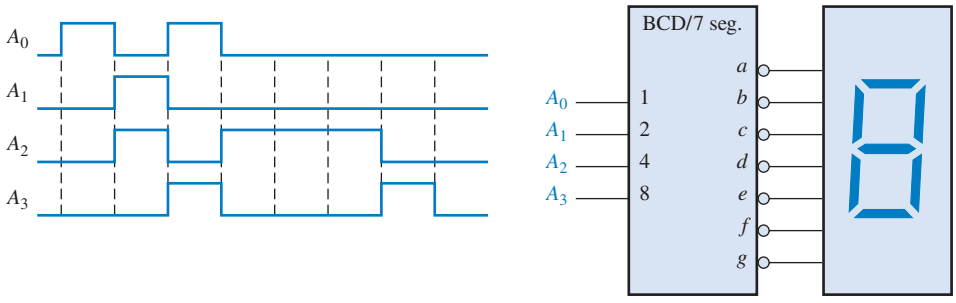
▶ FIGURA 6–82

19. Números BCD são aplicados seqüencialmente no decodificador de BCD para decimal conforme a Figura 6–83. Desenhe o diagrama de temporização mostrando cada saída devidamente relacionada com as outras e com as entradas.



▶ FIGURA 6–83

20. Um decodificador/driver de 7 segmentos aciona um display conforme mostra a Figura 6–84. Se as formas de onda são aplicadas conforme indicado, determine a seqüência de dígitos que aparecem no display.



▶ FIGURA 6–84

SEÇÃO 6–6 Codificadores

21. Para o codificador de decimal para BCD mostrado na Figura 6–38, considere que as entradas 9 e 3 sejam nível ALTO. Qual o código de saída? É um código BCD (8421) válido?

22. Um CI codificador 74HC147 tem nível BAIXO nos pinos 2, 5 e 12. Qual o código BCD que aparece nas saídas se todas as outras entradas estiverem em nível ALTO?

SEÇÃO 6–7 Conversores de Códigos

23. Converta os seguintes números decimais para BCD e em seguida para binário.

(a) 2 **(b)** 8 **(c)** 13 **(d)** 26 **(e)** 33

24. Mostre a lógica necessária para converter um número binário de 10 bits para código Gray e use essa lógica para converter os seguintes números binários para código Gray:

(a) 1010101010 **(b)** 1111100000 **(c)** 0000001110 **(d)** 1111111111

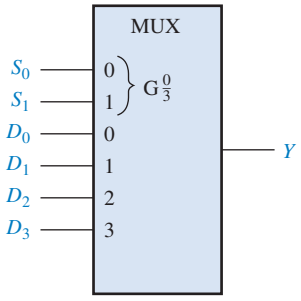
25. Mostre a lógica necessária para converter um código Gray de 10 bits para binário e use essa lógica para converter os seguintes códigos Gray para binário:

(a) 1010000000 **(b)** 0011001100 **(c)** 1111000111 **(d)** 0000000001

SEÇÃO 6–8 Multiplexadores (Seletores de Dados)

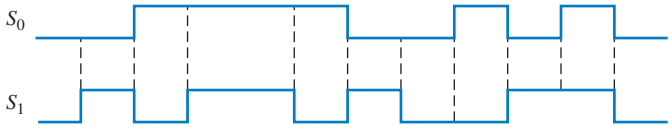
26. Para o multiplexador visto na Figura 6–85, determine a saída para os seguintes estados das entradas:

$D_0 = 0, D_1 = 1, D_2 = 1, D_3 = 0, S_0 = 1, S_1 = 0.$



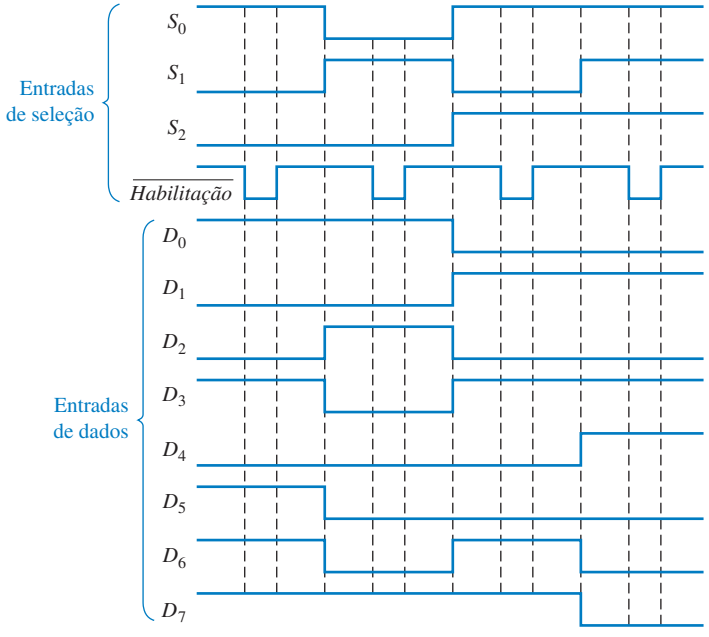
▶ FIGURA 6–85

27. Se as entradas de seleção de dados do multiplexador mostrado na Figura 6–85 são seqüenciadas como mostrado pelas formas de onda na Figura 6–86, determine a forma de onda de saída com as entradas de dados especificadas no Problema 26.



▶ FIGURA 6–86

28. As formas de onda vistas na Figura 6–87 são observadas nas entradas de um CI multiplexador de 8 entradas 74LS151. Desenhe a forma de onda da saída Y.



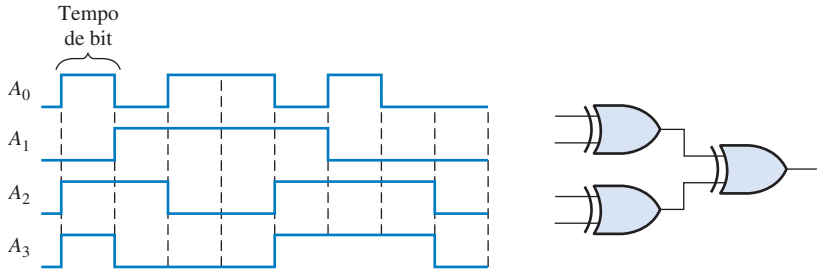
▶ FIGURA 6–87

SEÇÃO 6–9 Demultiplexadores

29. Desenvolva o diagrama de temporização total (entradas e saídas) para um 74HC154 usado numa aplicação de demultiplexação na qual as entradas são as seguintes: As entradas de seleção de dados recebem de forma repetitiva e em seqüência a saída de um contador que inicia com 0000 e a entrada de dados é um fluxo de dados em série que representa o número 2468 em BCD. O dígito menos significativo (8) é o primeiro na seqüência, sendo o primeiro bit o LSB, e ele deve aparecer nas posições dos 4 primeiros bits na saída.

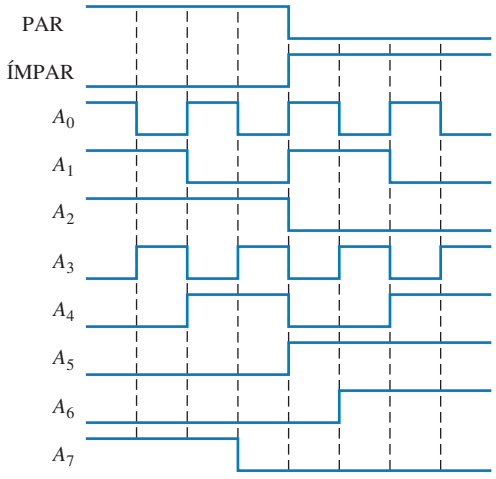
SEÇÃO 6–10 Geradores/Verificadores de Paridade

30. As formas de onda mostradas na Figura 6–88 são aplicadas à lógica de paridade de 4 bits. Determine a forma de onda de saída relacionando-a às entradas. Em quantos tempos de bit a paridade par ocorre e como ela é indicada? O diagrama de temporização inclui 8 tempos de bit.



▶ FIGURA 6–88

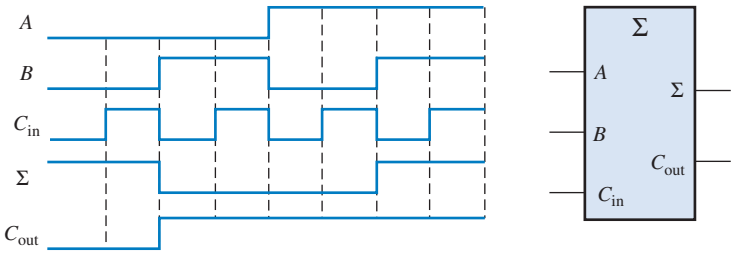
31. Determine as saídas ∑ Par e ∑ Ímpar de um CI gerador/verificador de paridade de 9 bits 74LS280 para as entradas mostradas na Figura 6–89. Consulte a tabela de funções na Figura 6–59.



▶ FIGURA 6–89

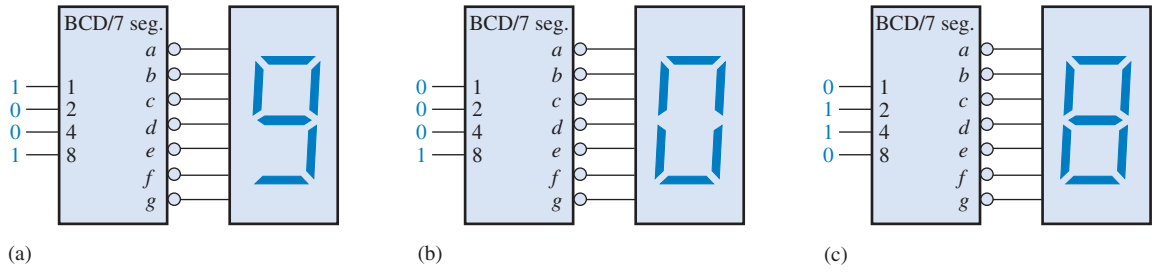
SEÇÃO 6–11 Análise de Defeito

32. O somador-completo visto na Figura 6–90 é testado para todas as condições de entrada com das formas de onda mostradas. A partir da sua observação das formas de onda de ∑ e C_{out}, a operação do somador é adequada? Em caso negativo, qual é o defeito mais provável?



▲ FIGURA 6–90

33. Faça uma lista dos possíveis defeitos para cada decodificador/display da Figura 6–91.

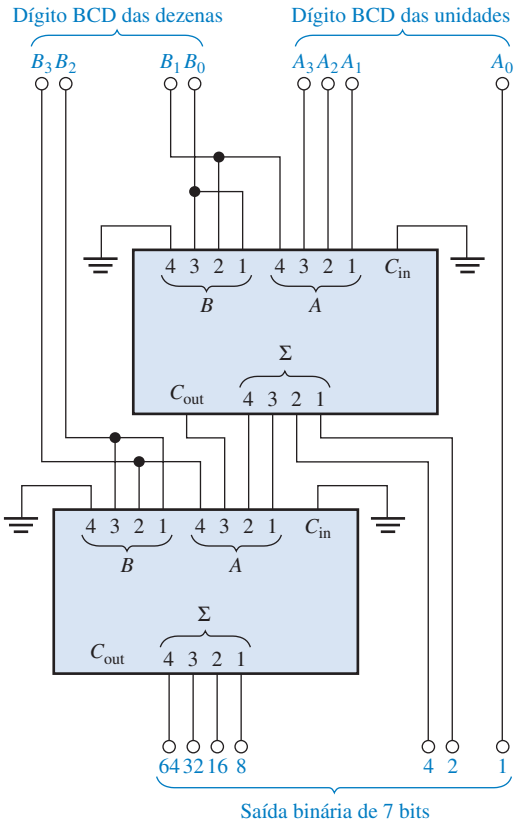


▲ **FIGURA 6–91**

34. Desenvolva um procedimento de teste sistemático para verificar a operação completa do codificador de teclado mostrado na Figura 6–42.

35. Suponha que você esteja testando um conversor de BCD para binário que consiste de 4 somadores conforme mostra a Figura 6–92. Primeiro verifique que o circuito converte de BCD para binário. O procedimento de teste diz para aplicar números BCD em ordem seqüencial começando com 0_{10} e verificando se há uma saída binária correta. Qual o sintoma, ou sintomas, que aparecem nas saídas binárias para cada um dos seguintes defeitos? Para qual número BCD cada um dos defeitos é detectado *primeiro*?

(a) A entrada A_1 está aberta (somador superior).
(b) A saída C_{out} está aberta (somador superior).
(c) A saída Σ_4 está em curto-circuito com GND (somador superior).
(d) A saída de peso 32 está em curto-circuito com GND (somador inferior).



▶ **FIGURA 6–92**

36. Para o sistema de multiplexação com display mostrado na Figura 6–52, determine a causa, ou as causas, mais prováveis para cada um dos seguintes sintomas:

(a) O display do dígito *B* (MSD) não liga mesmo.
(b) Nenhum display de 7 segmentos liga.

(c) O segmento *f* dos dois displays se apresenta ligado o tempo todo.

(d) Existe uma cintilação (*flicker*) visível nos displays.

37. Desenvolva um procedimento sistemático para testar completamente as funções do CI seletor de dados 74LS151.

38. Durante o teste do sistema de transmissão de dados visto na Figura 6–60, é aplicado um código nas entradas de D_0 a D_6 o qual contém um número ímpar de 1s. Um único erro de bit é deliberadamente introduzido na linha de transmissão de dados seriais entre o MUX e o DEMUX, porém o sistema não indica erro (saída erro = 0). Após uma investigação, você observa as entradas do verificador de paridade par e constata que os bits de D_0 a D_6 contêm um número par de 1s, como era esperado. Além disso, você constata que o bit de paridade (D_7) é nível 1. Quais são as possíveis razões para o sistema não indicar erro?

39. Descreva, de forma geral, qual seria o procedimento para um teste completo no sistema de transmissão de dados visto na Figura 6–60, e especifique um método para introdução de erros de paridade.

Aplicações em Sistemas Digitais

40. A lógica de acionamento de luzes pode ser implementada nessa aplicação de sistema fazendo uso de dispositivos lógicos de função fixa como o CI 74LS08 com portas AND operando como portas NOR negativas. Use um 74LS00 (quatro portas NAND) e qualquer outro dispositivo que seja necessário para produzir saídas ativas em nível BAIXO para as entradas dadas.

41. Implemente a lógica de acionamento de luzes com o CI 74LS00 se for necessário a saídas ativas em nível BAIXO.

Problemas Especiais de Projeto

42. Modifique o projeto do sistema de multiplexação de display de 7 segmentos mostrado na Figura 6–52 para acomodar dois dígitos adicionais.

43. Usando a Tabela 6–2, escreva as expressões na forma de soma-de-produtos para as saídas $\sum$ e C_{out} de um somador-completo. Use um mapa de Karnaugh para minimizar as expressões e em seguida implemente-as com inversores e lógica AND-OR. Mostre como substituir a lógica AND-OR com CIs seletores de dados 74LS151.

44. Implemente a função lógica especificada na Tabela 6–12 usando um CI seletor de dados 74LS151.

► **TABELA 6–12**

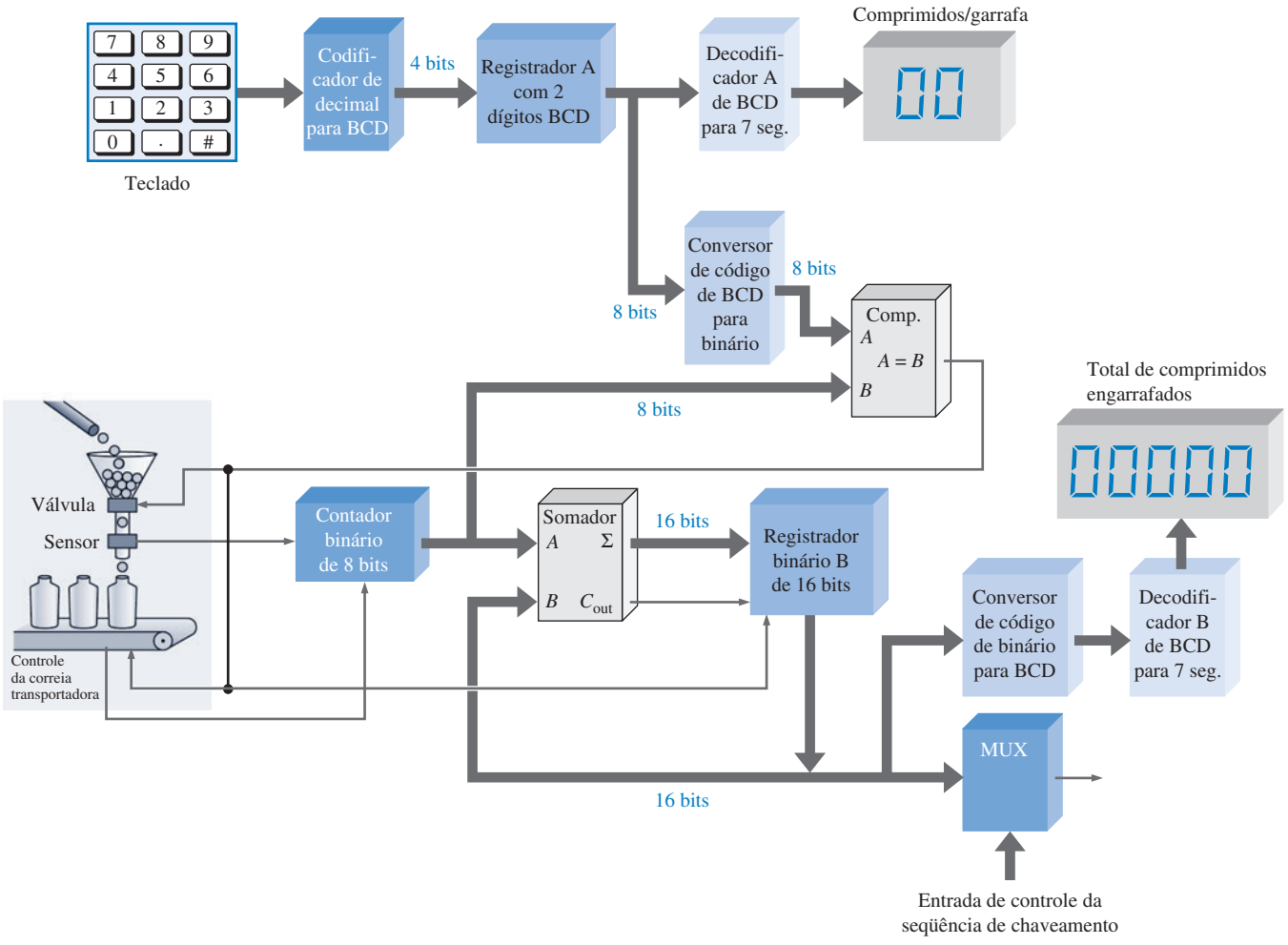
ENTRADAS				SAÍDA
A_3	A_2	A_1	A_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

45. Usando dois módulos somadores de 6 posições a partir da Figura 6–14, projete um sistema de votação de 12 posições.

46. O bloco somador usado no sistema de controle e contagem de comprimidos visto na Figura 6–93 realiza a adição do número binário de 8 bits a partir do contador com o número binário de 16 bits a partir do registrador B. O resultado do somador volta para o registrador B. Use CIs 74LS283 para implementar essa função e desenhe um diagrama lógico completo incluindo os números dos pinos. Consulte o tópico de aplicação em sistemas no Capítulo 1 para rever a operação.

47. Use CIs 74LS85 para implementar o bloco comparador no sistema de controle e contagem de comprimidos mostrado na Figura 6–93 e desenhe um diagrama lógico completo incluindo os números dos pinos. O comparador compara números binários de 8 bits (na realidade são necessários apenas 7 bits) do conversor de BCD para binário com os 8 bits do contador.

48. Dois decodificadores de BCD para 7 segmentos são usados no sistema de controle e contagem de comprimidos mostrado na Figura 6–93. Um deles é necessário para acionar um display de *comprimidos/garrafas* de 2 dígitos e o outro para acionar um display do *total de comprimidos engarrafados* de 5 dígitos. Use CIs 74LS47 para implementar cada decodificador e desenhe um diagrama lógico completo incluindo os números dos pinos.



▲ **FIGURA 6–93**

49. O codificador mostrado no diagrama em bloco do sistema visto na Figura 6–93 codifica cada tecla decimal acionada convertendo-a para BCD. Use um CI 74HC147 para implementar essa função e desenhe um diagrama lógico completo incluindo os números dos pinos.

50. O sistema visto na Figura 6–93 necessita de dois conversores de código. O conversor de binário para BCD transforma o número BCD de 2 dígitos no Registrador A para um código binário de 8 bits (na realidade apenas 7 bits são necessários porque o MSB é sempre 0). Use CIs conversores de código apropriados para implementar o conversor de BCD para binário e desenhe um diagrama lógico completo incluindo os números dos pinos.

PRÁTICA DE ANÁLISE DE DEFEITO USANDO O MULTISIM

51. Abra o arquivo P06-51 e teste o circuito lógico para determinar se existe um defeito. Em caso afirmativo, identifique-o se possível.

52. Abra o arquivo P06-52 e teste o circuito lógico para determinar se existe um defeito. Em caso afirmativo, identifique-o se possível.

53. Abra o arquivo P06-53 e teste o circuito lógico para determinar se existe um defeito. Em caso afirmativo, identifique-o se possível.

54. Abra o arquivo P06-54 e teste o circuito lógico para determinar se existe um defeito. Em caso afirmativo, identifique-o se possível.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 6–1 Somadores Básicos

1. **(a)** $\sum = 1, C_{out} = 0$ **(b)** $\sum = 0, C_{out} = 0$
(c) $\sum = 1, C_{out} = 0$ **(d)** $\sum = 0, C_{out} = 1$

2. $\sum = 1, C_{out} = 1$

SEÇÃO 6–2 Somadores Binários Paralelos

1. $C_{out}\sum_4\sum_3\sum_2\sum_1 = 11001$

2. São necessários três CIs 74LS283 para somar dois números de 10 bits.

SEÇÃO 6–3 Somadores com Carry Ondulante *Versus* Somadores com Carry Antecipado

1. $C_g = 0, C_p = 1$

2. $C_{out} = 1$

SEÇÃO 6–4 Comparadores

1. $A > B = 1, A < B = 0, A = B = 0$ quando $A = 1011$ e $B = 1010$

2. Comparador da direita: pino 7: $A < B = 1$; pino 6: $A = B = 0$; pino 5: $A > B = 0$
Comparador da esquerda: pino 7: $A < B = 0$; pino 6: $A = B = 0$; pino 5: $A > B = 1$

SEÇÃO 6–5 Decodificadores

1. A saída 5 é ativada quando 101 estiver nas entradas.

2. Quatro CIs 74HC154 são usados para decodificar um número binário de 6 bits.

3. Uma saída ativa em nível BAIXO aciona um display de LED de catodo comum.

SEÇÃO 6–6 Codificadores

1. **(a)** $A_0 = 1, A_1 = 1, A_2 = 0, A_3 = 1$
(b) Não, esse não é um código BCD válido.
(c) Apenas uma entrada pode ser ativada para uma saída válida.

2. **(a)** $\overline{A}_3 = 0, \overline{A}_2 = 1, \overline{A}_1 = 1, \overline{A}_0 = 1$
(b) A saída é 0111, que é o complemento de 1000 (8).

SEÇÃO 6–7 Conversores de Códigos

1. 10000101 (BCD) = 1010101_2
2. Um conversor de 8 bits de binário para Gray consiste de sete portas EX-OR num arranjo como o da Figura 6–43.

SEÇÃO 6–8 Multiplexadores (Seletores de Dados)

1. A saída é 0.
2. **(a)** 74LS157: quatro seletores de dados de 2 entradas
   (b) 74LS151: seletor de dados de 8 entradas
3. A saída de dados alterna entre BAIXO e ALTO conforme a seqüência das entradas de seleção de dados através dos estados binários.
4. **(a)** O CI 74HC157 multiplexa os dois códigos BCD para o decodificador de 7 segmentos.
   (b) O CI 74LS47 decodifica o número BCD para energizar o display.
   (c) O CI 74 LS139 habilita alternadamente os displays de 7 segmentos.

SEÇÃO 6–9 Demultiplexadores

1. Um decodificador pode ser usado como um multiplexador usando as linhas de entrada para a seleção de dados e uma linha *Enable* (Habilitação) para a entrada de dados.
2. As saídas são todas nível ALTO exceto D_{10}, que é nível BAIXO.

SEÇÃO 6–10 Geradores/Verificadores de Paridade

1. **(a)** Paridade par: 1110100 **(b)** Paridade par: 001100011
2. **(a)** Paridade ímpar: 11010101 **(b)** Paridade ímpar: 11000001
3. **(a)** O código está correto, quatro 1s. **(b)** O código está com erro, sete 1s.

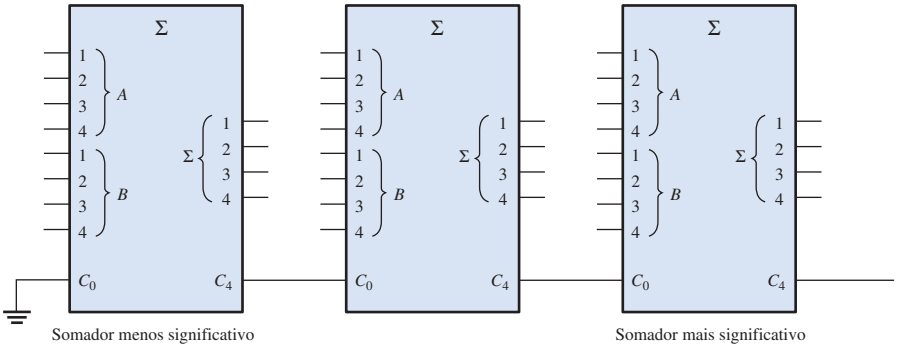
SEÇÃO 6–11 Análise de Defeito

1. Um glitch é um spike de tensão muito curto (geralmente indesejado).
2. Os glitches são provocados pelos estados de transição.
3. Strobe é a habilitação de um dispositivo por um período especificado quando o dispositivo não está em transição.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

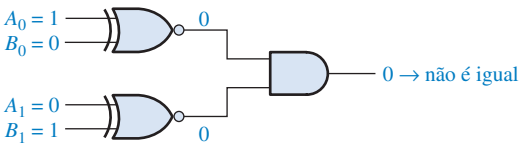
6–1. $\Sigma = 1$, $C_{out} = 1$ **6–2.** $\Sigma_1 = 0$, $\Sigma_2 = 0$, $\Sigma_3 = 1$, $\Sigma_4 = 1$

6–3. 1011 + 1010 = 10101 **6–4.** Veja a Figura 6–94.



▶ FIGURA 6–94

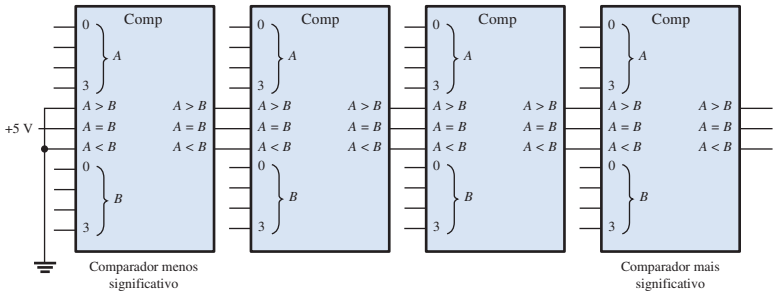
6–5 Veja a Figura 6–95.



▶ FIGURA 6–95

6–6. $A > B = 0, A = B = 0, A < B = 1$

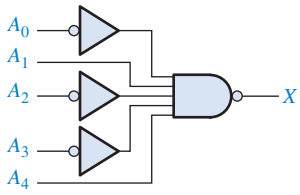
6–7. Veja a Figura 6–96.



▶ FIGURA 6–96

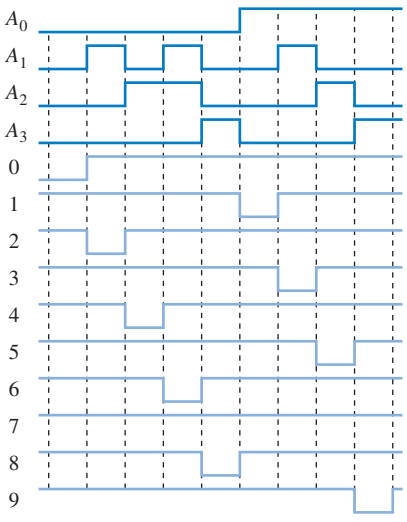
6–8. Veja a Figura 6–97.

6–9. Saída 22.



▶ FIGURA 6–97

6–10. Veja a Figura 6–98.



▶ FIGURA 6–98

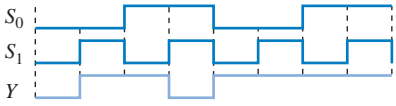
6–11. Todas as entradas em nível BAIXO: $\overline{A}_0 = 0, \overline{A}_1 = 1, \overline{A}_2 = 1, \overline{A}_3 = 0$
Todas as entradas em nível ALTO: todas as saídas em nível ALTO.

6–12. BCD 01000001

	00000001	1
	00101000	40
Binário	00101001	41

6–13. Sete portas EX-OR.

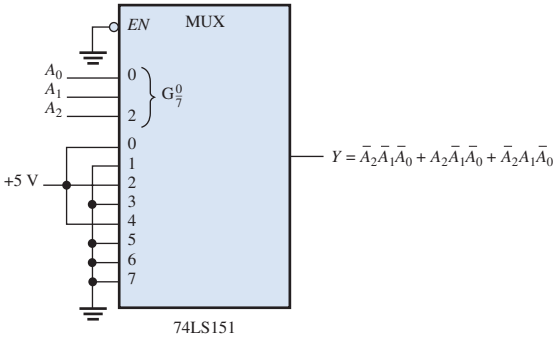
6–14. Veja a Figura 6–99.



▶ FIGURA 6–99

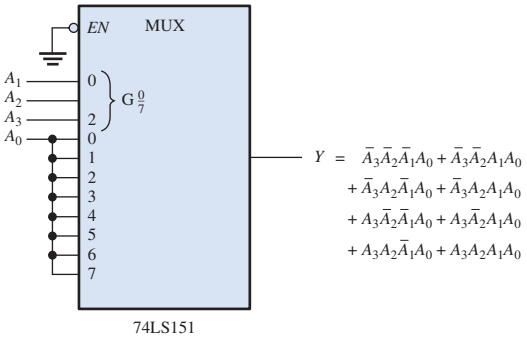
6–15. D_0: $S_3 = 0, S_2 = 0, S_1 = 0, S_0 = 0$
D_4: $S_3 = 0, S_2 = 1, S_1 = 0, S_0 = 0$
D_8: $S_3 = 1, S_2 = 0, S_1 = 0, S_0 = 0$
D_{13}: $S_3 = 1, S_2 = 1, S_1 = 0, S_0 = 1$

6–16. Veja a Figura 6–100



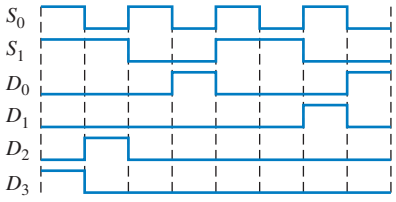
▶ FIGURA 6–100

6–17. Veja a Figura 6–101.



▶ FIGURA 6–101

6–18. Veja a Figura 6–102.



► FIGURA 6–101

AUTOTESTE

1. (a) **2.** (b) **3.** (c) **4.** (a) **5.** (d) **6.** (b) **7.** (c) **8.** (b)
2. (a) **10.** (d) **11.** (c) **12.** (f)

7

LATCHES, FLIP-FLOPS E TEMPORIZADORES

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Usar portas lógicas para construir latches básicos
- Explicar a diferença entre um latch S-R e um latch D
- Reconhecer a diferença entre um latch e um flip-flop
- Explicar as diferenças entre os flip-flops S-R, D e J-K
- Entender o significado dos atrasos de propagação, tempo de *setup* (preparação), tempo de *hold* (manutenção), freqüência máxima de operação, largura mínima de pulso de clock e dissipação de potência em aplicações de flip-flops
- Usar flip-flops em aplicações básicas

- Explicar em que diferem os monoestáveis redisparáveis e não-redisparáveis
- Configurar um temporizador 555 para operar como um multivibrador astável ou um monoestável
- Fazer análise de defeito em circuitos básicos com flip-flop

TERMOS IMPORTANTES

- Latch
- Biestável
- SET
- RESET
- Clock
- Flip-flop disparado por borda
- Síncrono
- Flip-flop D
- Flip-flop J-K
- T (*toggle*)
- *Preset*
- *Clear*
- Tempo de atraso de propagação
- Tempo de *setup*
- Tempo de *hold*
- Dissipação de potência
- Monoestável
- Temporizador
- Astável

INTRODUÇÃO

Este capítulo começa o estudo dos fundamentos da lógica seqüencial. Aqui são abordados os dispositivos lógicos biestável, monoestável e astável, denominados multivibradores. Duas categorias de dispositivos biestáveis são o latch e o flip-flop. Os dispositivos biestáveis têm dois estados estáveis, chamados de SET e RESET; tornando-os úteis como dispositivos de armazenamento. A diferença básica entre latches e flip-flops é a forma com que eles são comutados de um estado para o outro. O flip-flop é um bloco construtivo básico para contadores, registradores e outras lógicas de controle seqüencial e é usado em certos tipos de memórias. O multivibrador monoestável tem apenas um estado estável. Um monoestável produz um único pulso de largura controlada quando ativado ou disparado. O multivibrador astável não tem estado estável e é usado principalmente como oscilador, que é um gerador de forma de onda auto-sustentado. Os geradores de pulsos são usados como fontes para formas de onda de temporização em sistemas digitais.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

74XX74 74XX279 74XX122
555 74121 74XX75
74XX112

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

Esse tópico de Aplicações em Sistemas Digitais continua com o sistema de controle de semáforo do Capítulo 6. O foco neste capítulo é a parte do circuito de temporização do sistema que produz o clock, o intervalo de tempo longo para as luzes vermelha e verde e o intervalo de tempo curto para a luz de atenção (amarela). O clock é usado como o sinal de temporização do sistema básico para o avanço da lógica seqüencial através dos seus estados. A lógica seqüencial será desenvolvida no Capítulo 8.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

7-1 LATCHES

O **latch** é um tipo de dispositivo de armazenamento temporário que tem dois estados estáveis (biestável) e é normalmente colocado numa categoria separada dos flip-flops. Os latches são similares aos flip-flops porque eles são dispositivos biestáveis que podem permanecer em um dos dois estados estáveis usando uma configuração de realimentação, na qual as saídas são conectadas de volta às entradas opostas. A principal diferença entre os latches e os flip-flops é o método usado para a mudança de estado deles.

Ao final do estudo desta seção você deverá ser capaz de:

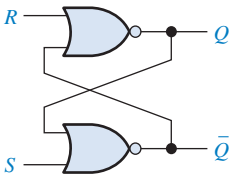
■ Explicar a operação de um latch S-R básico ■ Explicar a operação de um latch S-R implementado com portas lógicas ■ Explicar a operação de um latch D implementado com portas lógicas ■ Implementar um latch S-R ou D com portas lógicas ■ Descrever os CIs 74LS279 e 74LS75 (quatro latches)

NOTA: COMPUTAÇÃO

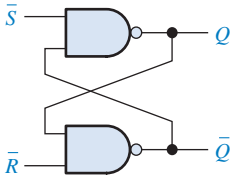
Os latches são algumas vezes usados em sistemas de computador para a multiplexação de dados num barramento. Por exemplo, os dados que entram no computador a partir de uma fonte externa têm que compartilhar o barramento de dados com os dados de outras fontes. Quando o barramento de dados se torna indisponível para uma fonte externa, o dado existente tem que ser temporariamente armazenado e os latches colocados entre a fonte externa e o barramento de dados podem ser usados para esse fim. Quando o barramento de dados está indisponível para a fonte externa, os latches têm que ser desconectados do barramento usando um método conhecido por tristate. Quando o barramento de dados se torna disponível, os dados externos passam através dos latches, originando assim a denominação de *latch transparente*. O latch D implementado com portas realiza essa função porque quando ele está habilitado, o dado em sua entrada aparece na saída como se existisse uma conexão direta através dele. O dado na entrada é armazenado logo que o latch seja desabilitado.

O Latch S-R

Um latch é um tipo de dispositivo lógico **biestável** ou **multivibrador**. Um latch S-R (SET-RESET) com entrada ativa em nível ALTO é formado com duas portas NOR tendo acoplamento cruzado, conforme mostra a Figura 7–1(a); um latch $\overline{S}$-$\overline{R}$ com entrada ativa em nível BAIXO é formado com duas portas NAND tendo acoplamento cruzado, conforme mostra a Figura 7–1(b). Observe que a saída de cada porta está conectada à entrada da porta oposta. Isso produz uma **realimentação** regenerativa que é características de todos os latches e flip-flops.



(a) Latch S-R com entrada ativa em nível ALTO.



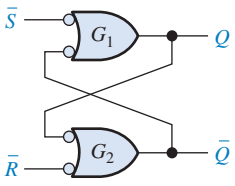
(b) Latch $\overline{S}$-$\overline{R}$ com entrada ativa em nível BAIXO.

▲ **FIGURA 7–1**

Duas versões de latches S-R (SET e RESET). Abra o arquivo F07-01 e verifique a operação dos dois latches.

Para explicar a operação do latch, usaremos o latch de portas NAND na Figura 7–1(b). Esse latch é redesenhado na Figura 7–2 com o símbolo equivalente da OR negativa usado para as portas NAND. Isso é feito porque os níveis BAIXOs nas linhas $\overline{S}$ e $\overline{R}$, são as entradas de ativação.

O latch mostrado na Figura 7–2 tem duas entradas, $\overline{S}$ e $\overline{R}$, e duas saídas Q e $\overline{Q}$. Vamos iniciar considerando que as duas entradas e a saída Q estejam em nível ALTO. Como a saída Q é conectada de volta na entrada da porta G_2 e a entrada $\overline{R}$ é nível ALTO, a saída de G_2 tem que ser nível BAIXO. Essa saída em nível BAIXO é acoplada de volta na entrada da porta G_1, garantindo que sua saída seja nível ALTO.



▶ **FIGURA 7–2**

Latch $\overline{S}$-$\overline{R}$ com portas OR negativa que equivale ao de portas NAND na Figura 7–1(b).

Quando a saída Q for nível ALTO, o latch está no estado **SET**. Ele permanece nesse estado indefinidamente até que um nível BAIXO seja temporariamente aplicado na entrada $\overline{R}$. Com um nível BAIXO na entrada $\overline{R}$ e um nível ALTO na entrada $\overline{S}$, a saída da porta G_2 é forçada para o ní-

vel ALTO. Esse nível ALTO na saída $\overline{Q}$ é acoplado de volta para uma entrada da porta G_1 e como a entrada $\overline{S}$ é nível ALTO a saída de G_1 vai para o nível BAIXO. Esse nível BAIXO na saída Q é então acoplado de volta para uma entrada de G_2, garantido que a saída $\overline{Q}$ permaneça em nível ALTO mesmo quando o nível BAIXO na entrada $\overline{R}$ é removido. Quando a saída Q é nível BAIXO, o latch está no estado de **RESET**. Agora o latch permanece indefinidamente no estado de RESET até que um nível BAIXO seja aplicado na entrada $\overline{S}$.

Um latch pode permanecer em um dos seus dois estados, SET ou RESET.

Em operação normal, as saídas de um latch são sempre complementares uma em relação a outra.

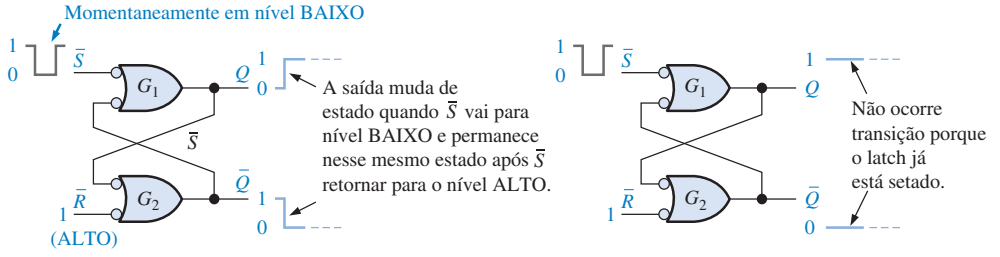
Quando Q for nível ALTO, $\overline{Q}$ será nível BAIXO e quando Q for nível BAIXO, $\overline{Q}$ será nível ALTO.

Uma condição inválida de um latch $\overline{S}$-$\overline{R}$ com entrada ativa em nível BAIXO ocorre quando níveis BAIXOs são aplicados em $\overline{S}$ e $\overline{R}$ ao mesmo tempo. Enquanto o nível BAIXO for mantido simultaneamente nas duas entradas, as saídas Q e $\overline{Q}$ são forçadas para nível ALTO, violando assim a operação básica de complementaridade das saídas. Além disso, se os níveis BAIXOs são liberados (desativados) simultaneamente, as duas saídas tentarão ir para nível BAIXO. Como sempre existe alguma pequena diferença no tempo de atraso de propagação das portas, uma das portas prevalecerá na transição para o estado de saída de nível BAIXO. Por sua vez, essa força a saída da porta mais lenta a permanecer em nível ALTO. Nessa situação, não podemos prever com certeza o próximo estado do latch.

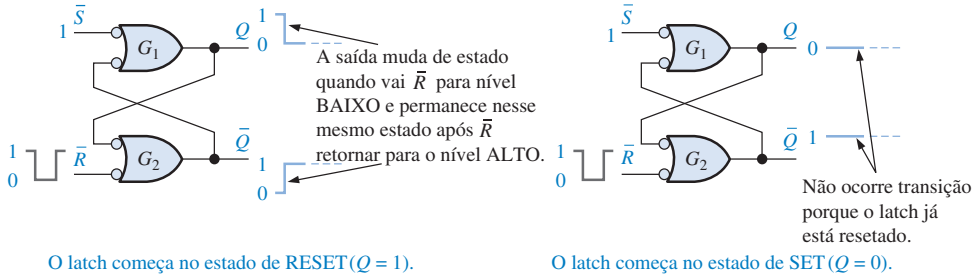
SET significa que a saída Q é nível ALTO.

A Figura 7–3 ilustra a operação do latch $\overline{S}$-$\overline{R}$ com entrada ativa em nível BAIXO para cada uma das quatro combinações possíveis de níveis nas entradas. (As três primeiras combinações são válidas, porém a última não). A Tabela 7–1 resume a operação lógica na forma de tabela-verdade.

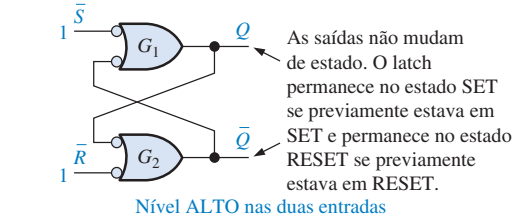
RESET significa que a saída Q é nível BAIXO.



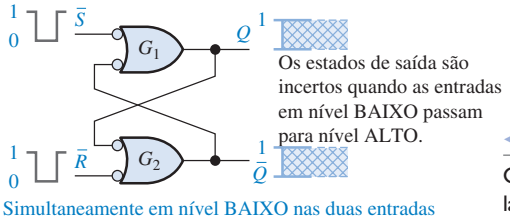
(a) Duas possibilidades para a operação SET



(b) Duas possibilidades para a operação RESET



(c) Condição sem alteração



(d) Condição inválida

◄ FIGURA 7–3

Os três modos de operação do latch $\overline{S}$-$\overline{R}$ básico (SET, RESET, repouso) e a condição inválida.

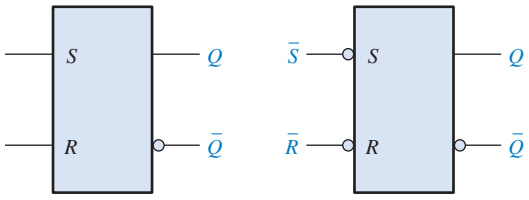
A operação do latch de portas NOR com entradas ativas em nível ALTO mostrado na Figura 7–1(a) é similar mas requer o uso de níveis lógicos opostos.

▶ TABELA 7–1

Tabela-verdade para um latch $\overline{S}$-$\overline{R}$ com entradas ativas em nível BAIXO

ENTRADAS		SAÍDAS		COMENTÁRIOS
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
1	1	NC	NC	Repouso. O latch permanece no estado atual.
0	1	1	0	Latch no estado SET.
1	0	0	1	Latch no estado RESET.
0	0	1	1	Condição inválida.

Os símbolos lógicos para os latches com entradas ativas em nível ALTO e entradas ativas em nível BAIXO são mostrados na Figura 7–4.



(a) Latch S-R com entradas ativas em nível ALTO.

(b) Latch $\overline{S}$-$\overline{R}$ com entradas ativas em nível BAIXO.

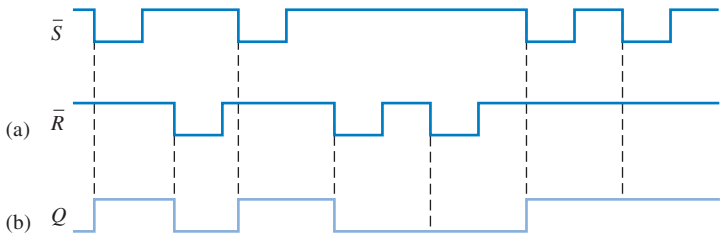
▶ FIGURA 7–4

Símbolos lógicos para os latches S-R e $\overline{S}$-$\overline{R}$.

O Exemplo 7–1 ilustra como um latch $\overline{S}$-$\overline{R}$ com entradas ativas em nível BAIXO responde às condições das entradas. Pulsos de nível BAIXO são aplicados em cada entrada numa certa seqüência e a forma de onda da saída Q resultante é observada. A condição $\overline{S} = 0$, $\overline{R} = 0$ é evitada porque resulta num modo de operação inválido sendo a principal desvantagem de qualquer latch do tipo SET-RESET.

EXEMPLO 7–1

Se as formas de onda $\overline{S}$ e $\overline{R}$ mostradas na Figura 7–5(a) são aplicadas nas entradas do latch visto na Figura 7–4(b), determine a forma de onda observada na saída Q. Considere que Q está inicialmente em nível BAIXO.



▲ FIGURA 7–5

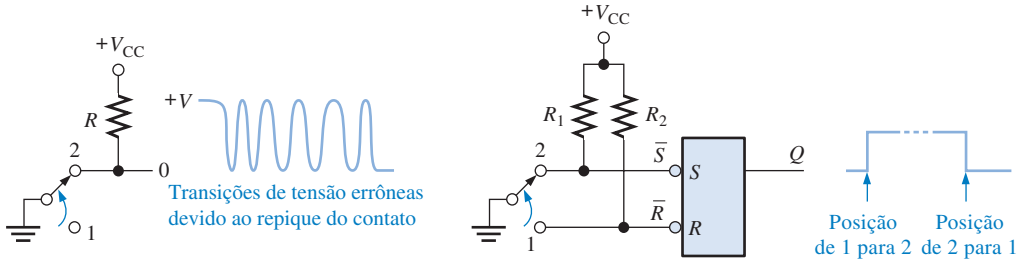
Solução Veja a Figura 7–5(b).

Problema relacionado* Determine a saída Q de um latch S-R com entradas ativas em nível ALTO se as formas de onda mostradas na Figura 7–5(a) forem invertidas e aplicadas nas entradas.

* As respostas estão no final do capítulo.

Uma Aplicação

Um Latch Usado como Eliminador de Trepidação de Contato Um bom exemplo de uma aplicação de um latch $\overline{S}$-$\overline{R}$ é na eliminação do "repique" (*bounce*) do contato de uma chave mecânica. Quando o pólo de uma chave comuta fazendo fechar o contato, este contato vibra fisicamente ou repica várias vezes antes de finalmente estabelecer um contato firme. Embora esses repiques sejam de durações muito curtas, eles produzem spikes de tensão que freqüentemente são inaceitáveis em sistemas digitais. Essa situação é ilustrada na Figura 7–6(a).



(a) Repique do contato de uma chave

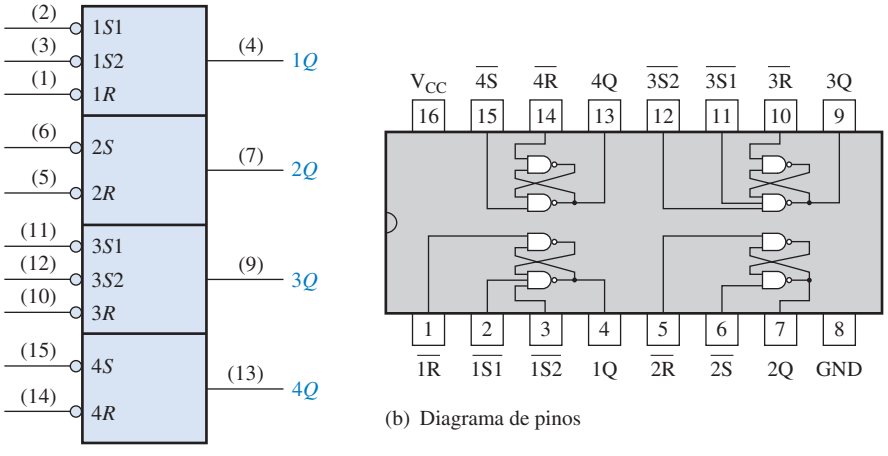
(b) Circuito eliminador de repique de contato

◀ **FIGURA 7–6**
Uso do latch $\overline{S}$-$\overline{R}$ para eliminar o repique do contato de uma chave.

Um latch $\overline{S}$-$\overline{R}$ pode ser usado para eliminar os efeitos do repique de uma chave como mostra a Figura 7–6(b). A chave está normalmente na posição 1, mantendo a entrada $\overline{R}$ em nível BAIXO e o latch no estado de RESET. Quando a chave é comutada para a posição 2, $\overline{R}$ vai para nível ALTO por causa do resistor de pull-up para V_{CC} e $\overline{S}$ vai para nível BAIXO no primeiro contato. Embora $\overline{S}$ permaneça em nível BAIXO por um intervalo muito curto de tempo antes de a chave repicar, isso é suficiente para *setar* o latch. Qualquer spike de tensão posterior na entrada $\overline{S}$ em função do repique da chave, não afeta o latch, que permanece *setado*. Observe que a saída Q do latch fornece uma transição "limpa" (única) do nível BAIXO para o alto, eliminando assim os spikes de tensão provocados pelo repique do contato. De forma similar, uma transição limpa do nível ALTO para o BAIXO ocorre quando a chave comuta de volta para a posição 1.

LATCH SET-RESET (74LS279)

O CI 74LS279 contém quatro latches $\overline{S}$-$\overline{R}$ representado pelo diagrama lógico visto na Figura 7–7(a) e o diagrama de pinos na parte (b). Observe que dois dos latches têm duas entradas $\overline{S}$.



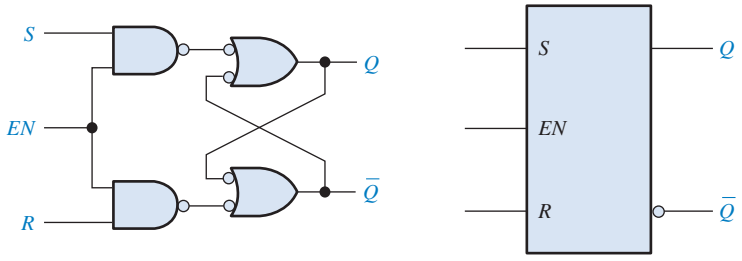
(a) Diagrama lógico

(b) Diagrama de pinos

▲ **FIGURA 7–7**
CI contendo quatro latches $\overline{S}$-$\overline{R}$ (74LS279).

Um Latch S-R Controlado

Um latch controlado necessita de uma entrada de habilitação, *EN* (a letra *G* também é usada para indicar uma entrada de habilitação). O diagrama lógico e o símbolo lógico para um latch S-R controlado são mostrados na Figura 7–8. As entradas *S* e *R* controlam o estado para o qual o latch irá quando um nível ALTO é aplicado na entrada *EN*. O latch não mudará de estado até que EN seja nível ALTO; porém enquanto essa entrada permanecer em nível ALTO, a saída é determinada pelos estados das entradas *S* e *R*. Nesse circuito o estado inválido ocorre quando *S* e *R* forem simultaneamente nível ALTO.

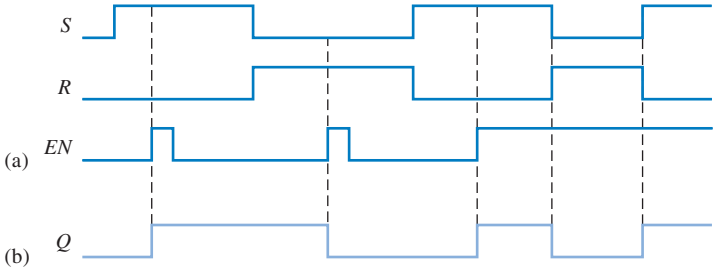


(a) Diagrama lógico (b) Símbolo lógico

▶ **FIGURA 7–8**
Um latch S-R controlado.

EXEMPLO 7–2

Determine a forma de onda da saída *Q* se as entradas mostradas na Figura 7–9(a) forem aplicadas no latch S-R controlado que está inicialmente resetado.



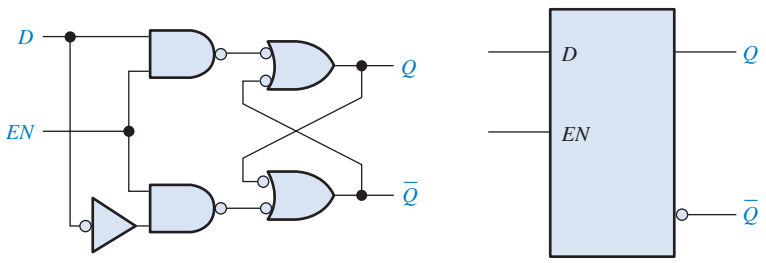
▲ **FIGURA 7–9**

Solução A forma de onda *Q* é mostrada na Figura 7–9(b). Quando *S* for nível ALTO e *R* for nível BAIXO, um nível ALTO na entrada *EN* seta o latch. Quando *S* for nível BAIXO e *R* for nível ALTO, um nível ALTO na entrada *EN* reseta o latch.

Problema relacionado Determine a saída *Q* de um latch S-R controlado se as entradas *S* e *R* mostradas na Figura 7–9(a) forem invertidas.

Latch D controlado

Um outro tipo de latch controlado é denominado de latch *D*. Esse difere do latch S-R por ter apenas uma entrada além de *EN*. A entrada mencionada é denominada de entrada *D* (dado). A Figura 7–10 contém o diagrama lógico e o símbolo lógico de um latch *D*. Quando a entrada *D* for nível ALTO e a entrada *EN* for nível ALTO, o latch será setado. Quando a entrada *D* for nível BAIXO e a entrada *EN* for nível ALTO, o latch será resetado. Dito de uma outra forma, a saída *Q* segue a entrada *D* quando *EN* for nível ALTO.

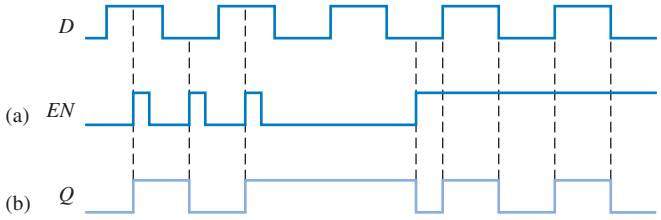


(a) Diagrama lógico (b) Símbolo lógico

◄ FIGURA 7–10
Um latch D controlado.

EXEMPLO 7–3

Determine a forma de onda da saída Q se as entradas mostradas na Figura 7–11(a) são aplicadas num latch D controlado, o qual inicialmente está *resetado*.



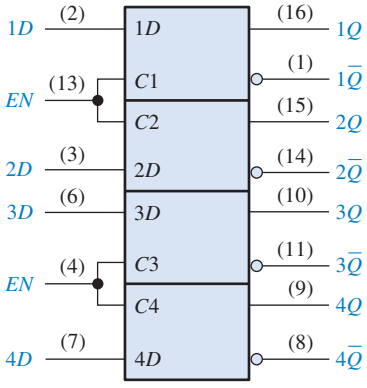
► FIGURA 7–11

Solução A forma de onda Q é mostrada na Figura 7–11(b). Quando D for nível ALTO e EN for nível ALTO, Q vai para o nível ALTO. Quando D for nível BAIXO e EN for nível ALTO, Q vai para nível BAIXO. Quando EN for nível BAIXO, o estado do latch não é afetado pela entrada D.

Problema relacionado Determine a saída Q do latch D controlado se a entrada D vista na Figura 7–11 for invertida.

LATCH D (74LS75)

Um exemplo de um latch D controlado é o CI 74LS75 representado pelo símbolo lógico mostrado na Figura 7–12(a). Esse dispositivo tem quatro latches. Observe que cada entrada *EN* ativa em nível ALTO é compartilhada por dois latches e é indicada como uma entrada de controle (C). A tabela-verdade para cada latch é mostrada na Figura 7–12(b). O X na tabela-verdade representa uma condição "don't care". Nesse caso, quando a entrada *EN* for nível BAIXO, não importa o que tem na entrada D porque as saídas não são afetadas permanecendo nos estados em que estavam.



(a) Símbolo lógico

Entradas		Saídas		
D	EN	Q	$\overline{Q}$	Comentários
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	Repouso

Nota: Q_0 é o nível da saída anterior antes que as condições de entrada fossem estabelecidas.

(b) Tabela-verdade (cada latch)

◄ FIGURA 7–12
Quatro latches D controlados (74LS75).

SEÇÃO 7–1 REVISÃO
As respostas estão no final do capítulo.

1. Faça uma lista com três tipos de latches.
2. Desenvolva a tabela-verdade para o latch S-R com entradas ativas em nível ALTO mostrado na Figura 7–1(a).
3. Qual é a saída *Q* de um latch D quando *EN* = 1 e *D* = 1?

7-2 FLIP-FLOPS DISPARADOS POR BORDA

Os flip-flops são dispositivos biestáveis síncronos, também conhecidos como *multivibradores biestáveis*. Nesse caso, o termo síncrono significa que a saída muda de estado apenas no memento especificado pela entrada de disparo denominada de **clock** (CLK), a qual é indicada como uma entrada de controle (*C*); ou seja, as mudanças na saída ocorrem em sincronismo com o clock.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir *clock* ■ Definir *flip-flop disparado por borda* ■ Explicar a diferença entre um flip-flop e um latch ■ Identificar um flip-flop disparado por borda pelo seu símbolo lógico ■ Discutir a diferença entre um flip-flop disparado por borda positiva e um disparado por borda negativa ■ Discutir e comparar a operação de flip-flops S-R, D e J-K disparados por borda e explicar as diferenças entre as suas tabelas-verdade ■ Discutir as entradas assíncronas de um flip-flop ■ Descrever os CIs de flip-flops 74HC74 e 74HC112

O indicador de entrada dinâmica ▷ significa que o flip-flop muda de estado apenas na borda de um pulso de clock.

Um **flip-flop disparado por borda** muda de estado na borda positiva (borda de subida) ou na borda negativa (borda de descida) do pulso de clock e é sensível às entradas apenas nas transições do clock. Essa seção aborda três tipos de flip-flops disparados por borda: S-R, D e J-K. Embora o flip-flop S-R não esteja disponível na forma de CI, ele é a base dos flip-flops D e J-K. O símbolo lógico para todos esses flip-flops são mostrados na Figura 7–13. Observe que cada tipo pode ser disparado na borda positiva (sem o pequeno círculo na entrada *C*) ou disparado na borda negativa (com o pequeno círculo na entrada *C*). O detalhe do símbolo lógico na identificação de um flip-flop disparado por borda é o pequeno triângulo dentro do bloco na entrada de clock (*C*). Esse triângulo é denominado de *indicador de entrada dinâmica*.

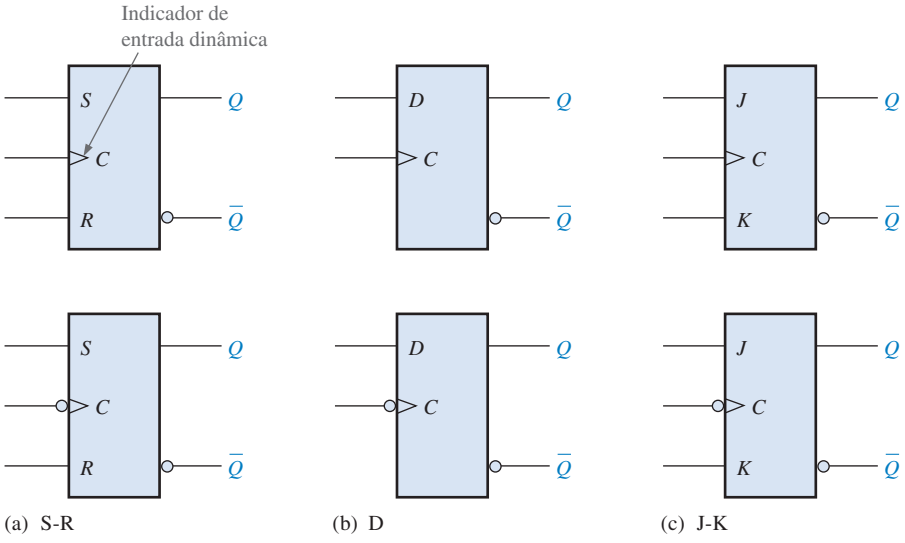


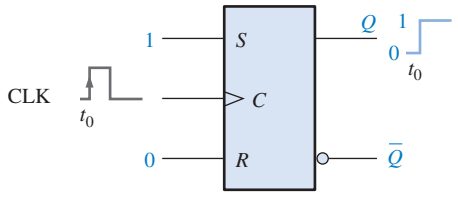
FIGURA 7–13
Símbolos lógicos de flip-flops disparados por borda (parte superior: disparo por borda positiva; parte inferior: disparo por borda negativa.

Flip-Flop S-R Disparado por Borda

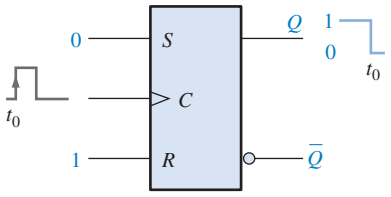
As entradas *S* e *R* do **flip-flop S-R** são denominadas entradas **síncronas** porque os dados nessas entradas são transferidos para a saída do flip-flop apenas na borda de disparo do pulso de clock. Quando *S* for nível ALTO e *R* for nível BAIXO, a saída *Q* vai para nível ALTO na borda de disparo do pulso de clock, estando o flip-flop setado. Quando *S* for nível BAIXO e *R* for nível ALTO, a saída *Q* vai para o nível BAIXO na borda de disparo do pulso de clock, estando o flip-flop resetado. Quando as entradas *S* e *R* estiverem em nível BAIXO, a saída não muda de estado permanecendo no estado anterior. Uma condição inválida existe quando *S* e *R* forem nível ALTO.

Um flip-flop S-R não pode ter as entradas *S* e *R* em nível ALTO ao mesmo tempo.

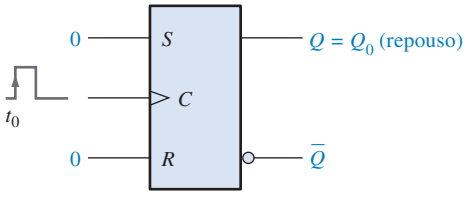
Essa operação básica de um flip-flop disparado por borda positiva é ilustrada na Figura 7–14, sendo a Tabela 7–2 a tabela-verdade para esse tipo de flip-flop. Lembre-se, *o flip-flop não pode mudar de estado exceto na borda de disparo de um pulso de clock*. As entradas *S* e *R* podem ser alteradas em qualquer momento que a entrada de clock for nível BAIXO ou ALTO (exceto por um intervalo muito curto em torno da transição de disparo do clock) sem afetar a saída.



(a) Com $S = 1$ e $R = 0$ o flip-flop é setado na borda positiva do clock (caso já esteja setado, permanecerá setado).



(b) Com $S = 0$ e $R = 1$ o flip-flop é resetado na borda positiva do clock (caso já esteja resetado, permanecerá resetado).



(c) Com $S = 0$ e $R = 0$ o flip-flop não muda de estado (caso esteja setado, permanecerá setado; caso esteja resetado, permanecerá resetado).

▲ **FIGURA 7–14**

Operação de um flip-flop S-R disparado por borda positiva.

NOTA: COMPUTAÇÃO

As memórias semicondutoras nos computadores consistem de um grande número de células individuais. Cada célula de armazenamento mantém um nível 1 ou um nível 0. Um tipo de memória é a Memória de Acesso Aleatório Estática (SRAM) que usa flip-flops para as células de armazenamento porque um flip-flop retém qualquer um dos dois estados indefinidamente enquanto a alimentação cc estiver aplicada, daí o termo *estática*. Esse tipo de memória é classificado como memória *volátil* porque todos os dados armazenados são perdidos quando a alimentação é desligada. Um outro tipo de memória, a Memória de Acesso Aleatório Dinâmica ou DRAM, usa capacitância em vez de flip-flops como elemento básico de armazenamento e tem que ser renovada (*refresh*) periodicamente para manter o dado armazenado.

◀ **TABELA 7–2**

Tabela-verdade para um flip-flop S-R disparado por borda positiva

ENTRADAS			SAÍDAS		
S	*R*	CLK	*Q*	$\overline{Q}$	COMENTÁRIOS
0	0	X	Q_0	$\overline{Q}_0$	Repouso
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	Inválido

↑ = transição do clock do nível BAIXO para o nível ALTO

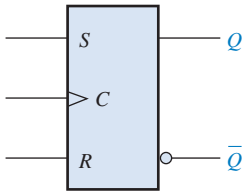
X = irrelevante ("don't care")

Q_0 = nível de saída antes da transição do clock

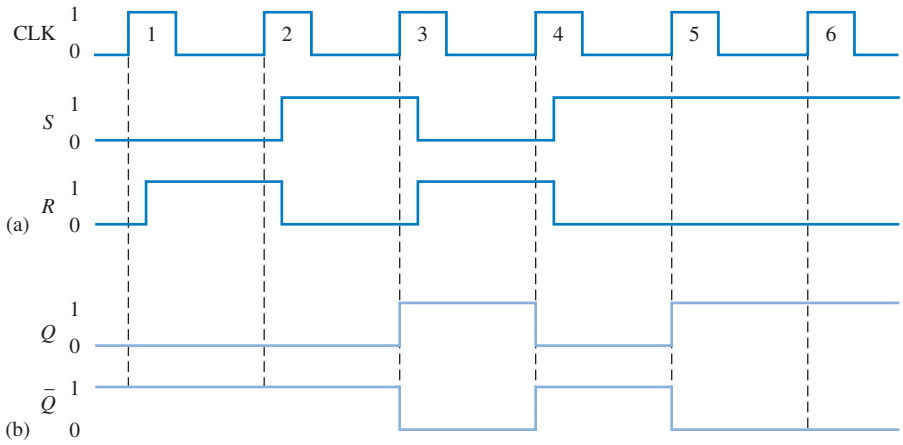
A operação e a tabela-verdade para um flip-flop S-R disparado por borda negativa são as mesmas que para um dispositivo disparado por borda positiva exceto que a borda de descida do pulso de clock é a borda de disparo.

EXEMPLO 7–4

Determine as formas de onda das saídas Q e $\overline{Q}$ do flip-flop mostrado na Figura 7–15 para as entradas S, R e CLK mostradas na Figura 7–16(a). Considere que o flip-flop disparado por borda positiva esteja inicialmente resetado.



▶ FIGURA 7–15



▲ FIGURA 7–16

Solução

1. No momento do pulso 1 de clock, S é nível BAIXO e R é nível BAIXO, assim Q não muda de estado.
2. No momento do pulso 2 de clock, S é nível BAIXO e R é nível ALTO, assim Q permanece em nível BAIXO (RESET).
3. No momento do pulso 3 de clock, S é nível ALTO e R é nível BAIXO, assim Q vai para nível ALTO (SET).
4. No momento do pulso 4 de clock, S é nível BAIXO e R é nível ALTO, assim Q vai para nível BAIXO (RESET).
5. No momento do pulso 5 de clock, S é nível ALTO e R é nível BAIXO, assim Q vai para nível ALTO (SET).
6. No momento do pulso 6 de clock, S é nível ALTO e R é nível BAIXO, assim Q permanece em nível ALTO.

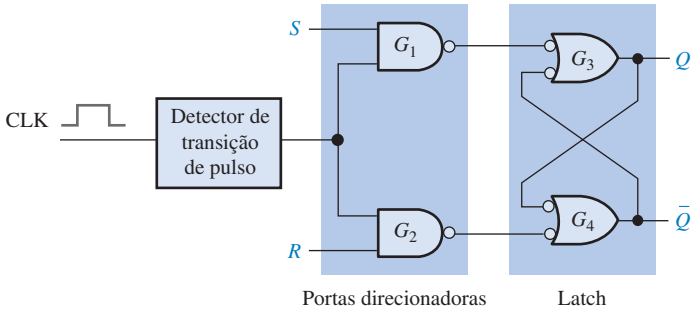
Uma vez determinada a saída Q, $\overline{Q}$ é facilmente determinada visto que ela é simplesmente o complemento da saída Q. As formas de onda resultantes para Q e $\overline{Q}$ são mostradas na Figura 7–16(b) para as formas de onda de entrada dadas na parte (a).

Problema relacionado Determine Q e $\overline{Q}$ para as entradas S e R dadas de acordo com a Figura 7–16(a) se o flip-flop for um dispositivo disparado por borda negativa.

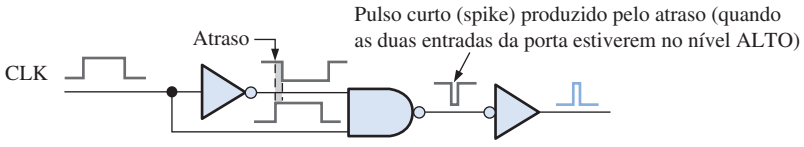
Um Método de Disparo por Borda

Uma implementação simplificada de um flip-flop S-R disparado por borda é ilustrada na Figura 7–17(a) e é usada para demonstrar o conceito de disparo por borda. Essa abordagem tomando como referência o flip-flop S-R não implica que ele seja o tipo mais importante. Na realidade, os flip-flops D e J-K são comercializados na forma de CI e são mais amplamente usados que o tipo

S-R. Entretanto, é importante entender o método para o tipo S-R porque os flip-flops D e J-K são derivados do flip-flop S-R. Observe que o flip-flop S-R difere do latch S-R controlado apenas no fato de que o primeiro tem um detector de transição de pulso.



(a) Um diagrama lógico simplificado para um flip-flop S-R disparado por borda positiva.



(b) Um tipo de detector de transição de pulso.

◀ FIGURA 7–17
Disparo por borda.

Um tipo básico de detector de transição de pulso é mostrado na Figura 7–17(b). Como podemos ver, existe um pequeno atraso numa entrada da porta NAND de forma que o pulso de clock invertido chega à entrada da porta alguns nanossegundos depois do pulso de clock original. Esse circuito produz um spike de duração muito curta na transição positiva do pulso de clock. Num flip-flop disparado por borda negativa, o pulso de clock é invertido primeiro, produzindo assim um spike na borda negativa.

O circuito dado na Figura 7–17 é dividido em duas seções, uma denominada Portas Direcionadoras e a outra denominada Latch. As portas direcionadoras encaminham, ou direcionam, o spike de clock para a entrada da porta G_3 ou para a entrada da porta G_4 dependendo do estado das entradas S e R. Para entender a operação desse flip-flop, comece admitindo que ele está no estado de RESET ($Q = 0$) e que as entradas S, R e CLK são todas nível BAIXO. Para essa condição, as saídas das portas G_1 e G_2 estão em nível ALTO. O nível BAIXO na saída Q é acoplado de volta numa entrada da porta G_4 tornando a saída $\overline{Q}$ nível ALTO. Como $\overline{Q}$ está em nível ALTO, as duas entradas da porta G_3 estão em nível ALTO (lembre-se que a saída da porta G_1 é nível ALTO), mantendo a saída Q em nível BAIXO. Se um pulso for aplicado na entrada CLK, as saídas das portas G_1 e G_2 permanecem em nível ALTO porque elas são desabilitadas pelo nível BAIXO presente nas entradas S e R; portanto, não há mudança no estado do flip-flop (ele permanece no estado resetado).

Agora vamos fazer com que S seja nível ALTO, deixando R em nível BAIXO e aplicando um pulso de clock. Como a entrada S para a porta G_1 é nível ALTO, a saída da porta G_1 vai para nível BAIXO por um intervalo de tempo muito curto (spike) quando CLK vai para nível ALTO, fazendo com que a saída Q vá para nível ALTO. Estando agora as duas entradas da porta G_4 em nível ALTO (lembre-se que a saída de G_2 é nível ALTO porque R é nível BAIXO), forçando a saída $\overline{Q}$ para nível BAIXO. Esse nível BAIXO em $\overline{Q}$ é acoplado de volta para uma entrada da porta G_3, garantido que a saída Q seja mantida em nível ALTO. O flip-flop agora está no estado SET. A Figura 7–18 ilustra as transições de nível lógico que acontecem dentro do flip-flop para essa condição.

Em seguida vamos tornar S nível BAIXO e R nível ALTO e aplicar um pulso de clock. Como a entrada R agora é nível ALTO, a borda positiva do clock produz um spike negativo na saída da porta G_2, fazendo com que a saída $\overline{Q}$ vá para nível ALTO. Por causa desse nível ALTO em $\overline{Q}$, as duas entradas da porta G_3 estão em nível ALTO (lembre-se, a saída da porta G_1 é nível ALTO por causa do nível BAIXO em S), forçando a saída Q para o nível BAIXO. Esse nível BAIXO em Q é acoplado de volta numa das entradas da porta G_4, garantido que $\overline{Q}$ permaneça em nível ALTO. O flip-flop agora está no estado de RESET.

NOTA: COMPUTAÇÃO

Todas as operações lógicas que são realizadas com hardware também podem ser implementadas por software. Por exemplo, a operação de um flip-flop J-K pode ser realizada com instruções específicas de computador. Se dois bits forem usados para representar as entradas *J* e *K*, o computador não alteraria o bit de saída (representando a saída *Q*) para uma entrada 00, esse bit de saída seria *setado* (1) para uma entrada 10, o bit de saída seria *resetado* (0) para uma entrada 01 e o mesmo bit seria complementado para uma entrada 11. Embora não possa ser comum o uso de um computador para simular um flip-flop, a questão é que todas as operações de hardware podem ser simuladas usando software.

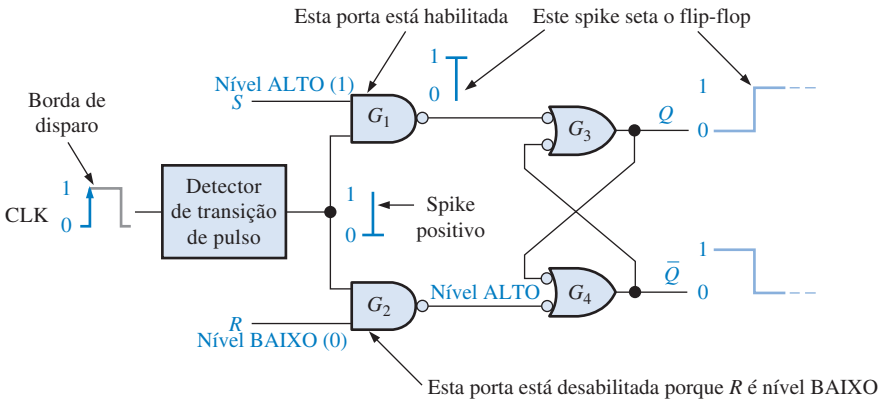


FIGURA 7–18

Flip-flop realizando uma transição do estado RESET para o estado SET na borda positiva do pulso de clock.

A Figura 7–19 ilustra as transições de nível lógico que ocorrem dentro do flip-flop para essa condição. Assim como com o latch controlado, uma condição inválida existe se um pulso de clock ocorrer quando as entrada *S* e *R* estiverem em nível ALTO ao mesmo tempo. Essa é a principal desvantagem do flip-flop S-R.

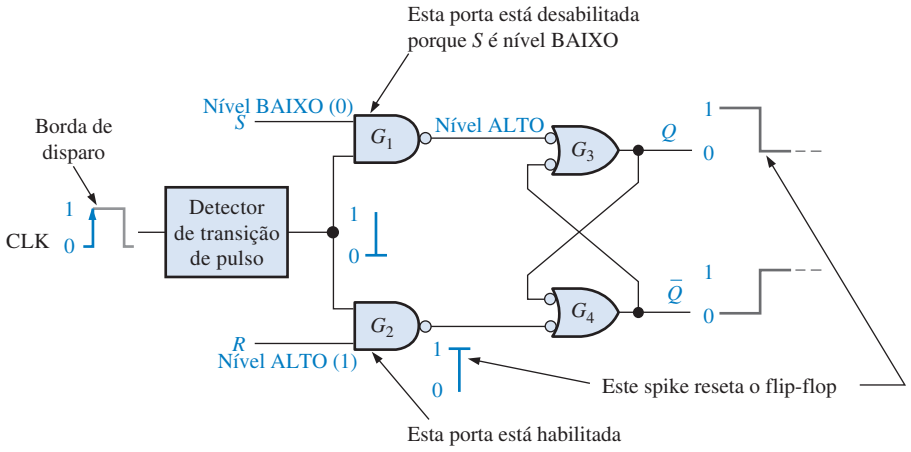


FIGURA 7–19

Flip-flop realizando uma transição do estado SET para o estado RESET na borda positiva do pulso de clock.

Flip-flop D Disparado por Borda

A saída *Q* de um flip-flop *D* considera o estado da entrada *D* na borda de disparo do clock.

O **flip-flop D** é usado quando um único bit de dado (1 ou 0) é para ser armazenado. A adição de um inversor num flip-flop S-R cria um flip-flop D básico, conforme a Figura 7–20, a qual mostra um tipo disparado por borda.

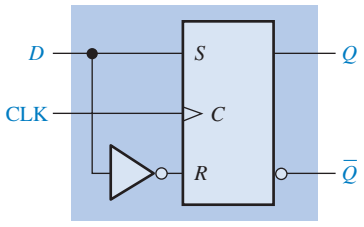


FIGURA 7–20

Um flip-flop D disparado por borda positiva construído a partir de um flip-flop S-R e um inversor.

Observe que o flip-flop na Figura 7–20 tem apenas uma entrada, a entrada *D*, além do clock. Caso exista um nível ALTO na entrada *D* quando um pulso de clock é aplicado, o flip-flop será setado sendo o nível ALTO na entrada *D* é armazenado pelo flip-flop na borda positiva do pul-

so de clock. Caso exista um nível BAIXO na entrada *D* quando o pulso de clock é aplicado, o flip-flop será resetado sendo o nível BAIXO na entrada *D* armazenado pelo flip-flop na borda de subida do pulso de clock. No estado SET o flip-flop armazena um nível 1 e no estado RESET ele armazena um nível 0.

A operação lógica do flip-flop D disparado por borda positiva é resumida na Tabela 7–3. A operação de um dispositivo disparado por borda negativa é evidentemente a mesma, exceto que o disparo ocorre na borda de descida do pulso de clock. Lembre-se, a saída *Q* segue a entrada *D* na borda ativa ou de disparo do clock.

◀ TABELA 7–3

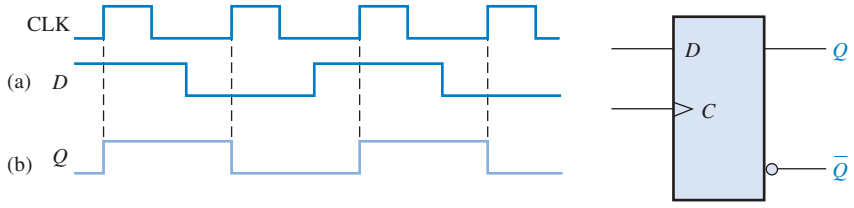
Tabela-verdade para um flip-flop D disparado por borda positiva

ENTRADAS		SAÍDAS		COMENTÁRIOS
D	CLK	*Q*	$\overline{Q}$	
1	↑	1	0	SET (armazena um nível 1)
0	↑	0	1	RESET (armazena um nível 0)

↑ = Transição do clock do nível BAIXO para o ALTO.

EXEMPLO 7–5

Dadas as formas de onda na Figura 7–21(a) para a entrada *D* e o clock, determine a forma de onda na saída *Q* se o flip-flop começar *resetado*.



▲ FIGURA 7–21

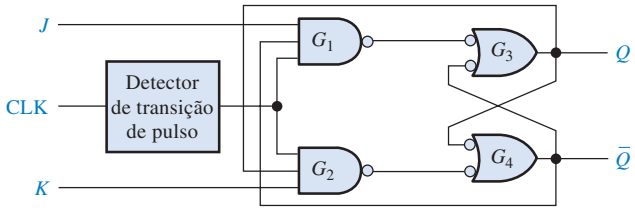
Solução A saída *Q* passa para o estado da entrada *D* no instante da transição positiva do clock. A saída resultante é mostrada na Figura 7–21(b).

Problema relacionado Determine a saída *Q* para o flip-flop D se a entrada *D* mostrada na Figura 7–21(a) for invertida.

Flip-flop J-K Disparado por Borda

O **flip-flop J-K** é versátil e é um tipo de flip-flop amplamente usado. O funcionamento de um flip-flop J-K é idêntico ao do flip-flop S-R nas condições de operação de SET, RESET e repouso. A diferença é que o flip-flop J-K não tem estado inválido como o flip-flop S-R.

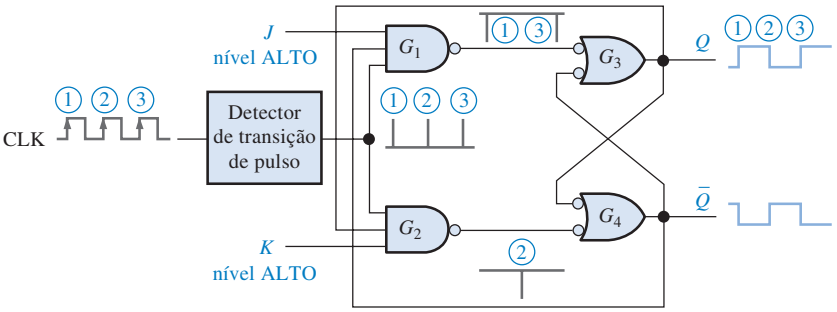
A Figura 7–22 mostra a lógica interna básica para um flip-flop J-K disparado por borda. Ele difere do flip-flop S-R disparado por borda em que a saída *Q* é conectada de volta na entrada da porta G_2 e a saída $\overline{Q}$ é conectada de volta na entrada da porta G_1. As duas entradas de controle são denominadas *J* e *K* em homenagem a Jack Kilby, inventor do circuito integrado. Um flip-flop J-K também pode ser do tipo disparado por borda negativa, caso no qual a entrada de clock é invertida.



▶ **FIGURA 7–22**

Um diagrama lógico simplificado para um flip-flop J-K disparado por borda positiva.

Vamos considerar que o flip-flop na Figura 7–23 esteja resetado e que a entrada *J* seja nível ALTO e que a entrada *K* seja nível BAIXO diferentemente do que é mostrado. Quando um pulso de clock ocorre, um spike de borda de subida indicado por $\overline{Q}$ passa através da porta G_1 porque $\overline{Q}$ é nível ALTO e *J* é nível ALTO. Isso faz com que a parte latch do flip-flop mude para o estado SET. O flip-flop agora está setado.



▶ **FIGURA 7–23**

Transições ilustrando a operação toggle (comutação) quando *J* = 1 e *K* = 1.

No modo toggle, um flip-flop J-K muda de estado a cada pulso de clock.

Se fizermos *J* nível BAIXO e *K* nível ALTO, o próximo spike de clock indicado por ¡ passará através da porta G_2 porque *Q* é nível ALTO e *K* é nível ALTO. Isso faz com que a porção latch do flip-flop mude para o estado RESET.

Se aplicarmos um nível BAIXO nas entradas *J* e *K*, o flip-flop permanecerá no atual estado quando ocorrer um pulso de clock. Um nível BAIXO nas entradas *J* e *K* resulta numa condição *sem mudança*.

Até agora, a operação lógica do flip-flop J-K é a mesma que a do tipo S-R para as condições de SET, RESET e sem mudança. A diferença na operação ocorre quando as entrada *J* e *K* estiverem em nível ALTO. Para entender isso, considere que o flip-flop esteja no estado de RESET. O nível ALTO em $\overline{Q}$ habilita a porta G_1, assim o spike de clock indicado por ¬ passa setando o flip-flop. Agora existe um nível ALTO em *Q*, permitindo que o próximo spike de clock passe através da porta G_2 e resete o flip-flop.

Como podemos ver, a cada spike de clock sucessivo, o flip-flop muda para o estado oposto. Esse modo é denominado operação **toggle** (comutação). A Figura 7–23 ilustra as transições quando o flip-flop está no modo toggle. Um flip-flop J-K conectado para o modo toggle é denominado algumas vezes de flip-flop *T*.

A Tabela 7–4 resume a operação lógica do flip-flop J-K na forma de tabela-verdade. Observe que não existe estado inválido como ocorre com o flip-flop S-R. Essa tabela verdade, para um dispositivo disparado por borda negativa, é idêntica a essa exceto que o flip-flop é disparado na borda de descida do pulso de clock.

▶ **TABELA 7–4**

Tabela-verdade para um flip-flop J-K disparado por borda positiva do clock

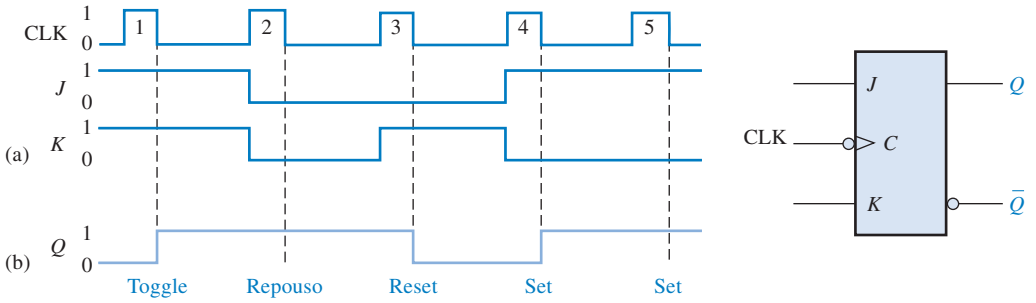
ENTRADAS			SAÍDAS		
J	*K*	CLK	*Q*	$\overline{Q}$	COMENTÁRIOS
0	0	↑	Q_0	$\overline{Q}_0$	Repouso
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Toggle

↑ = Transição do clock do nível BAIXO para o ALTO.

Q_0 = Nível da saída antes da transição do clock.

EXEMPLO 7–6

As formas de onda mostradas na Figura 7–24(a) são aplicadas nas entradas *J*, *K* e clock conforme indicado. Determine a saída *Q*, considerando que o flip-flop esteja inicialmente resetado.



▲ FIGURA 7–24

Solução

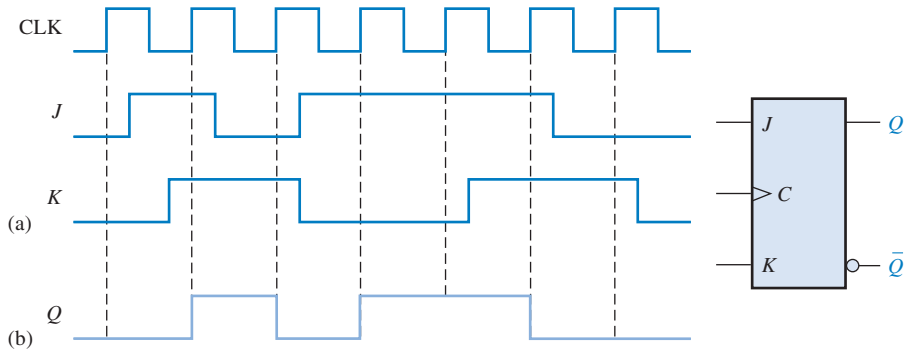
1. Primeiro, como esse flip-flop é disparado por borda negativa, conforme indicado pelo pequeno círculo na entrada de clock, a saída *Q* mudará apenas na borda negativa do pulso de clock.
2. No primeiro pulso de clock, *J* e *K* estão em nível ALTO; por ser o modo toggle, *Q* vai para nível ALTO.
3. No pulso de clock 2, existe uma condição de entrada que coloca o flip-flop no estado de repouso, mantendo *Q* no nível ALTO.
4. Quando ocorre o pulso de clock 3, *J* é nível BAIXO e *K* é nível ALTO, resultando na condição de RESET; assim *Q* vai para nível BAIXO.
5. No pulso de clock 4, *J* é nível ALTO e *K* é nível BAIXO, resultando na condição SET; assim *Q* vai para nível ALTO.
6. Uma condição SET ainda existe em *J* e *K* quando ocorre o pulso de clock 5, assim *Q* permanece em nível ALTO.

A forma de onda *Q* resultante é indicada na Figura 7–24(b).

Problema relacionado Determine a saída *Q* de um flip-flop J-K se as entrada *J* e *K* mostradas na Figura 7–24(a) forem invertidas.

EXEMPLO 7–7

As formas de onda vista na Figura 7–25(a) são aplicadas no flip-flop como mostrado. Determine a saída *Q* para o flip-flop começando no estado RESET.



▲ **FIGURA 7–25**

Solução A saída Q assume o estado determinado pelas entrada J e K na borda positiva (disparo por borda) do pulso de clock. Uma mudança em J ou K após a borda de disparo do clock não tem efeito na saída, conforme mostra a Figura 7–25(b).

Problema relacionado Troque entre si as entradas J e K e determine a saída Q resultante.

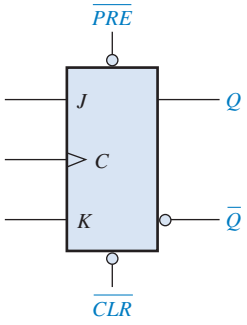
Entradas Assíncronas de Preset e Clear

Uma entrada *preset* ativa faz com que a saída Q seja nível ALTO (SET).

Uma entrada *clear* ativa faz com que a saída Q seja nível BAIXO (RESET).

Para os flip-flops discutidos, as entradas S-R, D e J-K são denominadas entradas síncronas porque os dados nessas entradas são transferidos para a saída do flip-flop apenas na borda de disparo do pulso de clock; ou seja, os dados são transferidos de forma sincronizada com o clock.

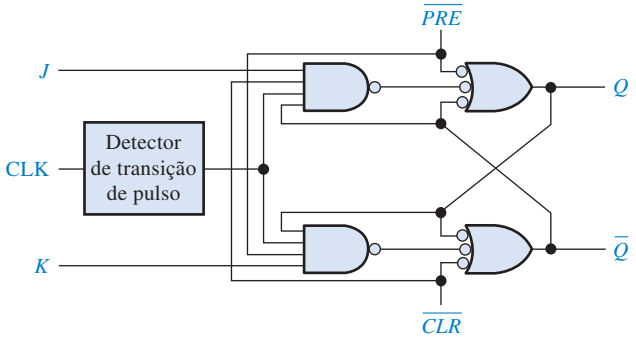
A maioria dos flip-flops em circuitos integrados também tem entradas **assíncronas**. Essas são entradas que afetam o estado do flip-flop *independente do clock*. Elas são normalmente denominadas **preset** (*PRE*) e **clear** (*CLR*), ou seta direto (S_D) e reseta direto (R_D) por alguns fabricantes. Um nível ativo na entrada preset irá setar o flip-flop e um nível ativo na entrada clear irá resetar o flip-flop. Um símbolo lógico para um flip-flop J-K com entradas preset e clear é mostrado na Figura 7–26. Essas entradas são ativas em nível BAIXO, conforme indicado pelos pequenos círculos. Essas entradas de preset e clear têm que ser mantidas em nível ALTO para a operação síncrona.



▶ **FIGURA 7–26**

Símbolo lógico para um flip-flop J-K com entradas preset e clear ativas em nível BAIXO.

A Figura 7–27 mostra o diagrama lógico para um flip-flop J-K disparado por borda em entradas preset ($\overline{PRE}$) e clear ($\overline{CLR}$). Essa figura ilustra basicamente como essas entradas funcionam. Como podemos ver, elas são conectadas de forma que o efeito delas se sobrepõem ao das entradas síncronas (*J*, *K* e clock).

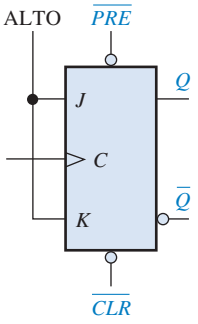


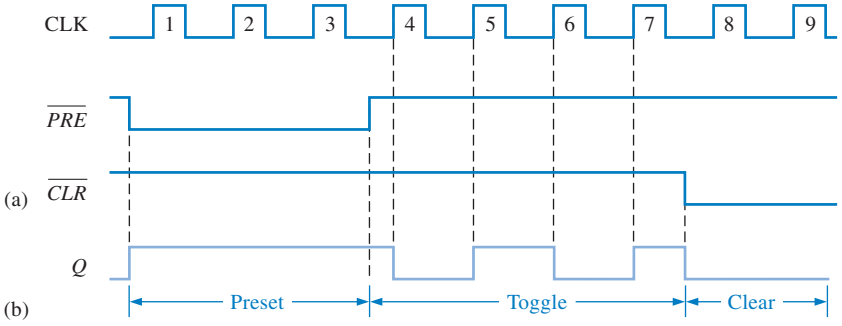
◀ FIGURA 7–27

Diagrama lógico para um flip-flop J-K básico com entradas preset e clear ativas em nível BAIXO.

EXEMPLO 7–8

Para o flip-flop J-K disparado por borda positiva com entradas preset e clear visto na Figura 7–28, determine a saída *Q* para as entradas mostradas no diagrama de temporização na parte (a) se *Q* estiver inicialmente em nível BAIXO.





▲ FIGURA 7–28

Abra o arquivo F07-28 para verificar a operação.

Solução

1. Durante os pulsos de clock 1, 2 e 3, preset ($\overline{PRE}$) é nível BAIXO, mantendo o flip-flop setado independente das entradas síncronas *J* e *K*.

2. Para os pulsos de clock 4, 5, 6 e 7 a operação toggle ocorre porque *J* é nível ALTO, *K* é nível ALTO e $\overline{PRE}$ e $\overline{CLR}$ estão nível ALTO.

3. Para os pulsos de clock 8 e 9, a entrada clear $\overline{CLR}$ é nível BAIXO, mantendo o flip-flop resetado independente das entradas síncronas.

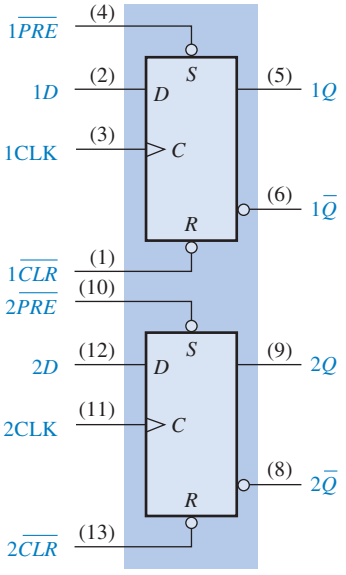
A saída *Q* resultante é mostrada na Figura 7–28(b).

Problema relacionado Se trocarmos entre si as formas de onda de $\overline{PRE}$ e $\overline{CLR}$ na Figura 7–28(a), qual será a saída *Q*?

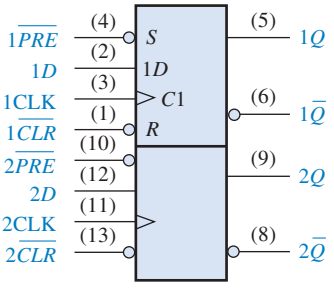
Vamos analisar dois flip-flops disparados por borda específicos. Eles são representantes dos diversos tipos de flip-flops disponíveis na forma de CI e, assim como em outros dispositivos, são comercializados nas famílias lógicas CMOS e TTL.

DUPLO FLIP-FLOP D (74HC74)

Esse dispositivo CMOS contém dois flip-flops D idênticos que são independentes um do outro exceto pelo compartilhamento de V_{CC} e GND. Esses flip-flops são disparados por borda positiva e têm entradas assíncronas preset e clear. Os símbolos lógicos para os flip-flops individuais dentro do encapsulamento são mostrados na Figura 7–29(a) e o símbolo lógico padrão de bloco único da ANSI/IEEE que representa o dispositivo inteiro é mostrado na parte (b). Os números dos pinos são mostrados entre parênteses.



(a) Símbolos lógicos individuais



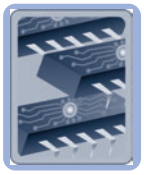
(b) Símbolo lógico de bloco nico
Nota: As letras *S* e *R* dentro do bloco indicam que $\overline{PRE}$ *seta* e $\overline{CLR}$ *reseta*.

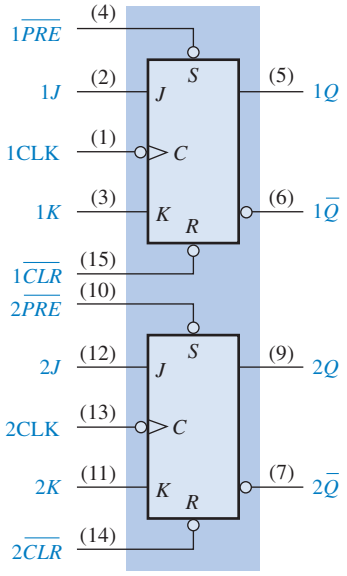
▲ FIGURA 7–29

Símbolos lógicos para o duplo flip-flop D disparado por borda positiva 74AHC74.

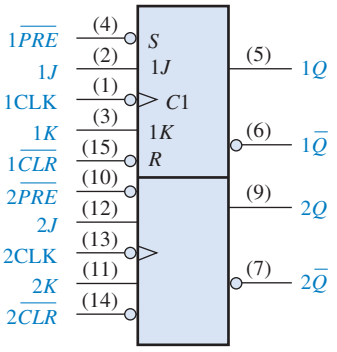
DUPLO FLIP-FLOP J-K (74HC112)

Esse dispositivo tem dois flip-flops idênticos que são disparados por borda negativa e têm entradas assíncronas preset e clear ativas em nível BAIXO. Os símbolos lógicos são mostrados na Figura 7–30.





(a) Símbolos lógicos individuais



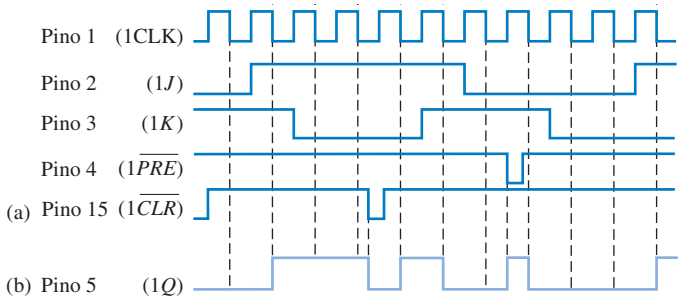
(b) Símbolo lógico de bloco único

◄ **FIGURA 7–30**

Símbolos lógicos para o CI com duplo flip-flop J-K disparado por borda negativa 74HC112.

EXEMPLO 7–9

As formas de onda de 1J, 1K, $1\overline{PRE}$ e $1\overline{CLR}$ mostradas na Figura 7–31(a) são aplicadas em um dos flip-flops disparados por borda negativa do CI 74HC112. Determine a forma de onda da saída $1Q$.



▲ **FIGURA 7–31**

Solução

A forma de onda $1Q$ resultante é mostrada na Figura 7–31(b). Observe que cada vez que um nível BAIXO é aplicado em $1\overline{PRE}$ ou $1\overline{CLR}$, o flip-flop é setado ou resetado independente dos estados das outras entradas.

Problema relacionado

Determine a forma de onda da saída $1Q$ se as formas de onda para $1\overline{PRE}$ e $1\overline{CLR}$ são trocadas entre si.

SEÇÃO 7–2 REVISÃO

1. Descreva a principal diferença entre um latch S-R controlado e um flip-flop disparado por borda.
2. Em que um flip-flop J-K difere de um flip-flop S-R em sua operação básica?
3. Considere que o flip-flop mostrado na Figura 7–21 seja disparado por borda negativa. Descreva a forma de onda de saída para as mesmas formas de onda para as entradas *D* e CLK.

7-3 CARACTERÍSTICAS DE OPERAÇÃO DOS FLIP-FLOPS

O desempenho, os requisitos de operação e as limitações dos flip-flops são especificados por diversas características de operação ou parâmetros encontrados nas folhas de dados dos dispositivos. Geralmente, essas especificações são aplicáveis a todos os flip-flops CMOS e TTL.

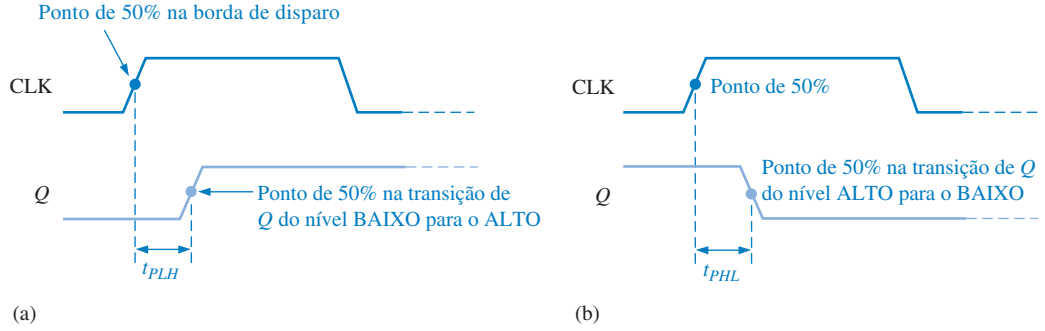
Ao final do estudo desta seção você deverá ser capaz de:

■ Definir *tempo de atraso de propagação* ■ Explicar as diversas especificações do tempo de atraso de propagação ■ Definir *tempo de setup* e discutir como ele limita a operação do flip-flop ■ Definir *tempo de hold* e discutir como ele limita a operação do flip-flop ■ Discutir o significado de freqüência de clock máxima ■ Discutir as diversas especificações de largura de pulso? Definir *dissipação de potência* e calcular o seu valor para um dispositivo especificado ■ Comparar diversas séries de flip-flops em termos dos seus parâmetros de operação

Tempos de Atraso de Propagação

Um **tempo de atraso de propagação** é o intervalo de tempo necessário após a aplicação de um sinal de entrada até que a mudança de saída resultante ocorra. Quatro categorias de tempos de atraso de propagação são importantes na operação de um flip-flop:

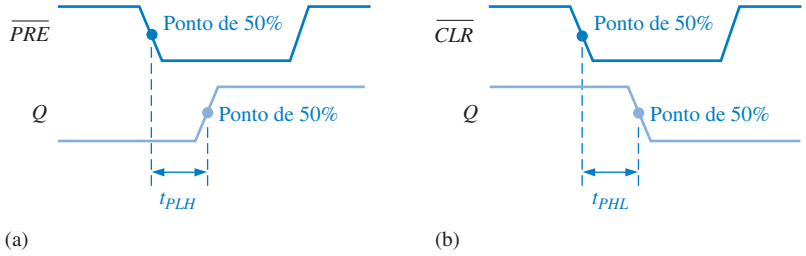
1. Atraso de propagação t_{PLH} medido a partir da borda de disparo do pulso de clock para a transição de nível BAIXO para nível ALTO na saída. Esse atraso é ilustrado na Figura 7–32(a).
2. Atraso de propagação t_{PHL} medido a partir da borda de disparo do pulso de clock para a transição de nível ALTO para nível BAIXO na saída. Esse atraso é ilustrado na Figura 7–32(b).



▲ **FIGURA 7–32**

Atrasos de propagação, do clock para a saída.

3. Atraso de propagação t_{PLH} medido a partir da borda de subida da entrada preset para a transição do nível BAIXO para nível ALTO na saída. Esse atraso é ilustrado na Figura 7–3(a) para uma entrada preset ativa em nível BAIXO.
4. Atraso de propagação t_{PHL} medido a partir da borda de subida da entrada de clear para a transição de nível ALTO para nível BAIXO na saída. Esse atraso é ilustrado na Figura 7–33(b) para uma entrada de clear ativa em nível BAIXO.

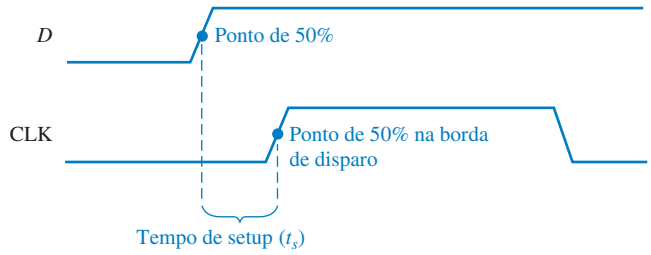


▲ **FIGURA 7–33**

Atrasos de propagação, da entrada à saída preset e da entrada à saída clear.

Tempo de Setup

O **tempo de setup** (preparação), t_s, é o intervalo mínimo necessário para os níveis lógicos se manterem estáveis nas entradas (J e K, ou S e R, ou D) antes da borda de disparo do pulso de clock para que os níveis sejam confiáveis na definição do estado do flip-flop. Esse intervalo é ilustrado na Figura 7–34 para um flip-flop D.



▲ **FIGURA 7–34**

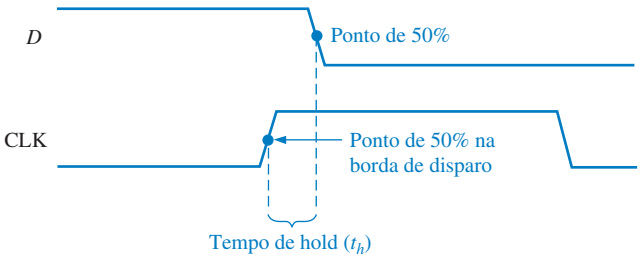
Tempo de *setup* (t_s). O nível lógico tem que estar presente na entrada D por um tempo igual ou maior que t_s antes da borda de disparo do pulso de clock para uma entrada confiável de dados.

DICA PRÁTICA

Uma vantagem de um dispositivo CMOS é que ele pode operar ao longo de uma faixa mais ampla de tensões cc de alimentação (tipicamente de 2 V a 6 V) que um dispositivo TTL e, portanto, podem ser usadas fontes de alimentação mais baratas que não necessitam ter uma regulação precisa. Podem ser usadas também baterias como fontes secundárias ou primárias para circuitos CMOS. Além disso, tensões menores implicam em dissipação de potência menor. A desvantagem é que o desempenho de dispositivos CMOS é degradado com tensões de alimentação menores. Por exemplo, a freqüência de clock máxima garantida de um flip-flop CMOS é muito menor para $V_{CC} = 2$ V que para $V_{CC} = 6$ V.

Tempo de Hold

O **tempo de hold** (manutenção), t_h, é o intervalo de tempo mínimo necessário para que os níveis lógicos permaneçam após a borda de disparo do pulso de clock para que os níveis sejam confiáveis na definição do estado do flip-flop. Isso é ilustrado na Figura 7–35 para um flip-flop D.



▶ **FIGURA 7–35**

Tempo de hold (t_h). O nível lógico tem que ser mantido na entrada *D* por um tempo igual ou maior que t_h após a borda de disparo do pulso de clock para uma operação confiável.

Freqüência de Clock Máxima

A freqüência de clock máxima ($f_{máx}$) é a maior taxa na qual um flip-flop pode ser disparado confiavelmente. Nas freqüências de clock acima da máxima, o flip-flop pode não ser capaz de responder de forma suficientemente rápida sendo a sua operação prejudicada.

Largura de Pulso

As larguras de pulso mínimas (t_W) para uma operação confiável são geralmente especificadas pelo fabricante para as entradas de clock, preset e clear. Tipicamente, o clock é especificado pelos seus tempos de nível ALTO e nível BAIXO mínimos.

Dissipação de Potência

A **dissipação de potência** de qualquer circuito digital é o consumo de potência total do dispositivo. Por exemplo, se o flip-flop opera com uma fonte cc de 5 V e drena 5 mA de corrente, a dissipação de potência é

$$P = V_{CC} \times I_{CC} = 5\text{ V} \times 5\text{ mA} = 25\text{ mW}$$

A dissipação de potência é muito importante na maioria das aplicações nas quais a capacidade da fonte de alimentação é um parâmetro importante. Como exemplo, considere um sistema digital que utiliza dez flip-flops e que cada um dissipa 25 mW de potência. A potência total necessária é

$$P_T = 10 \times 25\text{ mW} = 250\text{ mW} = 0{,}25\text{ W}$$

Isso nos diz qual deve ser a capacidade de saída necessária da fonte de alimentação cc. Se os flip-flops operam com +5 V cc, então a quantidade de corrente que a fonte tem que fornecer é

$$I = \frac{250\text{ mW}}{5\text{ V}} = 50\text{ mA}$$

Temos que usar uma fonte de alimentação de +5 V cc que seja capaz de fornecer pelo menos 50 mA de corrente.

Comparação das Especificações de Flip-Flops

A Tabela 7–5 fornece uma comparação, em termos dos parâmetros discutidos nessa seção de quatro CIs de flip-flops CMOS e TTL do mesmo tipo.

TABELA 7–5

Comparação dos parâmetros de operação para quatro famílias de CIs de flip-flops do mesmo tipo a 25°C

	CMOS		TTL	
PARÂMETRO	74HC74A	74AHC74	74LS74A	74F74
t_{PHL} (CLK para Q)	17 ns	4,6 ns	40 ns	6,8 ns
t_{PLH} (CLK para Q)	17 ns	4,6 ns	25 ns	8,0 ns
t_{PHL} ($\overline{CLR}$ para Q)	18 ns	4,8 ns	40 ns	9,0 ns
t_{PLH} ($\overline{PRE}$ para Q)	18 ns	4,8 ns	25 ns	6,1 ns
t_s (tempo de setup)	14 ns	5,0 ns	20 ns	2,0 ns
t_h (tempo de hold)	3,0 ns	0,5 ns	5 ns	1,0 ns
t_W (CLK ALTO)	10 ns	5,0 ns	25 ns	4,0 ns
t_W (CLK BAIXO)	10 ns	5,0 ns	25 ns	5,0 ns
$t_W(\overline{CLR}/\overline{PRE})$	10 ns	5,0 ns	25 ns	4,0 ns
$f_{máx}$	35 MHz	170 MHz	25 MHz	100 MHz
Potência quiescente	0,012 mW	1,1 mW		
Potência com ciclo de trabalho de 50%			44 mW	88 mW

SEÇÃO 7–3 REVISÃO

1. Defina:
   (a) tempo de setup **(b)** tempo de hold
2. Qual flip-flop em específico na Tabela 7–5 pode operar numa freqüência maior?

7-4 APLICAÇÕES DE FLIP-FLOPS

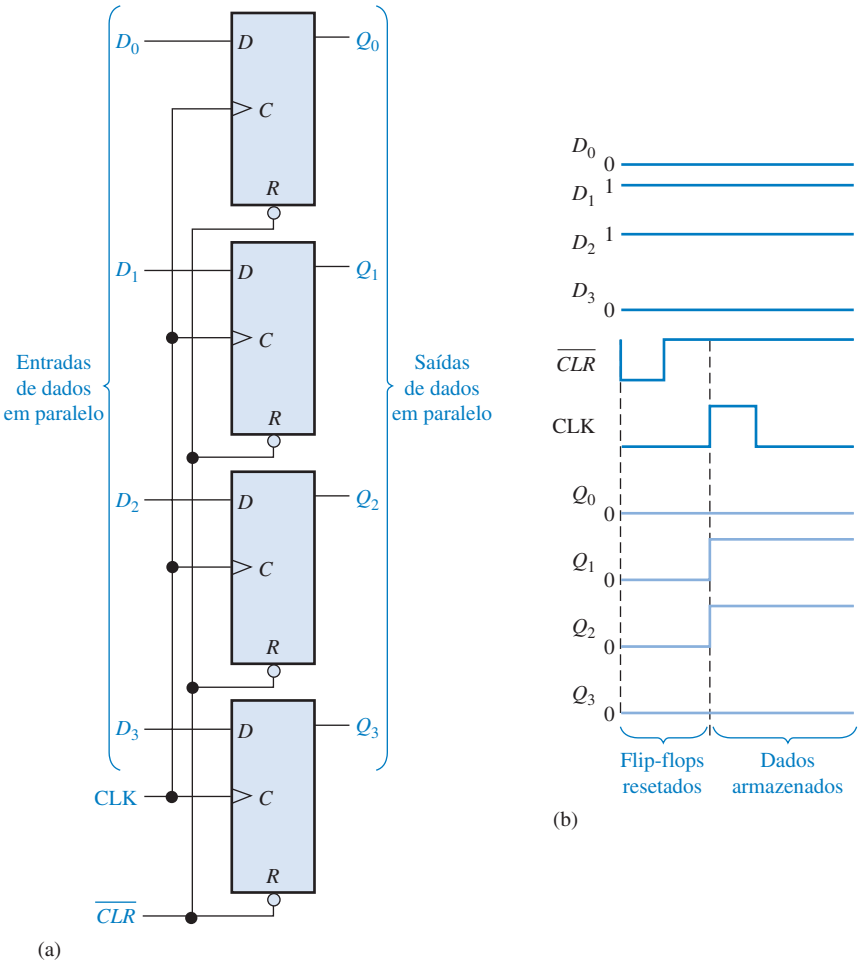
Nesta seção são discutidas três aplicações gerais de flip-flops para dar ao leitor uma idéia de como eles podem ser usados. Os Capítulos 8 e 9 abordam em detalhes as aplicações de flip-flops em contadores e registradores.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir aplicações de flip-flops no armazenamento de dados ■ Descrever como os flip-flops são usados para divisão de freqüência ■ Explicar como os flip-flops são usados em aplicações básicas de contadores

Armazenamento de Dados em Paralelo

Uma necessidade comum em sistemas digitais é armazenar diversos bits de dados em linhas em paralelo simultaneamente num grupo de flip-flops. Essa operação é ilustrada na Figura 7–36(a) usando quatro flip-flops. Cada uma das quatro linhas paralelas de dados é conectada na entrada D de um flip-flop. As entradas de clock dos flip-flops são conectadas juntas, de forma que cada flip-flop é disparado pelo mesmo pulso de clock. Nesse exemplo são usados flip-flops disparados por borda positiva, assim os dados nas entradas D são armazenados simultaneamente pelos flip-flops na borda positiva do clock, conforme indicado no diagrama de temporização visto na Figura 7–36(b). Além disso, as entradas assíncronas de reset (R) são conectadas numa linha $\overline{CLR}$ comum, a qual reseta todos os flip-flops.



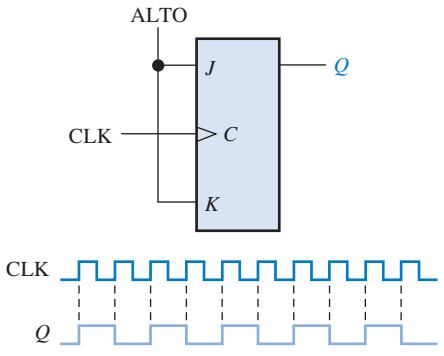
▶ **FIGURA 7–36**

Exemplo de flip-flops usados como um registrador básico para armazenamento paralelo de dados.

Esse grupo de quatro flip-flops é um exemplo de uso de um registrador básico para armazenamento de dados. Em sistemas digitais, os dados são normalmente armazenados em grupos de bits (geralmente oito ou múltiplos dele) que representam números, códigos ou outras informações. Os registradores serão abordados em detalhes no Capítulo 9.

Divisão de Freqüência

Uma outra aplicação de flip-flops é a divisão (redução) de freqüência de uma forma de onda periódica. Quando uma forma de onda retangular é aplicada na entrada de clock de um flip-flop J-K que é conectado no modo toggle ($J = K = 1$), a saída Q é uma onda quadrada com metade da freqüência do sinal na entrada de clock. Portanto, um único flip-flop pode ser usado como um dispositivo divisor por 2, conforme ilustrado na Figura 7–37. Como podemos ver, o flip-flop muda de

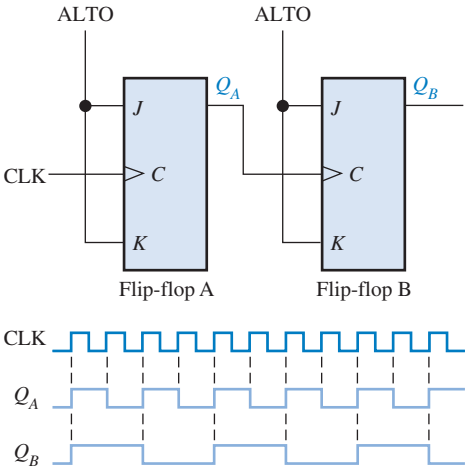


▶ **FIGURA 7–37**

O flip-flop J-K como um dispositivo divisor por 2. Q é a metade da freqüência de CLK.

estado a cada borda de disparo do clock (borda de disparo positiva nesse caso). Isso resulta numa saída que varia com uma freqüência que é metade da freqüência da forma de onda do clock.

Divisões posteriores da freqüência de clock podem ser conseguidas usando a saída de um flip-flop como entrada de clock de um segundo flip-flop, conforme mostra a Figura 7–38. A freqüência da saída Q_A é dividida por 2 pelo flip-flop *B*. Portanto, a saída Q_B é um quarto da freqüência da entrada de clock original. Os tempos de atraso de propagação não são mostrados nos diagramas de temporização.



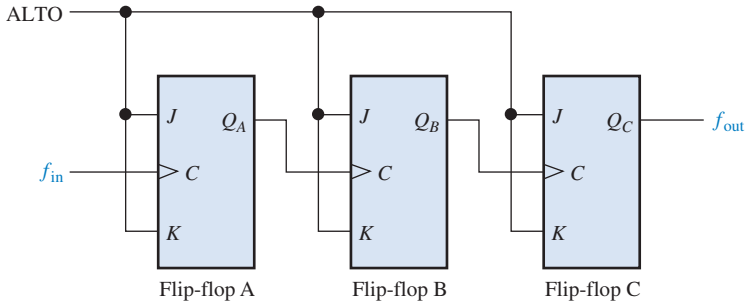
◀ **FIGURA 7–38**

Exemplo do uso de dois flip-flops J-K para dividir a freqüência de clock por 4. Q_A é metade da freqüência de CLK e Q_B é um quarto da mesma.

Conectando flip-flops dessa forma, obtemos uma divisão de freqüência por 2^n, onde *n* é o numero de flip-flops. Por exemplo, três flip-flops dividem a freqüência de clock por $2^3 = 8$; quatro flip-flops dividem a freqüência de clock por $2^4 = 16$; e assim por diante.

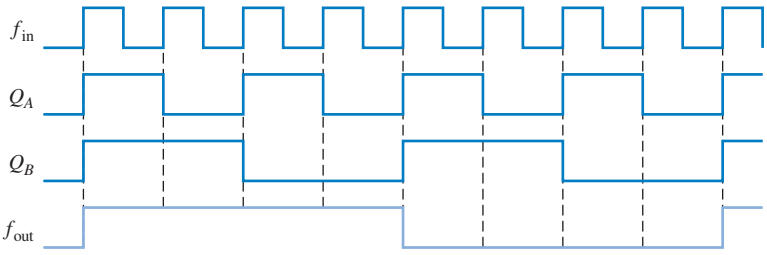
EXEMPLO 7–10

Determine a forma de onda f_{out} para o circuito dado na Figura 7–39 quando uma onda quadrada de 8 kHz for aplicada na entrada de clock do flip-flop A.



▲ **FIGURA 7–39**

Solução Os três flip-flops são conectados para dividir a freqüência de entrada por oito ($2^3 = 8$) e a forma de onda de f_{out} é mostrada na Figura 7–40. Como esses flip-flops são disparados na borda positiva, as saídas mudam na borda positiva do clock. Ocorre um pulso de saída para cada oito pulsos de entrada, assim a freqüência de saída é 1 kHz. As formas de onda de Q_A e Q_B também são mostradas.

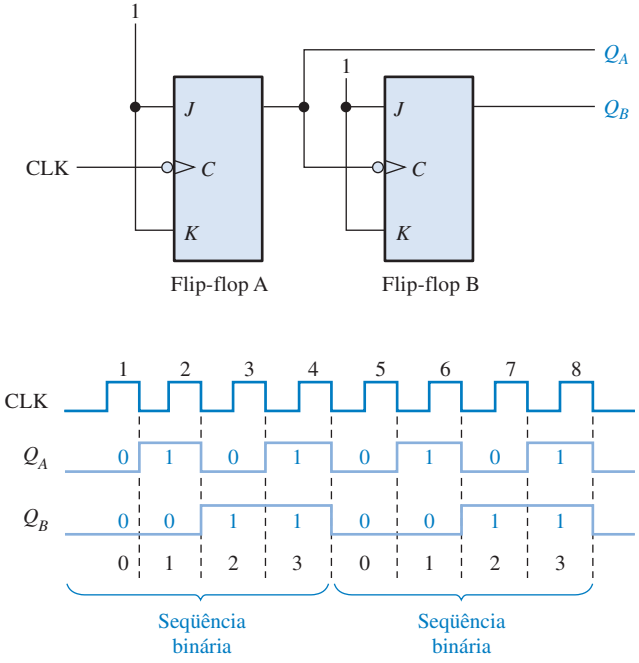


▲ **FIGURA 7–40**

Problema relacionado Quantos flip-flops são necessários para dividir uma freqüência por trinta e dois?

Contagem

Uma outra aplicação importante de flip-flops é os contadores digitais, os quais serão abordados em detalhes no Capítulo 8. O conceito é ilustrado na Figura 7–41. Os flip-flops são do tipo J-K disparados pela borda negativa. Os dois flip-flops estão inicialmente *resetados*. O flip-flop A comuta na transição negativa de cada pulso de clock. A saída Q do flip-flop A é o clock do flip-flop B, assim cada vez que Q_A faz uma transição de nível ALTO para nível BAIXO, o flip-flop B muda de estado (toggle). As formas de onda resultante de Q_A e Q_B são mostradas na figura.



▶ **FIGURA 7–41**

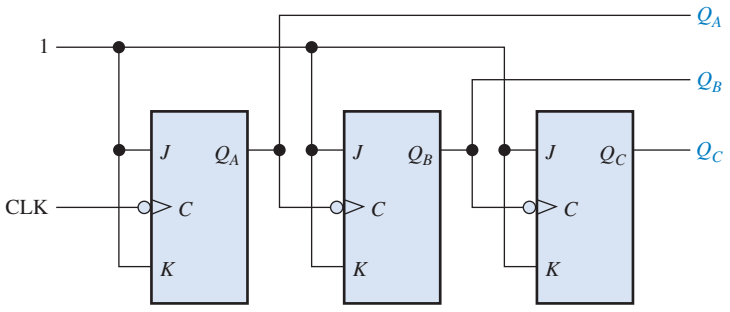
Flip-flops usados para gerar uma seqüência de contagem binária. São mostradas dois ciclos (00, 01, 10, 11).

Observe a seqüência de Q_A e Q_B na Figura 7–41. Antes do pulso de clock 1, $Q_A = 0$ e $Q_B = 0$; após o pulso de clock 1, $Q_A = 1$ e $Q_B = 0$; após o pulso de clock 2, $Q_A = 0$ e $Q_B = 1$; e após o pulso de clock 3, $Q_A = 1$ e $Q_B = 1$. Se tomarmos Q_A como sendo o bit menos significativo, uma seqüên-

cia de 2 bits é produzida enquanto os flip-flops recebem clocks. Essa seqüência binária se repete a cada quatro pulsos de clock, conforme mostrado no diagrama de temporização visto na Figura 7–41. Portanto, os flip-flops realizam uma contagem de 0 a 3 (00, 01, 10, 11) e então retornam para 0 começando a seqüência novamente.

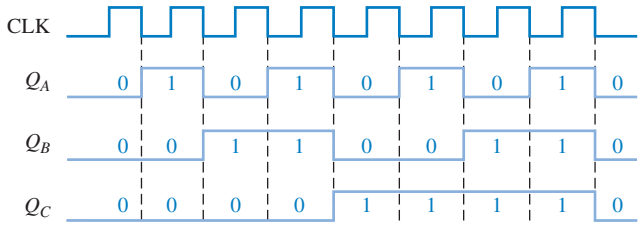
EXEMPLO 7–11

Determine as formas de onda de saída em relação ao clock para Q_A, Q_B e Q_C no circuito visto na Figura 7–42 e mostre a seqüência binária representada por essas formas de onda.



▲ FIGURA 7–42

Solução O diagrama de temporização da saída é mostrado na Figura 7–43. Observe que as saídas mudam na borda negativa dos pulsos de clock. As saídas passam pela seqüência binária 000, 001, 010, 011, 100, 101, 110 e 111, conforme indicado.



▲ FIGURA 7–43

Problema relacionado Quantos flip-flops são necessários para produzir uma seqüência binária que representa os números decimais de 0 a 15?

SEÇÃO 7–4 REVISÃO

1. Como é denominado o grupo de flip-flops que armazena dados?
2. Quantos flip-flops J-K devem ser conectados para funcionar como um dispositivo divisor por 2?
3. Quantos flip-flops são necessários para produzir um dispositivo divisor por 64?

7-5 MONOESTÁVEIS

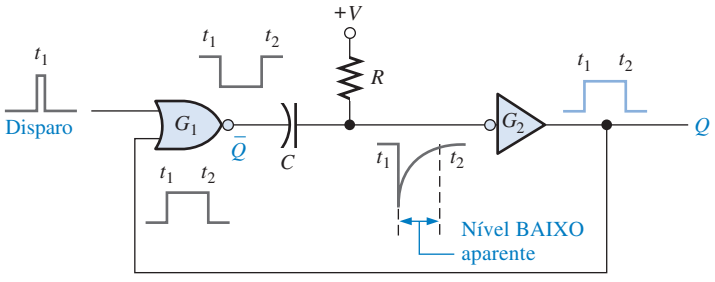
O **multivibrador monoestável** é um dispositivo com apenas um estado estável. Um monoestável está normalmente no estado estável mudando para o estado instável apenas quando disparado. Uma vez disparado, o monoestável permanece no estado instável por um período predeterminado de tempo retornando automaticamente para o estado estável. O tempo em que o dispositivo permanece no estado instável determina a largura do pulso na saída.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a operação básica de um monoestável ■ Explicar como funciona um monoestável não-redisparável ■ Explicar como funciona um monoestável redisparável ■ Configurar os CIs monoestáveis 74121 e 74LS122 para obter uma largura de pulso de saída especificada ■ Reconhecer o símbolo de um Schmitt-trigger e explicar basicamente como ele funciona

Um monoestável produz um único pulso cada vez que é disparado.

A Figura 7–44 mostra um monoestável básico (multivibrador monoestável) que é composto de uma porta lógica e um inversor. Quando um pulso é aplicado na entrada de **disparo** (trigger), a saída da porta G_1 vai para nível BAIXO. Essa transição de nível ALTO para nível BAIXO é acoplada através de um capacitor à entrada do inversor G_2. O nível BAIXO aparente em G_2 faz com que a saída vá para nível ALTO. Esse nível ALTO é conectado de volta em G_1, mantendo a saída de G_1 em nível BAIXO. Até esse ponto, o pulso de disparo fez com que a saída do monoestável (Q) fosse para nível ALTO.

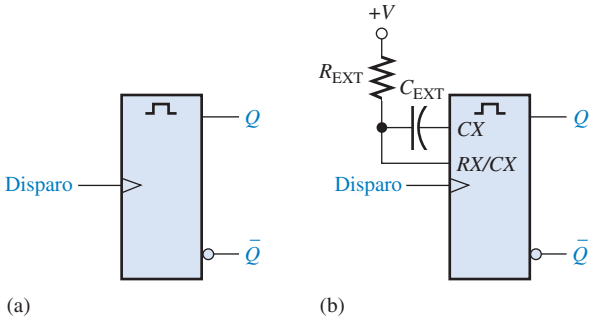


► **FIGURA 7–44**
Um circuito simples de um monoestável.

O capacitor começa a carregar imediatamente através de R em direção à tensão de nível ALTO. A taxa na qual ele se carrega é determinada pela constante de tempo RC. Quando o capacitor carrega até um certo nível, o qual é visto como nível ALTO por G_2, a saída retorna para o nível BAIXO.

Para resumir, a saída do inversor G_2 vai para nível ALTO em resposta à entrada de disparo. Ele permanece no nível ALTO por um tempo determinado pela constante de tempo RC. No final desse tempo, ele retorna para o nível BAIXO. Um único pulso de disparo estreito produz um único pulso de saída cuja duração é controlada pela constante de tempo RC. Essa operação está ilustrada na Figura 7–44.

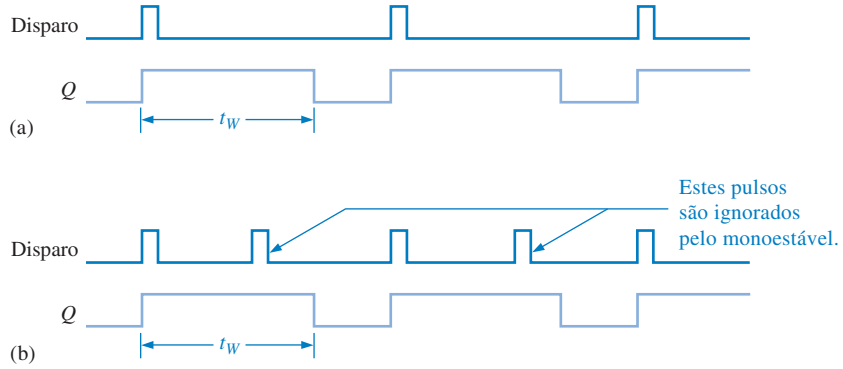
O símbolo lógico de um monoestável típico é mostrado na Figura 7–45(a) e o mesmo símbolo com R e C externos é mostrado na Figura 7–45(b). Os dois tipos básicos de monoestáveis na forma de CI são o não-redisparável e o redisparável.



► **FIGURA 7–45**
Símbolo lógico de um monoestável básico. *CX* e *RX* suportam as conexões dos componentes externos.

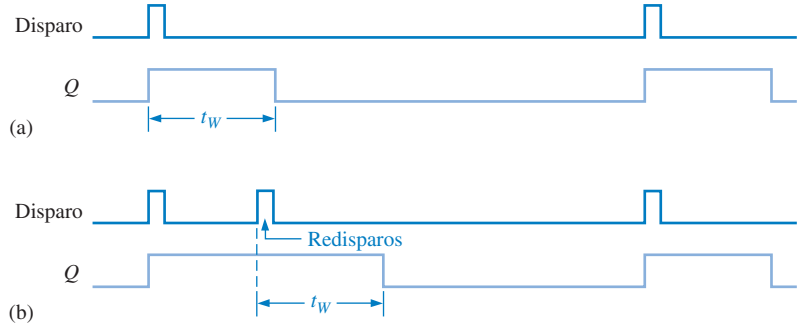
Um monoestável não-redisparável não responde a nenhum pulso de disparo adicional a partir do instante em que é disparado (passa para o estado instável) até retornar ao estado estável. Em outras palavras, ele ignora qualquer pulso de disparo que ocorre antes do término da temporização. O tempo no qual o monoestável permanece no estado instável é a largura do pulso de saída.

A Figura 7–46 mostra um monoestável não-redisparável sendo disparado em intervalos maiores que a largura de pulso e em intervalos menores que a largura de pulso. Observe que no segundo caso, os pulsos adicionais são ignorados.



◀ **FIGURA 7–46**
Atuação de um monoestável não-redisparável.

Um monoestável redisparável pode ser disparado antes do final da temporização. O resultado do redisparo é uma extensão da largura de pulso conforme ilustrado na Figura 7–47.



◀ **FIGURA 7–47**
Atuação de um monoestável redisparável.

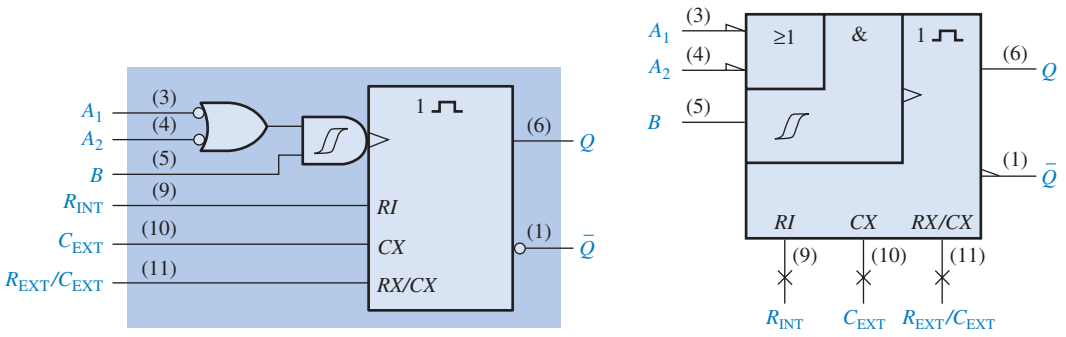
MONOESTÁVEL NÃO-REDISPARÁVEL (74121)

O CI 74121 é um exemplo de um monoestável não-redisparável. Ele provê conexões para *R* e *C* externos, conforme mostra a Figura 7–48. As entradas denominadas de A_1, A_2 e *B* são entradas de controle de disparo. A entrada R_{INT} conecta a um resistor de temporização interno de 2 kΩ interno.

Ajustando a Largura do Pulso Uma largura de pulso típica de 30 ns é produzida quando nenhum componente de temporização externo é usado e o resistor de temporização interno (R_{INT}) é conectado a V_{CC}, como mostra a Figura 7–49(a). A largura do pulso pode ser definida em algum valor entre cerca de 30 ns e 28 s através do uso de componentes externos. A Figura 7–49(b) mostra a configuração usando o resistor interno (2 kΩ) e um capacitor externo. A parte (c) mostra a configuração usando um resistor externo e um capacitor externo. A largura do pulso de saída é definida pelo valor do resistor (R_{INT} = 2 kΩ e R_{EXT} é selecionado) e o capacitor de acordo com a seguinte fórmula:

$$t_W = 0{,}7RC_{EXT}$$ **Equação 7–1**

Onde *R* é R_{INT} ou R_{EXT}. Quando *R* está em quiloohms (kΩ) e C_{EXT} está em picofarads (pF), a largura do pulso de saída t_W está em nanossegundos (ns).

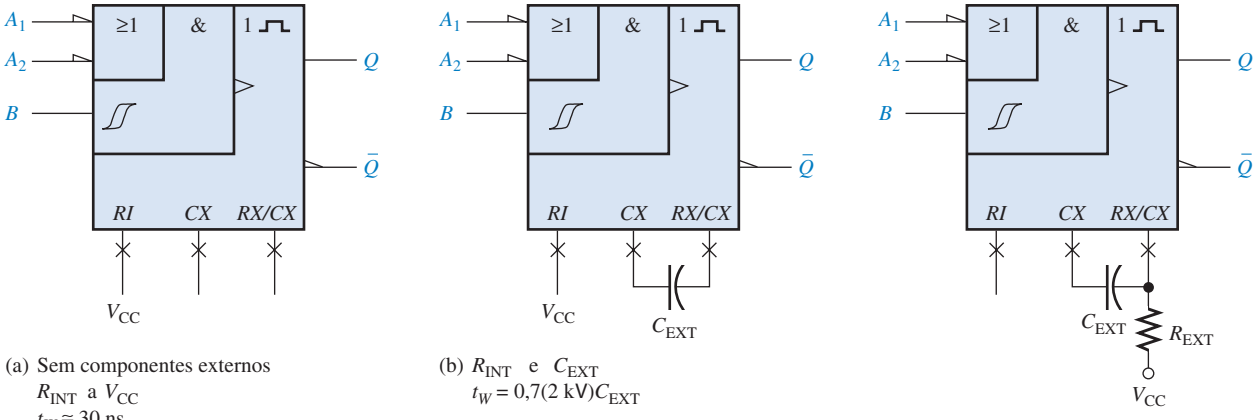


(a) Símbolo lógico tradicional

(b) Símbolo lógico padrão 91–1984 da ANSI/IEEE (X = sem conexão lógica). "1 ⊓⊔" é o símbolo qualificativo para um monoestável não-redisparável.

▲ **FIGURA 7–48**

Símbolos lógicos para o monoestável não-redisparável 74121.



(a) Sem componentes externos
R_{INT} a V_{CC}
$t_W \cong 30$ ns

(b) R_{INT} e C_{EXT}
$t_W = 0{,}7(2\text{ kV})C_{EXT}$

(c) R_{EXT} e C_{EXT}
$t_W = 0{,}7R_{EXT}\,C_{EXT}$

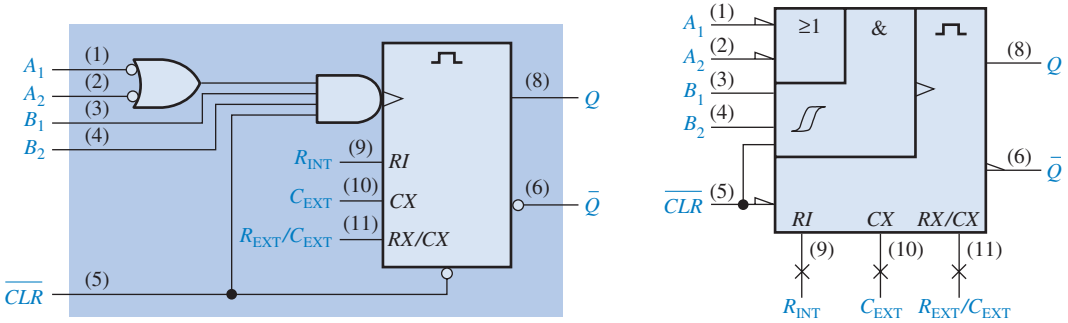
▲ **FIGURA 7–49**

Três formas de estabelecer a largura de pulso de um CI 74121.

O Símbolo do Schmitt-Trigger O símbolo ∫∫ indica uma entrada Schmitt-trigger. Esse tipo de entrada usa um circuito de limiar especial que produz **histerese**, uma característica que evita erros de chaveamento entre estados quando uma tensão de disparo de variação lenta paira em torno do nível de entrada crítico. Isso permite segurança no disparo sempre quando a entrada varia de forma lenta como 1 volt/segundo.

MONOESTÁVEL REDISPARÁVEL (74LS122)

O CI 74LS122 é um exemplo de CI monoestável redisparável com uma entrada de *clear*. Ele também provê conexões externas de *R* e *C*, conforme mostra a Figura 7–50. As entradas denominadas A_1, A_2, B_1 e B_2 são entradas de controle de disparo.



(a) Símbolo lógico tradicional.

(b) Símbolo lógico padrão 91–1984 da ANSI/IEEE (X = sem conexão lógica). ⊓⊔ é o símbolo qualificativo para um monoestável redisparável.

▲ **FIGURA 7–50**

Símbolo lógico para o CI monoestável redisparável 74LS122.

Uma largura de pulso mínima de aproximadamente 45 ns é obtida sem componentes externos. A maior largura de pulso é conseguida usando componentes externos. Uma fórmula geral para o cálculo dos valores desses componentes para uma largura de pulso específica (t_W) é

$$t_W = 0{,}32RC_{\text{EXT}}\left(1 + \frac{0{,}7}{R}\right)$$

Equação 7–2

onde 0,32 é uma constante determinada pelo tipo particular de monoestável, R é dado em kΩ podendo ser um resistor interno ou externo ao CI, C_{EXT} é dado em pF e t_W é dado em ns. A resistência interna é de 10 kΩ e pode ser usada em vez de um resistor externo. (Observe a diferença entre essa fórmula e aquela para o CI 74121, mostrada na Equação 7–1.)

EXEMPLO 7–12

Uma certa aplicação necessita de um monoestável com uma largura de pulso de aproximadamente 100 ms. Usando um 74121, mostre as conexões e os valores dos componentes.

Solução Arbitrariamente selecione $R_{\text{EXT}} = \mathbf{39\ k\Omega}$ e calcule a capacitância necessária.

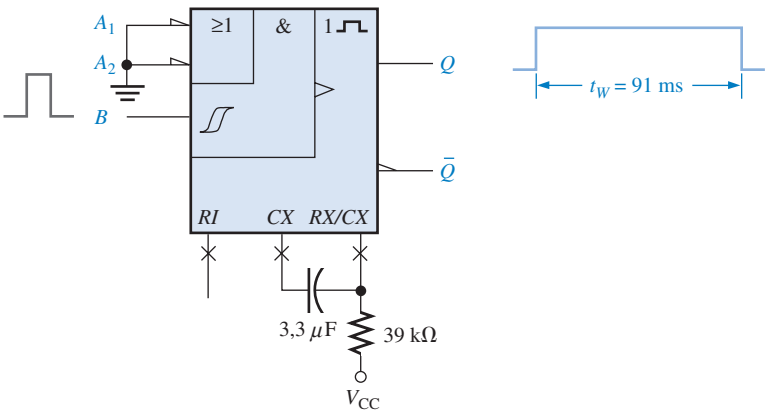
$$t_W = 0{,}7R_{\text{EXT}}C_{\text{EXT}}$$

$$C_{\text{EXT}} = \frac{t_W}{0{,}7R_{\text{EXT}}}$$

onde C_{EXT} está em pF, R_{EXT} está em kΩ e t_W está em ns. Como 100 ms = 1×10^8 ns,

$$C_{\text{EXT}} = \frac{1 \times 10^8\ \text{ns}}{0{,}7(39\ \text{k}\Omega)} = 3{,}66 \times 10^{-6}\ \text{pF} = \mathbf{3{,}66\ \mu F}$$

Um capacitor padrão de 3,3μF proporciona uma largura de pulso de saída de 91 ms. As conexões apropriadas são mostradas na Figura 7–51. Para conseguir uma largura de pulso mais próximo de 100 ms, outras combinações de valores para R_{EXT} e C_{EXT} podem ser testadas. Por exemplo, R_{EXT} = 68 kΩ e C_{EXT} de 2,2μF proporcionam uma largura de pulso de 105 ms.



▲ FIGURA 7–51

Problema relacionado Use um capacitor externo juntamente com R_{INT} para produzir uma largura de pulso de saída de 10μs a partir do CI 74121.

EXEMPLO 7–13

Determine os valores de R_{EXT} e C_{EXT} que produzirão uma largura de pulso de 1μs quando conectado a um 74LS122.

Solução Considere um valor de C_{EXT} = **560 pF** e então calcule para R_{EXT}. A largura do pulso tem que ser expressa em ns e C_{EXT} em pF. R_{EXT} será em kΩ.

$$t_W = 0{,}32R_{EXT}C_{EXT}\left(1 + \frac{0{,}7}{R_{EXT}}\right) = 0{,}32R_{EXT}C_{EXT} + 0{,}7\left(\frac{0{,}32R_{EXT}C_{EXT}}{R_{EXT}}\right)$$

$$= 0{,}32R_{EXT}C_{EXT} + (0{,}7)(0{,}32)C_{EXT}$$

$$R_{EXT} = \frac{t_W - (0{,}7)(0{,}32)C_{EXT}}{0{,}32C_{EXT}} = \frac{t_W}{0{,}32C_{EXT}} - 0{,}7$$

$$= \frac{1000\text{ ns}}{(0{,}32)560\text{ pF}} - 0{,}7 = \mathbf{4{,}88\ k\Omega}$$

Use um valor padrão de **4,7 kΩ**.

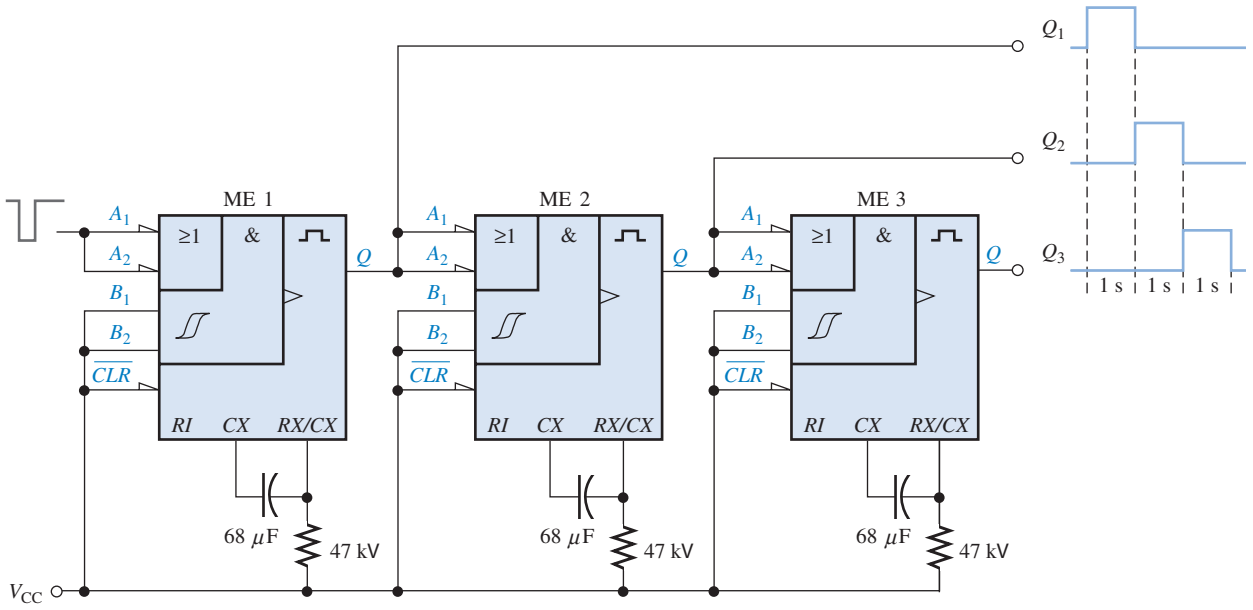
Problema relacionado Mostre as conexões e os valores dos componentes para um CI monoestável 74LS122 com uma largura de pulso de 5μs. Considere C_{EXT} = 560 pF.

Uma Aplicação

Uma aplicação prática de monoestável é um temporizador seqüencial que pode ser usado para iluminar uma série de luzes. Esse tipo de circuito pode ser usado, por exemplo, num indicador direcional de mudança de pista em projetos de construção de estradas ou em sinais de mudança seqüencial para automóveis.

A Figura 7–52 mostra três CIs monoestáveis 74LS122 conectados como um temporizador seqüencial. Esse circuito em particular produz uma seqüência de três pulsos de 1 s. O primeiro monoestável é disparado pelo fechamento de uma chave ou uma entrada de pulso de baixa freqüência, produzindo um pulso de saída de 1 s. Quando o primeiro monoestável (ME 1) finaliza a temporização e o pulso de 1 s vai para nível BAIXO, o segundo monoestável (ME 2) é disparado, pro-

duzindo também um pulso de saída de 1 s. Quando esse segundo pulso vai para nível BAIXO, o terceiro monoestável (ME 3) é disparado e o terceiro pulso de 1 s é produzido. A temporização de saída é ilustrada na figura. Podem ser usadas variações básicas desse arranjo para produzir uma variedade de saídas temporizadas.



▲ **FIGURA 7–52**

Um circuito de temporização seqüencial usando três CIs monoestáveis 74LS122.

SEÇÃO 7–5 REVISÃO

1. Descreva a diferença entre um monoestável não-redisparável e um redisparável.
2. Como é definida a largura de pulso de saída na maioria dos CIs monoestáveis?

7-6 TEMPORIZADOR 555

O **temporizador** 555 é um dispositivo, na forma de CI, versátil e amplamente usado pois pode ser configurado em dois modos diferentes: multivibrador monoestável ou um multivibrador astável (oscilador). Um multivibrador astável não tem estado estável oscilando entre dois estados estáveis sem qualquer disparo externo.

Ao final do estudo desta seção você deverá ser capaz de:

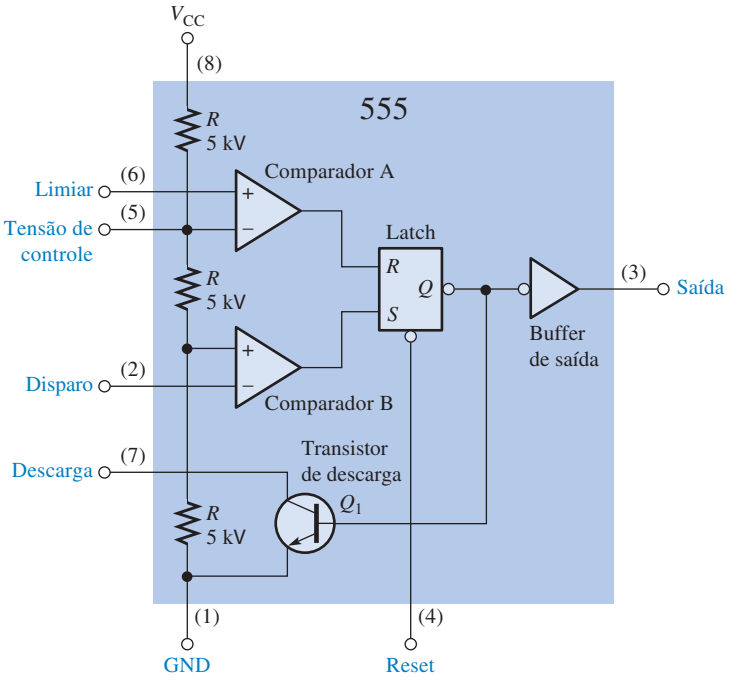
■ Descrever os blocos básicos de um temporizador 555 ■ Configurar um 555 como um monoestável ■ Configurar um 555 como um oscilador

Operação Básica

Um temporizador 555 pode operar como um monoestável ou como um oscilador (astável).

A Figura 7–53 apresenta um diagrama funcional mostrando os componentes internos do temporizador 555. Os comparadores são dispositivos cujas saídas são nível ALTO quando a tensão na entrada positiva (+) for maior que a tensão na entrada negativa (–) e nível BAIXO quando a tensão da entrada – for maior que a tensão na entrada +. O divisor de tensão consiste de três resistores de 5 kΩ, os quais fornecem um nível de limiar de 1/3 de V_{CC} e um nível de limiar de 2/3 de V_{CC}. A entrada de tensão de controle (pino 5) pode ser usada para ajuste externo dos níveis de disparo e limiar em outros

valores se necessário. Quando a entrada de disparo que é normalmente nível ALTO for momentaneamente para um valor abaixo de 1/3 V_{CC}, a saída do comparador *B* comuta de nível BAIXO para nível ALTO e seta o latch S-R, fazendo com que a saída (pino 3) passe para o nível ALTO e desligue o transistor de descarga Q_1. A saída permanece em nível ALTO até que a entrada de limiar, que normalmente é nível BAIXO, atinja um valor acima de 2/3 V_{CC} e faça com que a saída do comparador A comute de nível BAIXO para nível ALTO. Isso reseta o latch, fazendo com que a saída do 555 retorne para o nível BAIXO e ligue o transistor de descarga. A entrada de reset externo pode ser usada para resetar o latch independente do circuito de limiar. As entradas de disparo e limiar (pinos 2 e 6) são controladas por componentes externos conectados para produzir ações de monoestável e astável.



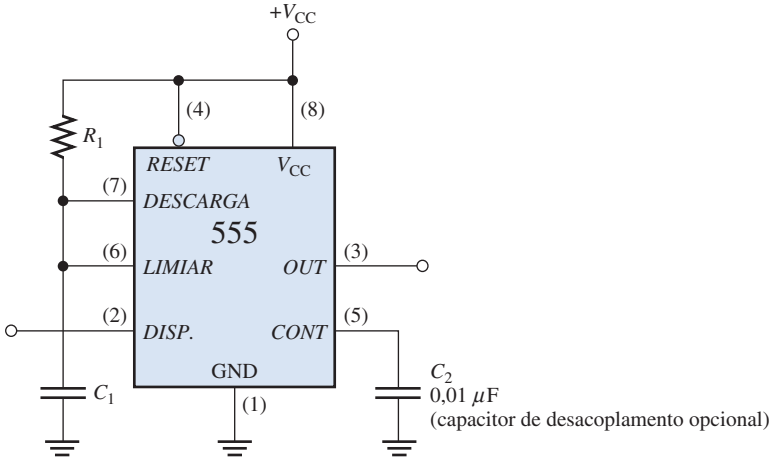
▶ **FIGURA 7–53**
Diagrama funcional interno de um temporizador 555 (os números dos pinos estão entre parênteses).

Operação Monoestável

Para configurar o temporizador 555 como monoestável não-redisparável são usados um resistor e um capacitor externos como mostra a Figura 7–54. A largura do pulso da saída é determinado pela constante de tempo de R_1 e C_1 de acordo com a fórmula

Equação 7–3

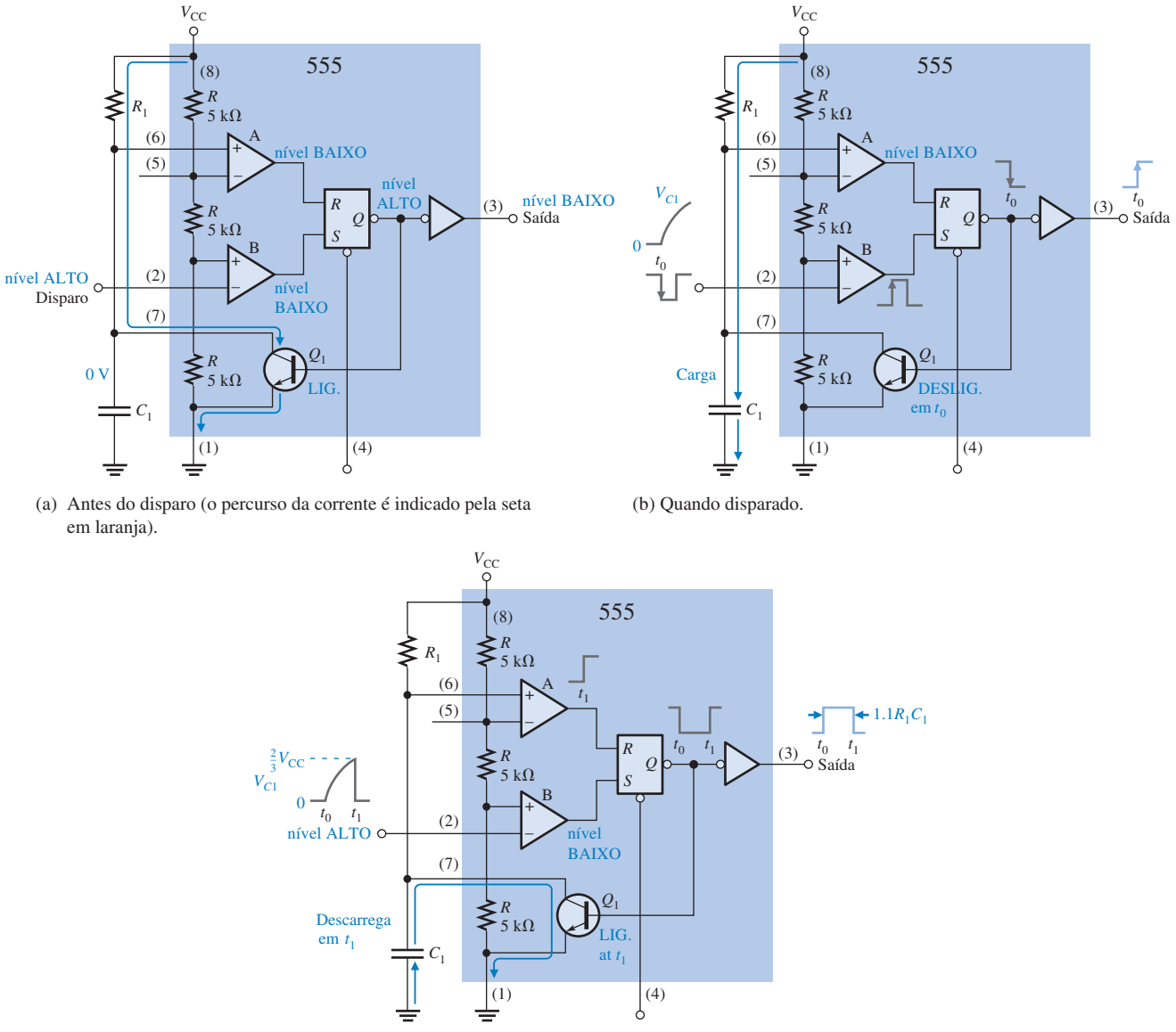
$$t_W = 1{,}1R_1C_1$$



▶ **FIGURA 7–54**
O temporizador 555 conectado como um monoestável.

A entrada de tensão de controle não é usada e é conectada a um capacitor de desacoplamento C_2 para evitar que um ruído afete os níveis de disparo e limiar.

Antes que um pulso de disparo seja aplicado, a saída é nível BAIXO e o transistor de descarga Q_1 está ligado (*on*), mantendo C_1 descarregado como mostra a Figura 7–55(a). Quando um pulso de disparo negativo (de nível ALTO para BAIXO) é aplicado no instante t_0, a saída vai para nível ALTO e o transistor de descarga é desligado (*off*), permitido que o capacitor C_1 seja carregado através de R_1 conforme mostra a parte (b). Quando C_1 se carrega até 2/3 V_{CC}, a saída retorna para o nível BAIXO em t_1 e Q_1 é imediatamente *ligado*, descarregando C_1 como mostra a parte (c). Como podemos ver, a taxa de carga de C_1 determina por quanto tempo a saída permanece em nível ALTO.



(a) Antes do disparo (o percurso da corrente é indicado pela seta em laranja).

(b) Quando disparado.

(c) No término do intervalo de carga.

▲ **FIGURA 7–55**

Operação do temporizador 555 como monoestável.

EXEMPLO 7–14

Qual é a largura do pulso de um circuito monoestável com 555 tendo $R_1 = 2{,}2\ \text{k}\Omega$ e $C_1 = 0{,}001\mu\text{F}$?

Solução A partir da Equação 7–3 a largura é

$$t_W = 1{,}1R_1C_1 = 1{,}1(2{,}2\ \text{k}\Omega)(0{,}01\ \mu\text{F}) = \mathbf{24{,}2\ \mu s}$$

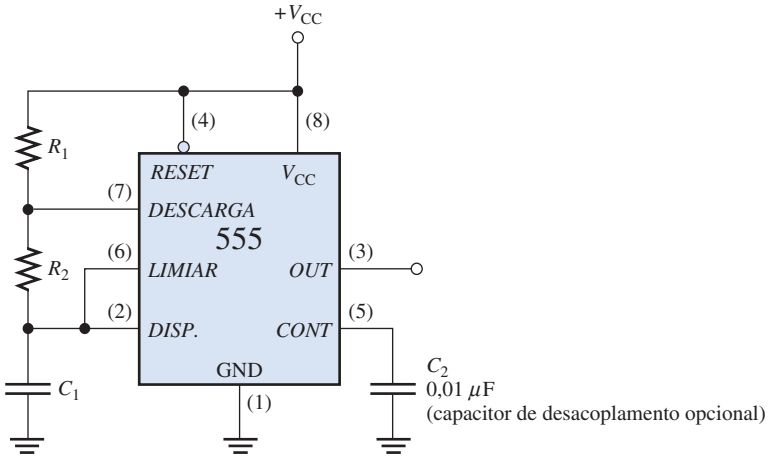
Problema relacionado Para $C_1 = 0{,}001\mu\text{F}$, determine o valor de R_1 para uma largura de pulso de 1 ms.

NOTA: COMPUTAÇÃO

Todos os computadores necessitam de uma fonte de temporização para prover formas de onda de clock precisas. A seção de temporização controla toda a temporização do sistema e é responsável pela correta operação do hardware. A seção de temporização geralmente consiste de um oscilador controlado a cristal e contadores para divisão de freqüência. O uso de um oscilador de alta freqüência que tem a freqüência dividida para um valor menor proporciona maior precisão e estabilidade na freqüência.

Operação Astável

Um temporizador 555 conectado para operar como um multivibrador **astável**, o qual não é um **oscilador** senoidal, é mostrado na Figura 7–56. Observe que a entrada de limiar agora está conectada na entrada de disparo. Os componentes externos R_1, R_2 e C_1 formam o circuito de temporização que define a freqüência de oscilação. O capacitor de 0,001μF, C_2, é conectado na entrada de controle (CONT) estritamente para desacoplamento e não tem efeito na operação; em alguns casos ele pode ser suprimido.



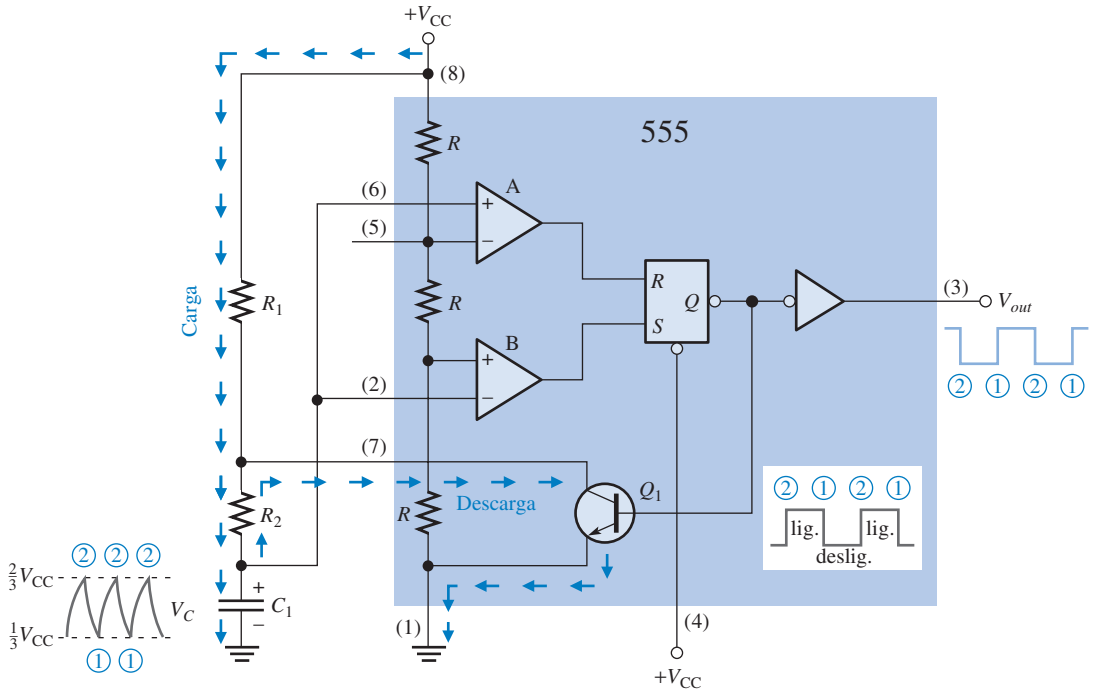
▲ **FIGURA 7–56**

O temporizador 555 conectado como um multivibrador astável (oscilador).

Inicialmente, quando a alimentação é ligada, o capacitor (C_1) está descarregado e a tensão de disparo (pino 2) é 0 V. Isso faz com que a saída do comparador B seja nível ALTO e a saída do comparador A seja nível BAIXO, forçando a saída do latch, e assim a base de Q_1, para nível BAIXO mantendo o transistor desligado. Agora, C_1 começa carregando através de R_1 e R_2, conforme indicado na Figura 7–57. Quando a tensão no capacitor alcança 1/3 V_{CC}, o comparador *B* comuta sua saída para o estado BAIXO; e quando a tensão no capacitor alcança 2/3 V_{CC}, o comparador *A* comuta sua saída para o estado ALTO. Isso reseta o latch, fazendo com que a base de Q_1 passe para o nível ALTO desligando o transistor. Essa seqüência cria um percurso de descarga para o capacitor através de R_2 e do transistor, conforme indicado. Agora o capacitor começa a descarregar, fazendo com que o comparador A vá para nível BAIXO. Quando o capacitor se descarrega para um valor abaixo de 1/3 V_{CC}, o comparador comuta para nível ALTO; isso seta o latch, fazendo com que a base de Q_1 seja nível BAIXO desligando o transistor. Um outro ciclo de carga inicia, e o processo se repete. O resultado é uma onda de saída retangular cujo ciclo de trabalho depende dos valores de R_1 e R_2. A freqüência de oscilação é dada pela seguinte fórmula, ou ainda pode ser determinada de forma gráfica conforme a Figura 7–58.

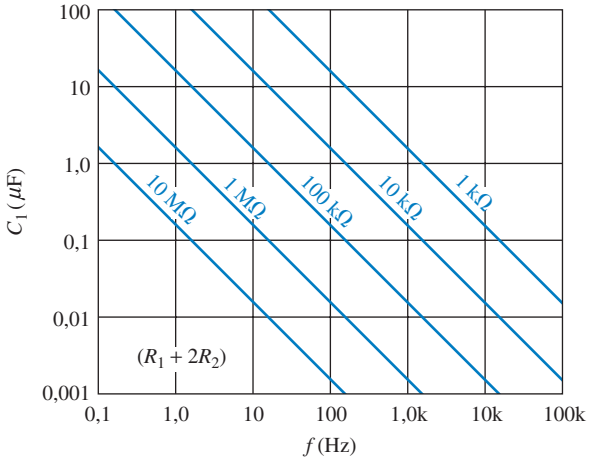
Equação 7–4

$$f = \frac{1{,}44}{(R_t + 2R_2)C_1}$$



▲ **FIGURA 7–57**

Operação do temporizador 555 no modo astável.



◀ **FIGURA 7–58**

Freqüência de oscilação como uma função de C_1 e $R_1 + 2R_2$. As linhas inclinadas são os valores de $R_1 + 2R_2$.

Selecionando R_1 e R_2, o ciclo de trabalho da saída pode ser ajustado. Como C_1 carrega através de $R_1 + R_2$ e descarrega apenas através de R_2, os ciclos de trabalho que se aproximam de um mínimo de 50% podem ser obtidos se $R_2 >> R_1$ de forma que os tempos de carga e descarga sejam aproximadamente iguais.

Uma expressão para o ciclo de trabalho é desenvolvida a seguir. O tempo em que a saída é nível ALTO (t_H) é o tempo que C_1 leva entre 1/3 V_{CC} e 2/3 V_{CC}. Esse tempo é expresso como a seguir:

$$t_H = 0{,}7(R_1 + R_2)C_1$$ **Equação 7–5**

O tempo em que a saída é nível BAIXO (t_L) é o tempo que C_1 leva para descarregar de 1/3 V_{CC} para 2/3 V_{CC}. Esse tempo é expresso como a seguir:

$$t_L = 0{,}7R_2C_1$$ **Equação 7–6**

O período, T, da forma de onda de saída é a soma de t_H com t_L. Esse tempo é o inverso de f na Equação 7–4.

$$T = t_H + t_L = 0{,}7(R_1 + 2R_2)C_1$$

Finalmente, o ciclo de trabalho é

$$\text{Ciclo de trabalho} = \frac{t_H}{T} = \frac{t_H}{t_H + t_L}$$

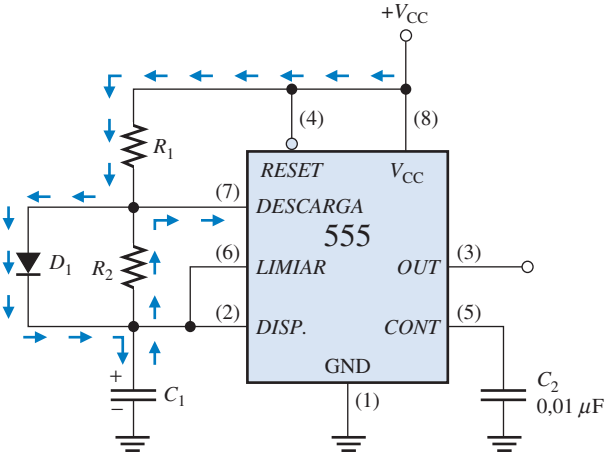
Equação 7–7

$$\text{Ciclo de trabalho} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\%$$

Para obter um ciclo de trabalho menor que 50%, o circuito mostrado na Figura 7–56 pode ser modificado de forma que C_1 carregue através de R_1 apenas e descarregue através de R_2. Isso é conseguido usando o diodo D_1, conforme a Figura 7–59. O ciclo de trabalho pode ser menor que 50% fazendo R_1 menor que R_2. Sob essa condição, a expressão para o ciclo de trabalho é

Equação 7–8

$$\text{Ciclo de trabalho} = \left(\frac{R_1}{R_1 + R_2}\right)100\%$$

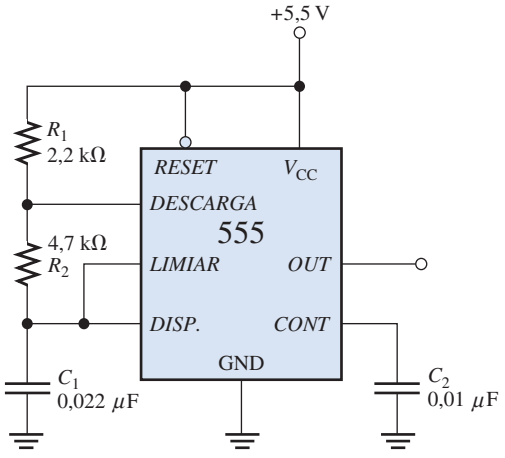


▶ FIGURA 7–59

A adição do diodo D_1 permite que o ciclo de trabalho da saída seja ajustado em menos de 50% fazendo $R_1 < R_2$.

EXEMPLO 7–15

A Figura 7–60 mostra um temporizador 555 configurado no modo astável (oscilador). Determine a freqüência de saída e o ciclo de trabalho.



▶ FIGURA 7–60

Abra o arquivo F07-60 para verificar a operação.

Solução Use as Equações 7–4 e 7–7.

$$f = \frac{1{,}44}{(R_1 + 2R_2)C_1} = \frac{1{,}44}{(2{,}2\ \text{k}\Omega + 9{,}4\ \text{k}\Omega)0{,}022\ \mu\text{F}} = \mathbf{5{,}64\ kHz}$$

$$\text{Ciclo de trabalho} = \left(\frac{R_1 + R_2}{R_1 + 2R_2}\right)100\% = \left(\frac{2{,}2\ \text{k}\Omega + 4{,}7\ \text{k}\Omega}{2{,}2\ \text{k}\Omega + 9{,}4\ \text{k}\Omega}\right)100\% = \mathbf{59{,}5\%}$$

Problema relacionado Determine o ciclo de trabalho do circuito dado na Figura 7–60 se um diodo for conectado em paralelo com R_2 conforme indicado na Figura 7–59.

SEÇÃO 7–6 REVISÃO

1. Explique a diferença na operação entre um multivibrador astável e um multivibrador monoestável.
2. Para um certo multivibrador astável, t_H = 15 ms e T = 20 ms. Qual o ciclo de trabalho da saída.

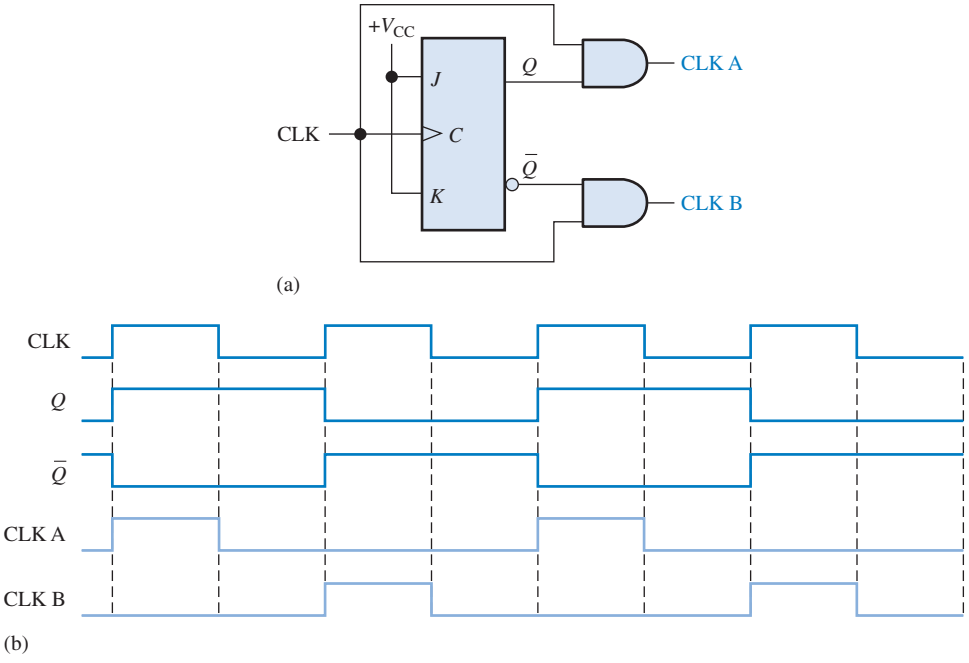
7-7 ANÁLISE DE DEFEITO

É uma prática padrão testar um novo circuito projetado para ter certeza que ele funciona como especificado. Os projetos de funções fixas novas são simulados e testados antes da finalização do projeto. O termo simulação se refere ao método de verificação de um circuito de forma que sua operação possa ser verificada e sanado qualquer defeito antes que o protótipo seja construído.

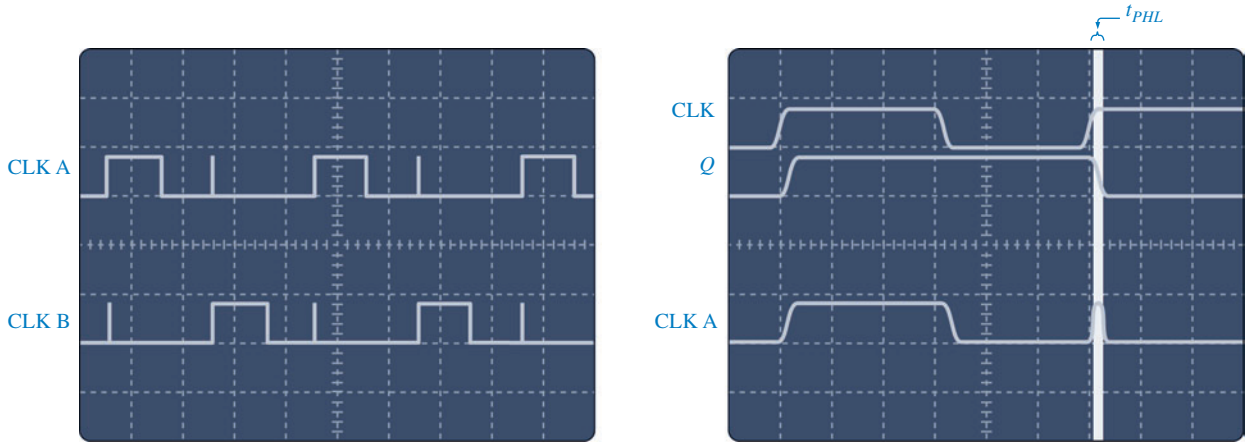
Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever como a temporização de um circuito pode produzir glitches falsos ■ Fazer a análise de defeito de um novo projeto com uma avaliação mais detalhada e consciente dos problemas potenciais

O circuito mostrado na Figura 7–61(a) gera duas formas de onda de clock (CLK A e CLK B) as quais apresentam pulsos alternados. Cada forma de onda é metade da freqüência do clock original (CLK), como mostra o diagrama de temporização ideal na parte (b).



◀ **FIGURA 7–61**
Gerador de clock de duas fases com formas de onda ideais. Abra o arquivo F07-61 e verifique a operação.



(a) Tela de um osciloscópio mostrando as formas de onda CLK A e CLK B com glitches indicados pelos spikes.

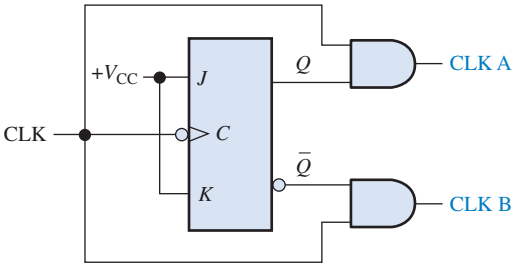
(b) Tela de um osciloscópio mostrando o atraso de propagação que cria glitches na forma de onda CLK A.

▲ FIGURA 7–62

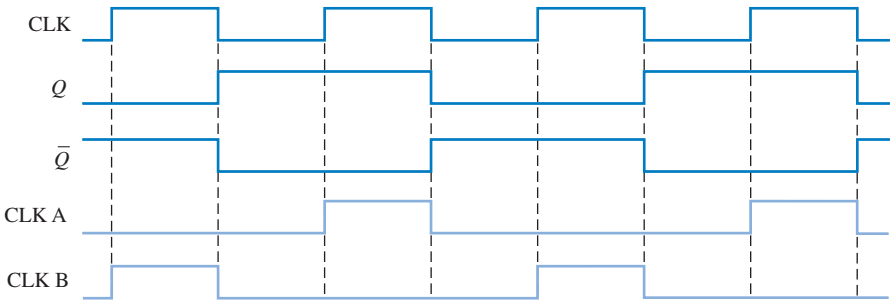
Tela de um osciloscópio para o circuito dado na Figura 7–61.

Quando o circuito é testado com um osciloscópio ou analisador lógico, as formas de onda CLK A e CLK B aparecem na tela como mostra a Figura 7–62(a). Como ocorrem glitches nas duas formas de onda, algo está errado com o circuito no projeto básico ou nas conexões. Investigações posteriores revelam que os glitches são causados por uma condição de **corrida** entre o sinal CLK e os sinais Q e $\overline{Q}$ nas entradas das portas AND. Conforme mostrado na Figura 7–62(b), os atrasos e propagação entre CLK e Q e $\overline{Q}$ criam uma coincidência de curta duração no nível ALTO nas bordas de subida dos pulsos de clock alternados. Portanto, existe uma falha no projeto básico.

O problema pode ser corrigido usando um flip-flop disparado por borda negativa no lugar do dispositivo disparado por borda positiva, como mostra a Figura 7–63(b).



(a)



(b)

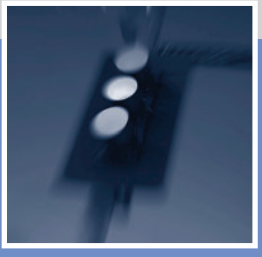
▶ FIGURA 7–63

Gerador de clock de duas fases usando um flip-flop disparado por borda negativa para eliminar glitches. Abra o arquivo F07-63 para verificar a operação.

SEÇÃO 7–7 REVISÃO

1. Um flip-flop D disparado por borda negativa pode ser usado no circuito dado na Figura 7–63?
2. Qual dispositivo pode ser usado para prover o clock para o circuito dado na Figura 7–63?

Os problemas de análise de defeito que são abordados no CD-ROM estão disponíveis na Seção "Prática de Análise de Defeito Usando o Multisim" no final dos problemas do capítulo.



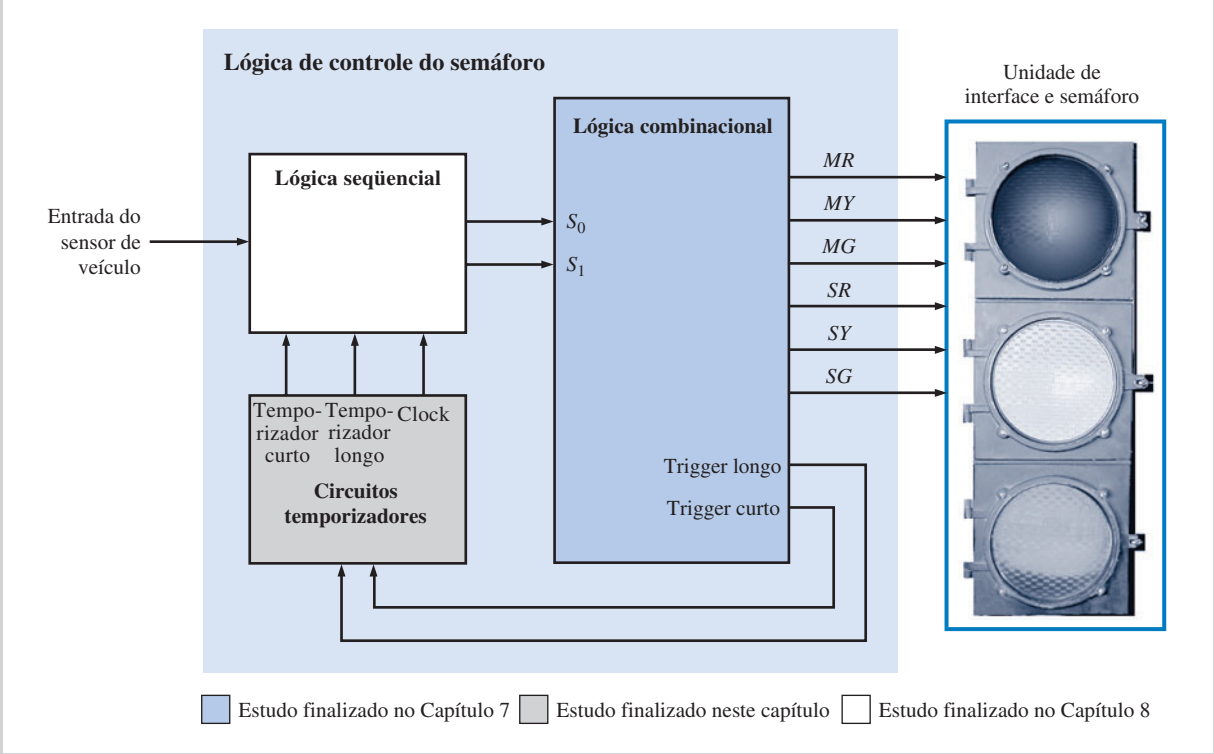
APLICAÇÕES EM SISTEMAS DIGITAIS

O sistema de controle de semáforo que iniciamos no Capítulo 6 tem continuidade neste capítulo. No capítulo anterior, foi desenvolvida a lógica combinacional.

Neste capítulo, são desenvolvidos os circuitos de temporização. Esses circuitos produzem um intervalo de tempo de 4 s para a luz de atenção (amarela) e um intervalo de tempo de 25 s para as luzes vermelha e verde. Além disso, é produzido um sinal de clock pelos circuitos de temporização. O diagrama em bloco geral do sistema de controle de semáforo que foi introduzido no Capítulo 6 é mostrado novamente na Figura 7–64 para consulta.

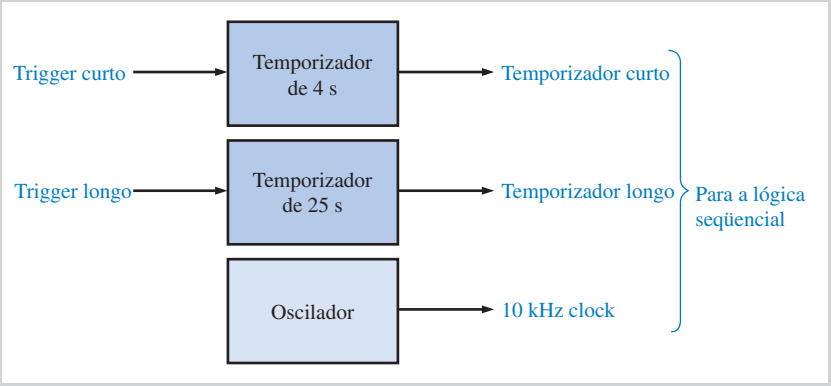
Requisitos do Circuito de Temporização

Os circuitos de temporização consistem de três partes – o temporizador de 4 s, o temporizador de 25 s e o oscilador de 10 kHz – conforme mostrado no diagrama em bloco mostrado na Figura 7–65. Os temporizadores de 4 s e 25 s são implementados com monoestáveis 74121 conforme mos-



▲ **FIGURA 7–64**

Diagrama em bloco do sistema de controle de um semáforo.



▲ **FIGURA 7–65**

Diagrama em bloco dos circuitos de temporização.

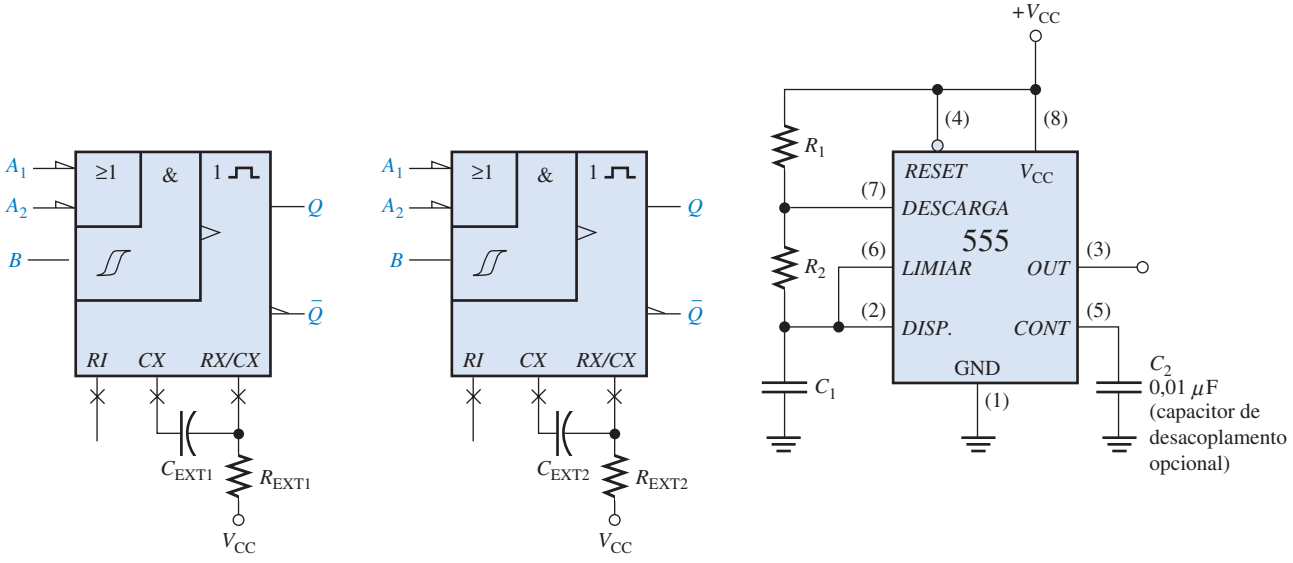
tra a Figura 7–66 (a) e (b).O oscilador de 10 kHz é implementado com um temporizador 555 como mostra a Figura 7–66(c).

Atribuições do Sistema

■ *Atividade 1* Determine os valores de *R* e *C* externos para o temporizador de 4 s dado na Figura 7–66(a).

■ *Atividade 2* Determine os valores de *R* e *C* externos para o temporizador de 25 s dado na Figura 7–66(a).

■ *Atividade 3* Determine os valores de *R* e *C* para o oscilador 555 de 10 kHz dado na Figura 7–66(c).



▲ **FIGURA 7–66**

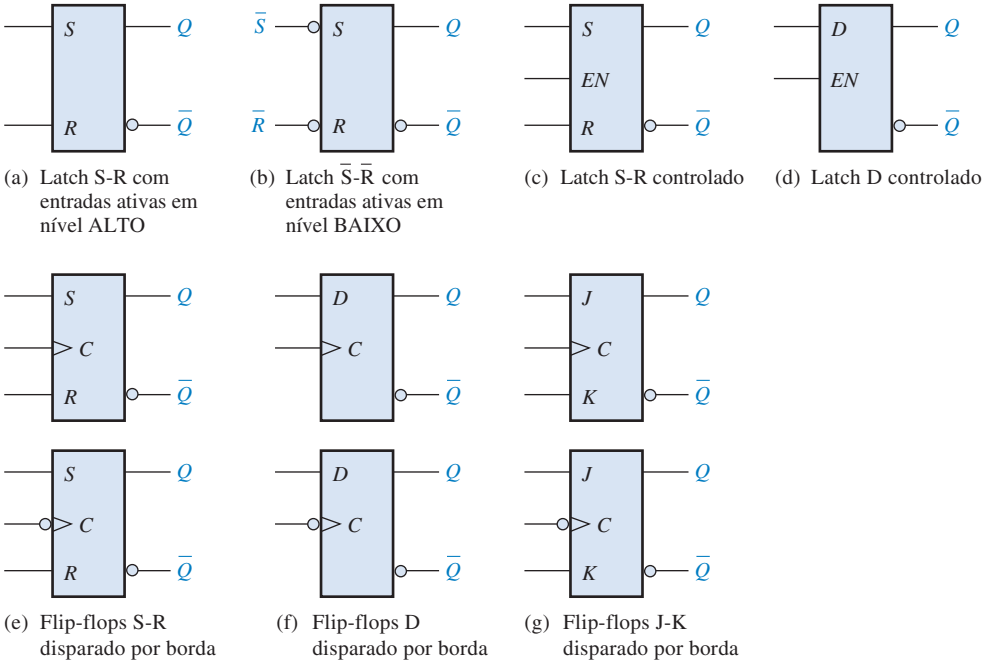
Os circuitos de temporização.

DICA PRÁTICA

Os glitches que ocorrem em sistemas digitais são muito rápidos (de duração extremamente curta) e podem ser difíceis de visualizar num osciloscópio, particularmente em taxas de varreduras mais lentas. Entretanto, um analisador lógico pode mostrar um glitch facilmente. Para procurar glitches usando um analisador lógico, selecione o modo "latch" ou (se disponível) amostragem de transição. No modo latch, o analisador procura por uma mudança de nível de tensão. Quando ocorre uma mudança, mesmo que ela seja de duração extremamente curta (alguns nanossegundos), a informação é "capturada" pela memória do analisador como um outro ponto de dado amostrado. Quando os dados são mostrados, o glitch aparece como uma mudança óbvia nos dados amostrados, sendo fácil identificá-lo.

RESUMO

- Símbolos para latches e flip-flops são mostrados na Figura 7–67.



(a) Latch S-R com entradas ativas em nível ALTO
(b) Latch $\overline{S}$-$\overline{R}$ com entradas ativas em nível BAIXO
(c) Latch S-R controlado
(d) Latch D controlado
(e) Flip-flops S-R disparado por borda
(f) Flip-flops D disparado por borda
(g) Flip-flops J-K disparado por borda

▶ FIGURA 7–67

- Latches são dispositivos biestáveis cujos estados geralmente dependem das entradas assíncronas.
- Flip-flops disparados por borda são dispositivos biestáveis com entradas síncronas cujos estados dependem das entradas apenas no momento da transição ativa do pulso de clock. As mudanças na saída ocorrem na transição ativa do clock.
- Multivibradores monoestáveis têm um estado estável. Quando o monoestável é disparado, a saída vai para o estado instável por um tempo determinado por um circuito *RC*.
- Os multivibradores astáveis não têm estados estáveis e são usados como osciladores para gerar formas de onda de temporização em sistemas digitais.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Astável Não tem estado estável. Um multivibrador astável oscila entre dois estados quase estáveis.

Biestável Tem dois estados estáveis. Flip-flops e latches são multivibradores biestáveis.

Clear Uma entrada assíncrona usada para resetar um flip-flop (faz a saída $Q = 0$).

Clock A entrada de disparo de um flip-flop.

Dissipação de potência A quantidade de energia requerida por um circuito.

Flip-flop D Um tipo de multivibrador biestável no qual a saída assume o estado da entrada *D* na borda de disparo do pulso de clock.

Flip-flop disparado por borda Um tipo de flip-flop no qual os dados são inseridos e aparecem na saída na mesma borda do clock.

Flip-flop J-K Um tipo de flip-flop que pode operar nos modos SET, RESET, repouso e toggle (comutação).

Latch Um circuito digital biestável usado para armazenar um bit.

Monoestável Tem apenas um estado estável. Um multivibrador monoestável que produz um único pulso em resposta a uma entrada de disparo.

Preset Uma entrada assíncrona usada para setar um flip-flop (fazer a saída *Q* igual a 1).

RESET O estado de um flip-flop ou latch quando a saída é 0; a ação que produz o estado de RESET.

SET O estado de um flip-flop ou latch quando a saída é 1; a ação que produz o estado de SET.

Síncrono Tem uma relação de tempo fixo.

Tempo de atraso de propagação O intervalo de tempo necessário após um sinal de entrada ter sido aplicado e a mudança resultante na saída ocorrer.

Tempo de hold O intervalo de tempo necessário para os níveis de controle permanecerem nas entradas de um flip-flop após a borda de disparo do clock para a ativação confiável do dispositivo.

Tempo de setup O intervalo de tempo necessário para os níveis de controle serem colocados nas entradas de um circuito digital, tal como um flip-flop, antes da borda de disparo de um pulso de clock.

Temporizador Um circuito que pode ser usado como um monoestável ou um oscilador.

Toggle A ação de um flip-flop quando comuta de estado a cada pulso de clock.

AUTOTESTE

As respostas estão no final do capítulo.

1. Se um latch S-R tem um nível 1 na entrada *S* e um nível 0 na entrada *R* e em seguida a entrada *S* passa para o nível 0, o latch estará no estado
(a) set **(b)** reset
(c) inválido **(d)** clear

2. O estado inválido de um latch S-R ocorre quando
(a) $S = 1, R = 0$ **(b)** $S = 0, R = 1$
(c) $S = 1, R = 1$ **(b)** $S = 0, R = 0$

3. Para um latch D controlado, a saída *Q* sempre é igual a entrada *D*
(a) antes do pulso de habilitação
(b) durante o pulso de habilitação
(c) imediatamente após o pulso de habilitação
(d) as respostas (b) e (c) estão corretas

4. Assim como o latch, o flip-flop pertence a uma categoria de circuitos lógicos conhecida como
(a) multivibradores monoestáveis **(b)** multivibradores biestáveis
(c) multivibradores astáveis **(d)** monoestáveis

5. A finalidade da entrada de clock num flip-flop é
(a) resetar o dispositivo
(b) setar o dispositivo
(c) sempre provocar uma mudança de estado na saída
(d) fazer com que a saída assuma o estado que depende das entradas de controle (*S-R*, *J-K* ou *D*).

6. Para um flip-flop D disparado por borda,
(a) uma mudança de estado de um flip-flop pode ocorrer apenas na borda de um pulso de clock
(b) o estado para o qual um flip-flop comuta depende da entrada *D*
(c) a saída segue a entrada para cada pulso de clock
(d) todas as alternativas anteriores estão corretas

7. Uma característica que distingue o flip-flop J-K do flip-flop S-R é
(a) a condição toggle
(b) a entrada preset
(c) o tipo de clock
(d) a entrada clear

8. Um flip-flop está na condição toggle quando
(a) $J = 1, K = 0$ **(b)** $J = 1, K = 1$
(c) $J = 0, K = 0$ **(d)** $J = 0, K = 1$

9. Um flip-flop J-K com $J = 1$ e $K = 1$ tem uma entrada de clock de 10 kHz. A saída *Q* é
(a) constantemente nível ALTO
(b) constantemente nível BAIXO
(c) uma onda quadrada de 10 kHz
(d) uma onda quadrada de 5 kHz

10. Um monoestável é um tipo de
(a) multivibrador com um estado estável **(b)** multivibrador astável **(c)** temporizador
(d) as respostas (a) e (c) estão corretas **(e)** as respostas (b) e (c) estão corretas

11. A largura do pulso de saída de um monoestável não-redisparável depende
(a) dos intervalos de disparo **(b)** da tensão de alimentação
(c) de um resistor e um capacitor **(d)** da tensão de limiar

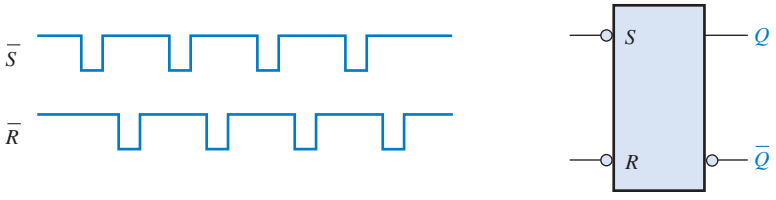
12. Um multivibrador astável
(a) requer uma entrada e disparo periódica **(b)** não tem estado estável
(c) é um oscilador **(d)** produz uma saída de pulsos periódica
(e) as respostas (a), (b), (c) e (d) estão corretas **(f)** as respostas (b), (c) e (d) estão corretas

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

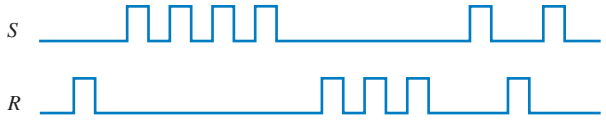
SEÇÃO 7–1 Latches

1. Se as formas de onda vistas na Figura 7–68 são aplicadas no latch com entradas ativas em nível BAIXO, desenhe a forma de onda da saída Q resultante em relação às entradas. Considere a saída Q iniciando em nível BAIXO.



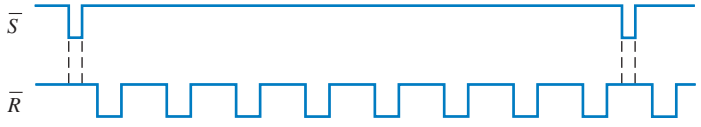
▶ FIGURA 7–68

2. Resolva o Problema 1 para as formas de onda dadas na Figura 7–69 aplicadas num latch S-R com entradas ativas em nível ALTO.



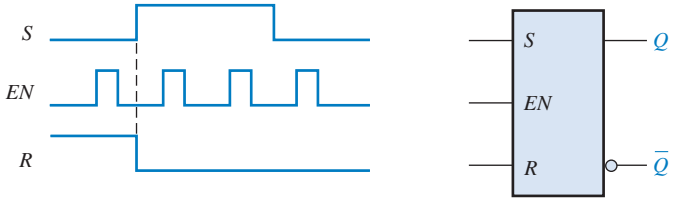
▶ FIGURA 7–69

3. Resolva o Problema 1 para as formas de onda de entrada dadas na Figura 7–70.



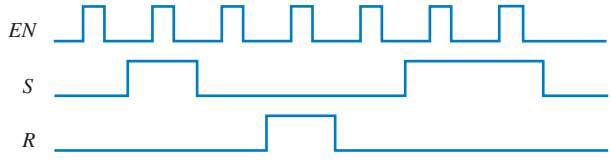
▶ FIGURA 7–70

4. Para um latch S-R controlado, determine as saídas Q e $\overline{Q}$ para as entradas dadas na Figura 7–71. Mostre-as em relação à entrada de habilitação. Considere a saída Q iniciando em nível BAIXO.

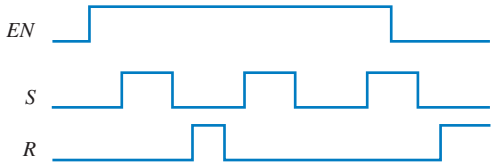


▶ FIGURA 7–71

5. Resolva o Problema 4 para as entradas dadas na Figura 7–72.
6. Resolva o Problema 4 para as entradas dadas na Figura 7–73.

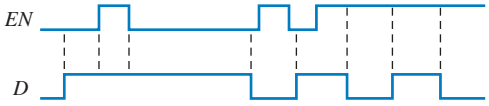


▲ FIGURA 7–72



▲ FIGURA 7–73

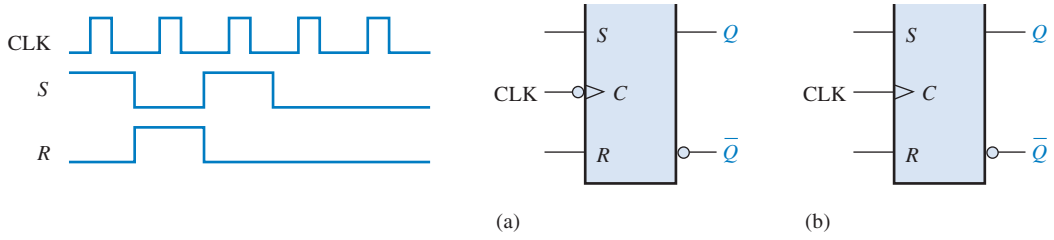
7. Para um latch controlado, as formas de onda mostradas na Figura 7–74 são observadas em suas entradas. Desenhe o diagrama de temporização mostrando a forma de onda de saída que você espera ver em *Q* se o latch estiver inicialmente *resetado*.



▶ FIGURA 7–74

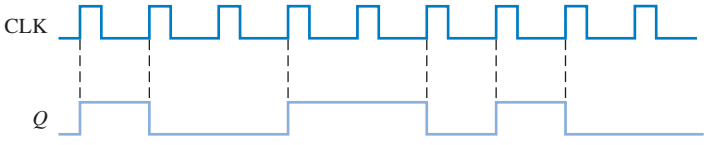
SEÇÃO 7–2 Flip-Flops Disparados por Borda

8. A Figura 7–75 mostra dois flip-flops S-R disparados por borda. Se as entradas são como mostra a figura, desenhe a saída *Q* de cada flip-flop em relação ao clock e explique a diferença entre os dois. Os flip-flops estão inicialmente resetados.



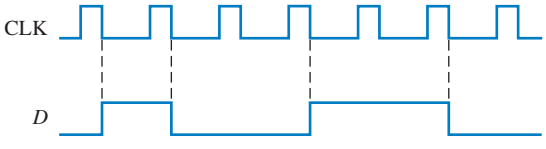
▲ FIGURA 7–75

9. A saída *Q* de um flip-flop S-R disparado por borda é mostrada em relação ao sinal de clock na Figura 7–76. Determine as formas de onda de entrada nas entradas *S* e *R* que são necessárias para produzir essa saída se o flip-flop é do tipo disparado por borda positiva.



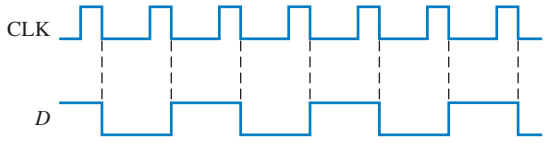
▶ FIGURA 7–76

10. Desenhe a saída *Q* em relação ao clock para um flip-flop D com as entradas conforme mostra a Figura 7–77. Considere a entrada de clock ativa na borda de subida e a saída *Q* inicialmente em nível BAIXO.



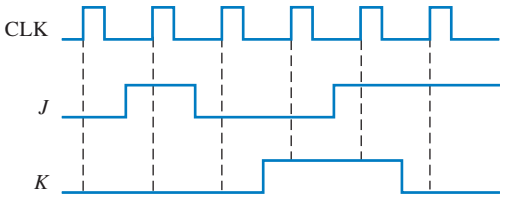
▶ FIGURA 7–77

11. Resolva o Problema 10 para as entradas dadas na Figura 7–78.



▲ **FIGURA 7–78**

12. Para um flip-flop J-K disparado por borda positiva com entradas conforme mostra a Figura 7–79, determine a saída *Q* em relação ao clock. Considere que a saída *Q* começa em nível BAIXO.

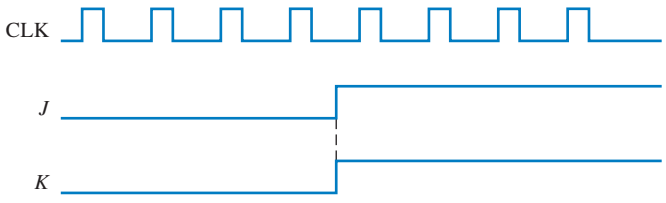


▶ **FIGURA 7–79**

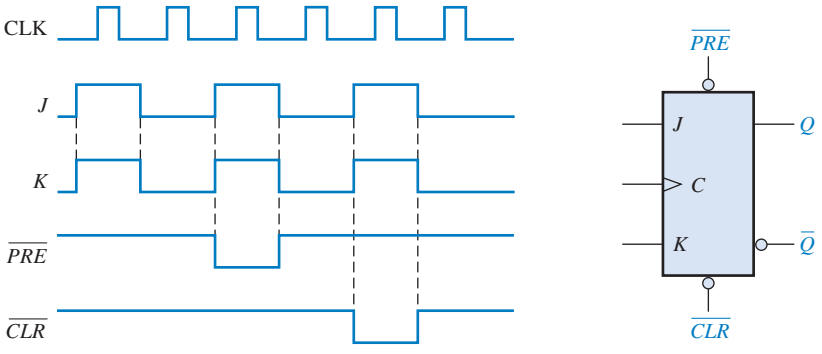
13. Resolva o Problema 12 para as entradas dadas na Figura 7–80.

14. Determine a forma de onda de *Q* em relação ao clock se os sinais mostrados na Figura 7–81 forem aplicados nas entradas do flip-flop J-K. Considere a saída *Q* inicialmente em nível BAIXO.

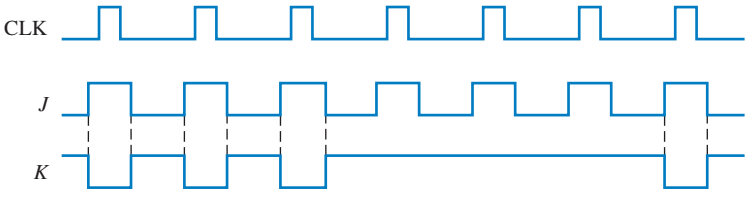
15. Para um flip-flop J-K disparado por borda negativa com as entradas dadas na Figura7–82, desenvolva a forma de onda da saída *Q* em relação ao clock. Considere a saída *Q* inicialmente em nível BAIXO.



▶ **FIGURA 7–80**



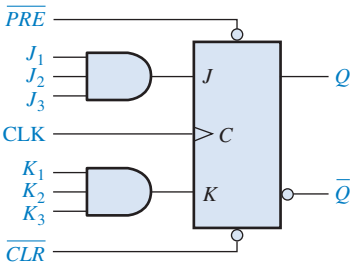
▶ **FIGURA 7–81**



▶ **FIGURA 7–82**

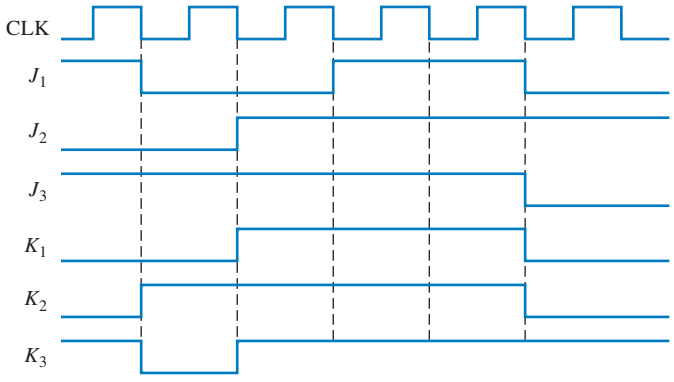
16. Os seguintes dados em série são aplicados ao flip-flop através das portas AND como indicado na Figura 7–83. Determine os dados em série resultantes que aparecem na saída Q. Existe um pulso de clock para cada tempo de bit. Considere a saída Q inicialmente em 0 e que $\overline{PRE}$ e $\overline{CLR}$ estejam em nível ALTO. Os bits mais à direita são aplicados primeiro.

J_1: 1 0 1 0 0 1 1
J_2: 0 1 1 1 0 1 0
J_3: 1 1 1 1 0 0 0
K_1: 0 0 0 1 1 1 0
K_2: 1 1 0 1 1 0 0
K_3: 1 0 1 0 1 0 1



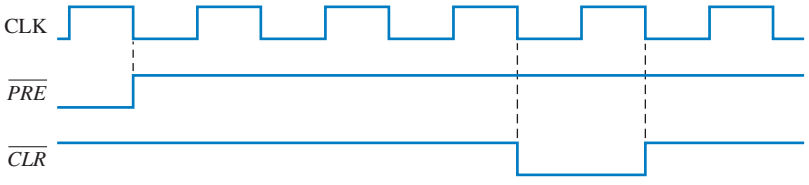
▶ FIGURA 7–83

17. Para o circuito dado na Figura 7–83, complete o diagrama de temporização dado na Figura 7–84 mostrando a saída Q (que está inicialmente em nível BAIXO). Considere que $\overline{PRE}$ e $\overline{CLR}$ permanecem em nível ALTO.



▶ FIGURA 7–84

18. Resolva o Problema 17 com as mesmas entradas J e K porém as entradas $\overline{PRE}$ e $\overline{CLR}$ como mostrado na Figura 7–85 em relação ao clock.



▶ FIGURA 7–85

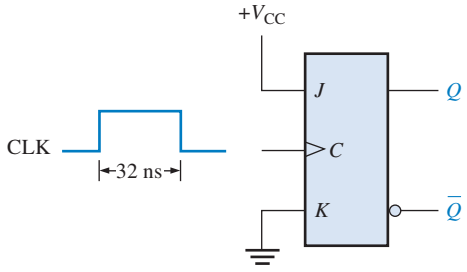
SEÇÃO 7–3 Características de Operação dos Flip-Flops

19. O que determina a dissipação de potência de um flip-flop?

20. As folhas de dados de um fabricante especifica tipicamente quatro diferentes tempos de atrasos de propagação associados a um flip-flop. Nomeie e descreva cada um deles.

21. As folhas de dados de um certo flip-flop especifica que o tempo em nível ALTO mínimo para o pulso de clock é 30 ns e que o tempo em nível BAIXO mínimo é 37 ns. Qual a freqüência máxima de operação?

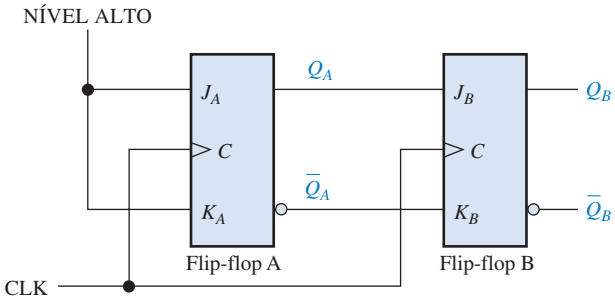
22. O flip-flop mostrado na Figura 7–86 está inicialmente resetado. Mostre a relação entre a saída Q e o pulso de clock se o atraso de propagação t_{PLH} (clock para Q) é 8 ns.



▶ **FIGURA 7–86**

23. A corrente direta necessária para um flip-flop em particular que opera com uma fonte de +5 V cc é 10 mA. Um certo dispositivo digital usa 15 desses flip-flops. Determine a capacidade de corrente necessária para a fonte de +5 V cc e a dissipação de potência total do sistema.

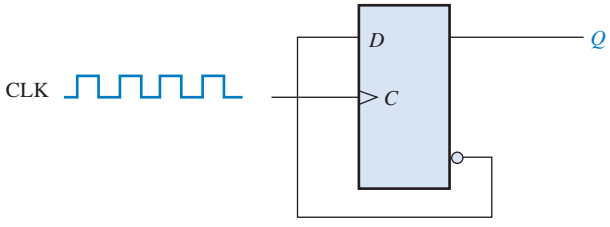
24. Para o circuito dado na Figura 7–87, determine a freqüência máxima do sinal de clock para uma operação confiável se o tempo de setup de cada flip-flop é de 2 ns e os atrasos de propagação (t_{PLH} e t_{PHL}) a partir do clock para a saída são de 5 ns para cada flip-flop.



▶ **FIGURA 7–87**

SEÇÃO 7–4 Aplicações de Flip-Flops

25. Um flip-flop D é conectado como mostra a Figura 7–88. Determine a saída Q em relação ao clock. Qual a função específica que esse dispositivo realiza?



▶ **FIGURA 7–88**

26. Para o circuito dado na Figura 7–87, desenvolva um diagrama de temporização para oito pulsos de clock, mostrando as saídas Q_A e Q_B em relação ao clock.

SEÇÃO 7–5 Monoestáveis

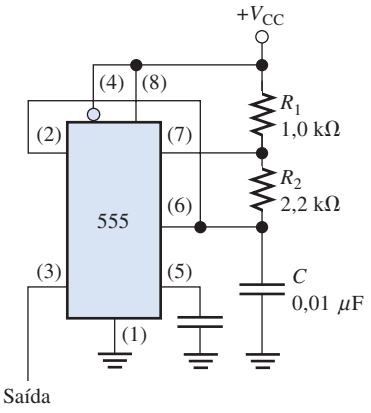
27. Determine a largura de pulso de um CI monoestável 74121 se o resistor externo for 3,3 kΩ e o capacitor externo for 2000 pF.

28. Um pulso de saída de 5μs de duração deve ser gerado por um CI monoestável 74LS122. Usando um capacitor de 10000 pF, determine o valor da resistência externa necessária.

SEÇÃO 7–6 Temporizador 555

29. Projete um monoestável usando um temporizador 555 que produza um pulso de saída de 0,25 s.

30. Um temporizador 555 é configurado para funcionar como um multivibrador astável como mostra a Figura 7–89. Determine a freqüência dele.

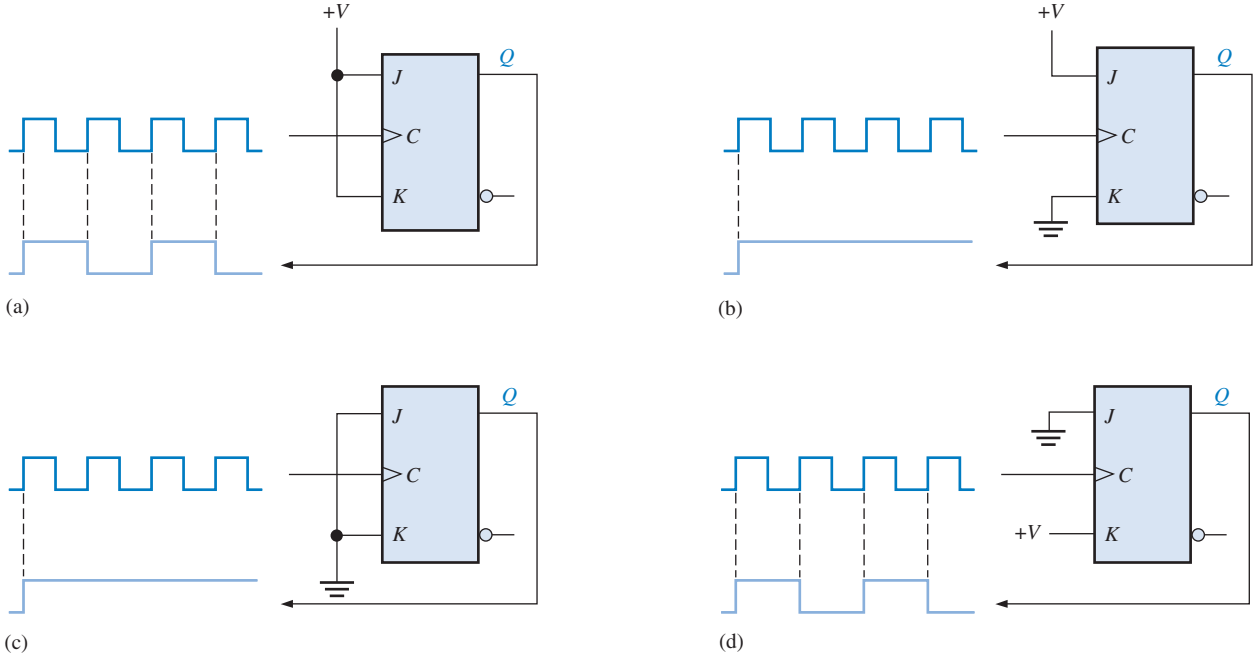


▶ FIGURA 7–89

31. Determine os valores dos resistores externos para um temporizador 555 usado como um multivibrador astável com uma freqüência de saída de 20 kHz, se o capacitor externo C é de 0,002μF e o ciclo de trabalho é aproximadamente 75%?

SEÇÃO 7–7 Análise de Defeito

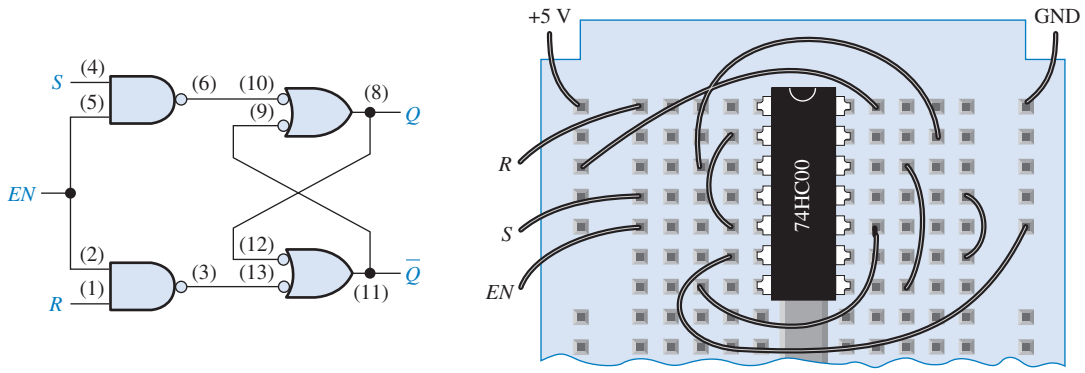
32. O flip-flop mostrado na Figura 7–90 é testado em todas as condições de entrada conforme mostrado. Ele está operando adequadamente? Em caso negativo, qual é o defeito mais provável?



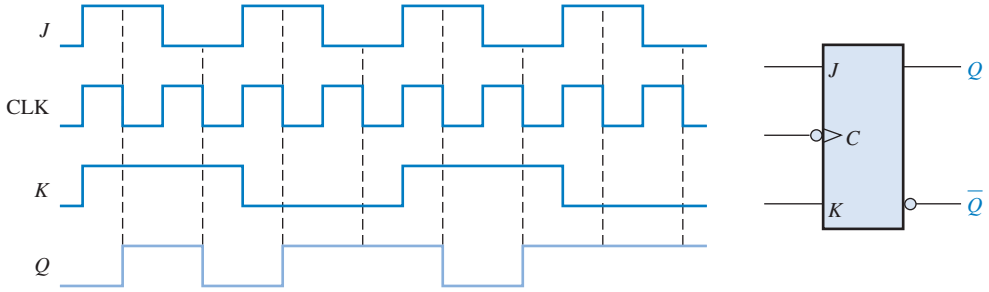
▲ FIGURA 7–90

33. Um CI de quatro portas NAND (74HC00) é usado para construir um latch S-R controlado num protoboard num laboratório de eletrônica como mostra a Figura 7–91. O esquemático na parte (a) é usado para montar o circuito na parte (b). Quando se coloca o latch em operação, verifica-se que a saída Q permanece em nível ALTO não importando como as entradas estão. Determine o problema.

34. Determine se o flip-flop mostrado na Figura 7–92 está operando corretamente e, em caso negativo, identifique o defeito mais provável.



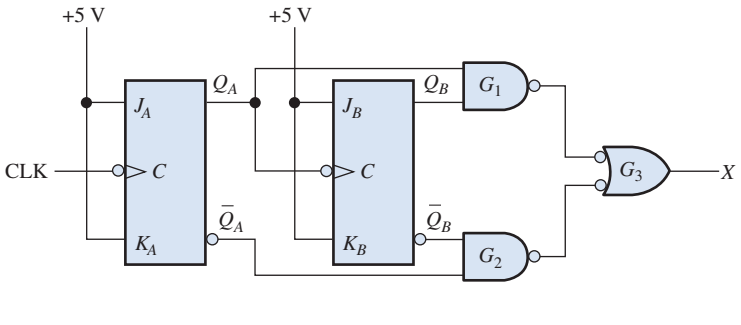
▲ **FIGURA 7–91**



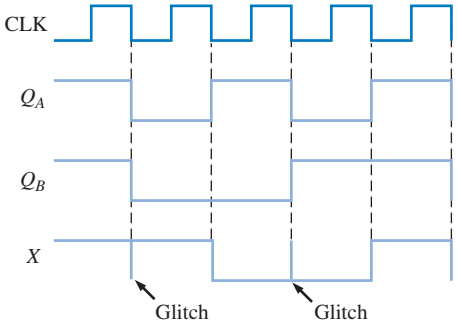
▲ **FIGURA 7–92**

35. O circuito de armazenamento de dados em paralelo mostrado na Figura 7–36 não está funcionando corretamente. Para verificar isso, primeiro é verificado as conexões de V_{CC} e GND e em seguida aplica-se nível BAIXO em todas as entradas D e aplica-se um pulso na linha de clock. Verifica-se que as saídas Q estão todas em nível BAIXO; até agora, tudo bem. Em seguida, aplica-se nível ALTO em todas as entradas D e novamente aplica-se um pulso na linha de clock. Quando se verifica as saídas Q, encontra-se todas em nível BAIXO. Qual o problema e qual procedimento você usaria para isolar o defeito a um único dispositivo?

36. O circuito do flip-flop mostrado na Figura 7–93(a) é usado para gerar uma seqüência de contagem binária. As portas formam um decodificador que supostamente produz um nível ALTO quando ocorre um binário zero ou um dos três estados binários (00 a 11). Quando se verifica as saídas Q_A e Q_B, encontramos o que podemos ver na parte (b) da figura, a qual revela glitches na saída do decodificador (X) além dos pulsos corretos. O que está causando os glitches e como eles podem ser eliminados?



(a)



(b)

▲ **FIGURA 7–93**

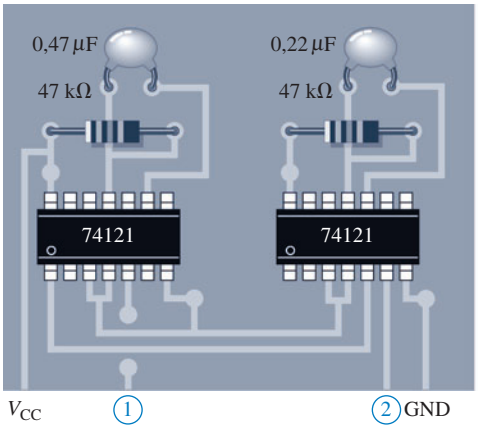
37. Determine as saídas Q_A, Q_B e X ao longo de seis pulsos de clock no circuito dado na Figura 7–93(a) para cada um dos seguintes defeitos no circuito TTL. Comece com Q_A e Q_B em nível BAIXO.

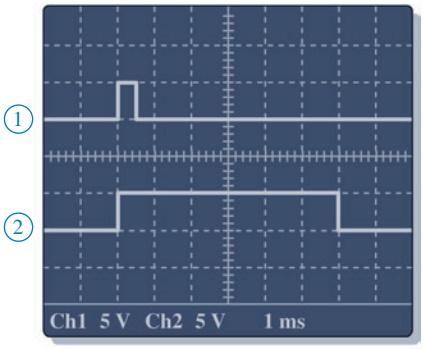
(a) entrada J_A aberta **(b)** entrada K_B aberta

(c) saída Q_B aberta **(d)** entrada de clock do flip-flop B curto-circuitada

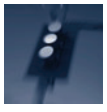
(e) saída da porta G_2 aberta

38. Dois CIs monoestáveis 74121 são conectados numa placa de circuito como mostra a Figura 7–94. após observar a tela do osciloscópio você conclui que o circuito está operando corretamente? Em caso negativo, qual deve ser o problema mais provável?





▶ **FIGURA 7–94**



Aplicações em Sistemas Digitais

39. Use temporizadores 555 para implementar monoestáveis de 4 s e 25 s para os circuitos de temporização do sistema de controle de semáforo. A entrada de disparo do 555 não pode permanecer em nível BAIXO após a transição negativa, assim deve-se desenvolver um circuito que produza um pulso negativo (borda de descida) muito estreito para disparar os temporizadores longo e curto quando o sistema for para cada estado.

Problemas Especiais de Projeto

40. Projete um circuito de contagem básico que produza uma seqüência binária de zero a sete usando flip-flops J-K disparados por borda negativa.

41. No departamento de embalagem de uma fábrica de bolas, as bolas caem num transportador e seguem em fila através de uma rampa até às caixas para serem embaladas. Cada bola que passa na rampa ativa uma chave que produz um pulso elétrico. A capacidade de cada caixa é de 32 bolas. Projete um circuito lógico para indicar quando uma caixa está cheia de forma que uma caixa vazia possa ser movida para a posição da extremidade da rampa.

42. Faça uma lista com as alterações de projeto que seriam necessárias no sistema de controle de semáforo para acrescentar 15 s à seta de conversão à esquerda na via principal. A seta de conversão ocorre após a luz vermelha e antes da luz verde. Modifique o diagrama de estado do Capítulo 6 para mostrar essas alterações.

Prática de Análise de Defeito Usando o Multisim

43. Abra o arquivo P07-43 e teste os latches para determinar qual deles está com defeito.

44. Abra o arquivo P07-44 e teste os flip-flops J-K para determinar qual deles está com defeito.

45. Abra o arquivo P07-45 e teste os flip-flops D para determinar qual deles está com defeito.

46. Abra o arquivo P07-46 e teste os monoestáveis para determinar qual deles está com defeito.

47. Abra o arquivo P07-47 e teste o circuito divisor por quatro para determinar se existe um defeito. Em caso afirmativo, identifique-o se possível.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 7–1 Latches

1. Três tipos de latches são S-R, S-R controlado e D controlado.

2. $SR = 00$, NC; $SR = 01$, $Q = 0$; $SR = 10$, $Q = 1$; $SR = 11$, inválido.

3. $Q = 1$

SEÇÃO 7–2 Flip-Flops Disparados por Borda

1. A saída de um latch S-R controlado pode mudar em qualquer momento que a entrada de habilitação (*EN*) da porta estiver ativa. A saída de um flip-flop S-R disparado por borda pode mudar apenas na borda de disparo de um pulso de clock.

2. O flip-flop J-K não tem estado inválido como o flip-flop S-R.

3. A saída Q vai para nível ALTO na borda de subida do primeiro pulso de clock, vai para nível BAIXO na borda de subida do segundo pulso, vai para nível ALTO na borda de subida do terceiro pulso e vai para nível BAIXO na borda de subida do quarto pulso.

SEÇÃO 7–3 Características de Operação dos Flip-Flops

1. **(a)** O tempo de setup é o tempo necessário em que os dados na entradas devem estar presentes antes da borda de disparo do pulso de clock.

(b) Tempo de hold é o tempo necessário em que os dados devem permanecer nas entradas após a borda de disparo do pulso de clock.

2. O CI 74AHC74 pode operar na maior freqüência, de acordo com a Tabela 7–5.

SEÇÃO 7–4 Aplicações de Flip-Flops

1. Um grupo de flip-flops de armazenamento de dados é um registrador.

2. Para a operação de divisão por 2 o flip-flop tem que ser toggle ($J = 1$, $K = 1$).

3. Seis flip-flops são usados num divisor por 64.

SEÇÃO 7–5 Monoestáveis

1. Um monoestável não-redisparável finaliza a temporização antes de responder a um outro disparo na entrada. Um monoestável redisparável responde a cada disparo na entrada.
2. A largura de pulso é definida pelos componentes *R* e *C* externos.

SEÇÃO 7–6 Temporizador 555

1. Um multivibrador astável não tem estado estável. Um multivibrador monoestável tem um estado estável.
2. Ciclo de trabalho = (15 ms/20 ms)100% = 75%

SEÇÃO 7–7 Análise de Defeito

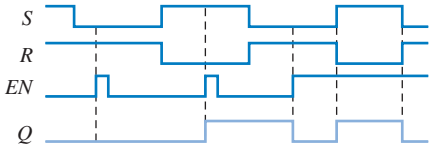
1. Sim, um flip-flop D disparado por borda negativa pode ser usado.
2. Um multivibrador astável usando um temporizador 555 pode ser usado para fornecer o clock.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

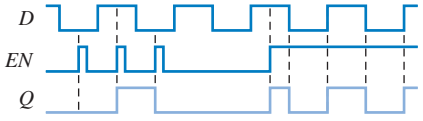
7–1. A saída *Q* é a mesma como mostra a Figura 7–5(b).

7–2. Veja a Figura 7–95. **7–3.** Veja a Figura 7–96.

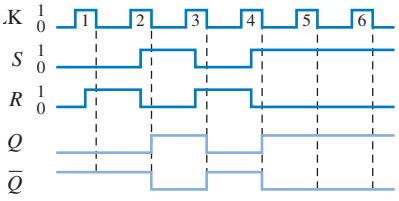
7–4. Veja a Figura 7–97. **7–5.** Veja a Figura 7–98.



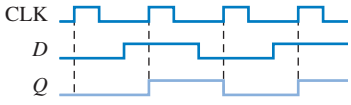
▲ FIGURA 7–95



▲ FIGURA 7–96



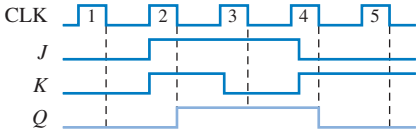
▲ FIGURA 7–97



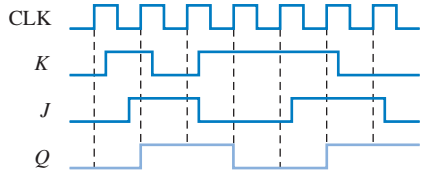
▲ FIGURA 7–98

7–6. Veja a Figura 7–99. **7–7.** Veja a Figura 7–100.

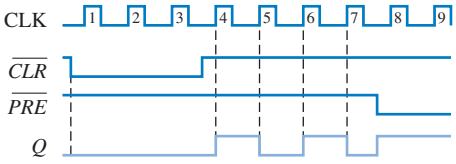
7–8. Veja a Figura 7–101. **7–9.** Veja a Figura 7–102.



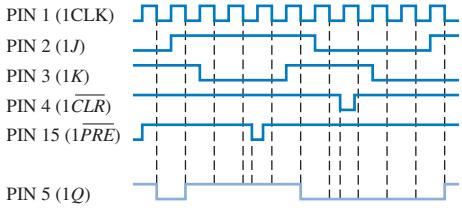
▲ **FIGURA 7–99**



▲ **FIGURA 7–100**



▲ **FIGURA 7–101**



▲ **FIGURA 7–102**

7–10. $2^5 = 32$. São necessários cinco flip-flops.

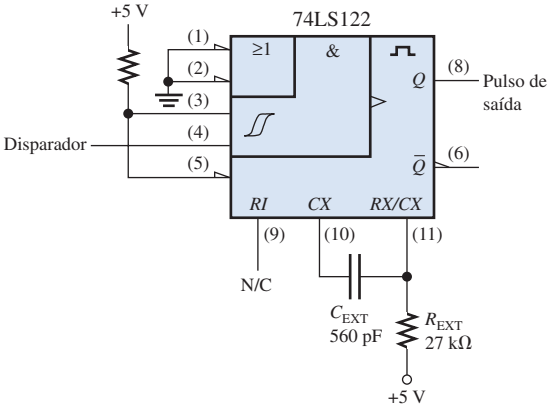
7–11. Dezesseis estados necessitam de quatro flip-flops ($2^4 = 16$).

7–12. C_{EXT} = 7143 pF conectado de *CX* para *RX*/*CX* do CI 74121.

7–13. C_{EXT} = 560 pF, R_{EXT} = 27 kΩ. Veja a Figura 7–103.

7–14. R_1 = 91 kΩ

7–15. Ciclo de trabalho ≅ 32%



▶ **FIGURA 7–103**

AUTOTESTE

1. (a) **2.** (c) **3.** (d) **4.** (b) **5.** (d) **6.** (d) **7.** (a) **8.** (b)
9. (d) **10.** (d) **11.** (c) **12.** (f)

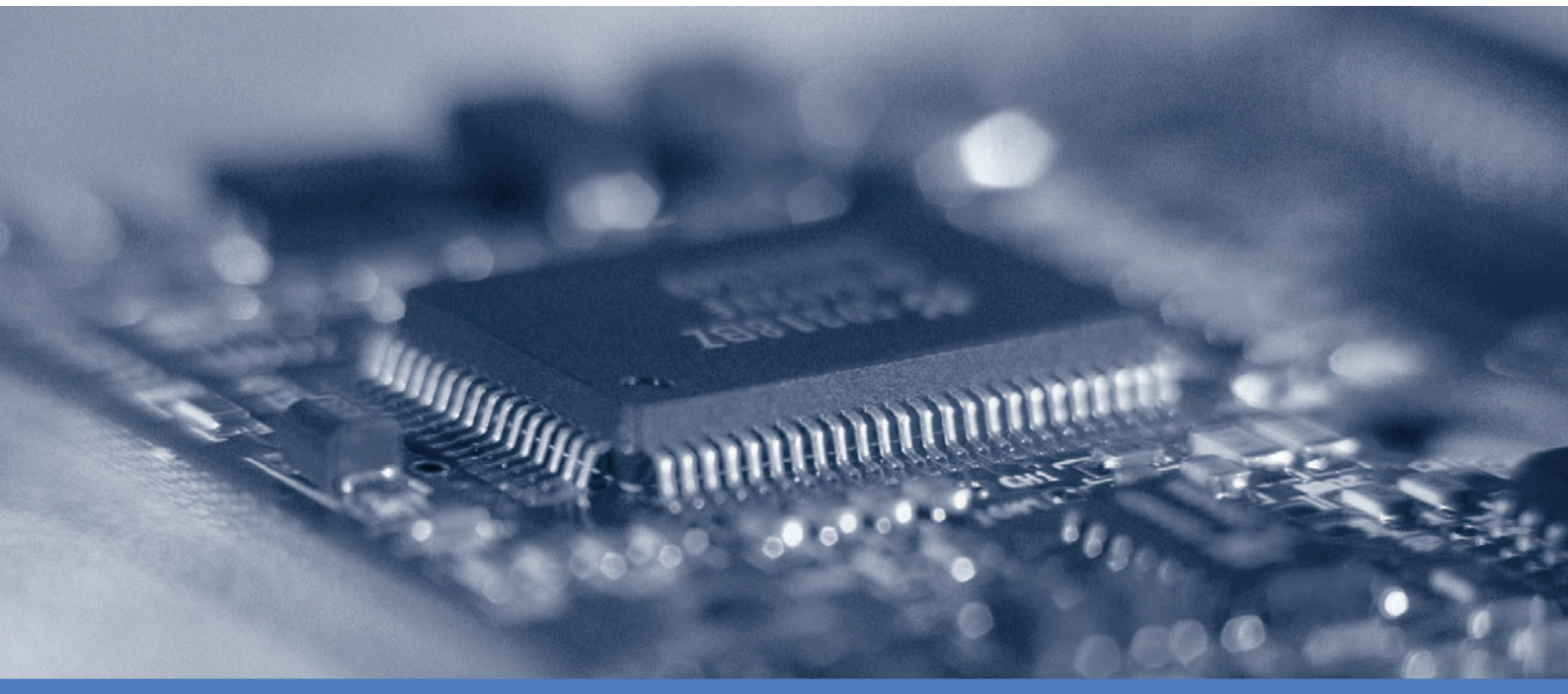
8

CONTADORES

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Descrever a diferença entre um contador assíncrono e um síncrono
- Analisar diagramas de temporização de contadores
- Analisar circuitos contadores
- Explicar como os atrasos de propagação afetam a operação de um contador
- Determinar o módulo de um contador
- Modificar o módulo de um contador
- Reconhecer a diferença entre um contador binário de 4 bits e um contador de década



- Usar um contador crescente/decrescente para gerar seqüências binárias diretas e inversas
- Determinar a seqüência de um contador
- Usar CIs contadores em diversas aplicações
- Projetar um contador que tenha uma seqüência especificada de estados
- Usar contadores em cascata para obter um módulo maior
- Usar portas lógicas para decodificar qualquer estado especificado de um contador
- Eliminar glitches na decodificação de um contador
- Explicar como funciona um relógio digital
- Interpretar os símbolos lógicos de contadores que usam notação de dependência
- Realizar a análise de defeito em circuitos de contadores com diversos tipos de defeitos

TERMOS IMPORTANTES

- Assíncrono
- Reciclagem
- Módulo
- Década
- Síncrono
- Contagem final
- Máquina de estados
- Diagrama de estados
- Conexão em cascata

INTRODUÇÃO

Conforme estudamos no Capítulo 7, os flip-flops podem ser conectados juntos para realizar operações de contagem. Tal grupo de flip-flops é um contador. O número de flip-flops usados e a forma na qual eles são conectados determinam o número de estados (denominado módulo) e também a seqüência específica de estados que o contador percorre durante cada ciclo completo.

Os contadores são classificados em duas grandes categorias de acordo com a forma que eles recebem os pulsos de o clock: assíncronos e síncronos. Nos contadores assíncronos, normalmente chamados de *contadores ondulantes* (*ripple counters*), o primeiro flip-flop recebe o clock por meio de um pulso de clock externo e cada flip-flop sucessivo recebe o clock através da saída do flip-flop anterior. Em contadores síncronos, a entrada de clock é conectada a todos os flip-flops de forma que eles recebem o clock simultaneamente. Dentro de cada uma dessas categorias, os contadores são classificados principalmente pelo tipo de seqüência, o número de estados, ou o número de flip-flops no contador.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

74XX93	74XX161	74XX162
74XX163	74XX190	74XX47

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

Esse tópico de Aplicações em Sistemas Digitais ilustra os conceitos desse capítulo. Continuamos com o desenvolvimento do sistema de controle de semáforo abordado nos dois últimos capítulos. O foco aqui é a parte da lógica seqüencial do sistema que produz a base da seqüência do semáforo para as entradas dos circuitos de temporização e sensor de veículos. As partes do sistema desenvolvidas nos Capítulos 6 e 7 são combinadas com a lógica seqüencial para completar o sistema.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

8-1 OPERAÇÃO DE CONTADORES ASSÍNCRONOS

O termo **assíncrono** se refere aos eventos que não têm uma relação temporal fixa entre si e, geralmente, não ocorrem ao mesmo tempo. Um **contador assíncrono** é aquele no qual os flip-flops (FF) que constituem o contador não mudam de estado exatamente ao mesmo tempo porque eles não têm um pulso de clock comum.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a operação de um contador binário assíncrono de 2 bits ■ Descrever a operação de um contador binário assíncrono de 3 bits ■ Definir o termo *ondulante* (*ripple*) em relação a contadores ■ Descrever a operação de um contador de década assíncrono ■ Desenvolver diagramas de temporização para contadores ■ Discutir o CI contador binário assíncrono de 4 bits 74LS93.

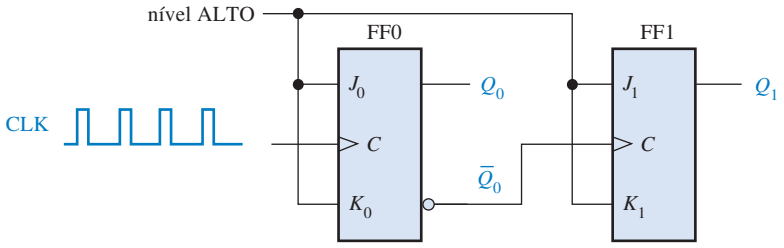
Contador Binário Assíncrono de 2 Bits

A entrada de clock de um contador assíncrono é sempre conectada apenas no flip-flop LSB.

A Figura 8–1 mostra um contador de 2 bits conectado para operação assíncrona. Observe que o clock (CLK) é aplicado na entrada de clock (*C*) *apenas* do primeiro flip-flop (FF0) que é sempre o bit menos significativo (LSB). O segundo flip-flop (FF1) é disparado pela saída $\overline{Q}_0$ do FF0. O FF0 muda de estado na borda positiva de cada pulso de clock, porém FF1 muda apenas quando disparado por uma transição positiva da saída $\overline{Q}_0$ do FF0. Devido ao tempo de atraso de propagação inerente através de um flip-flop, a transição do pulso de clock (CLK) de entrada e a transição da saída $\overline{Q}_0$ do FF0 nunca podem mudar exatamente ao mesmo tempo. Portanto, os dois flip-flops nunca são disparados simultaneamente. Assim, a operação do contador é assíncrona.

► FIGURA 8–1

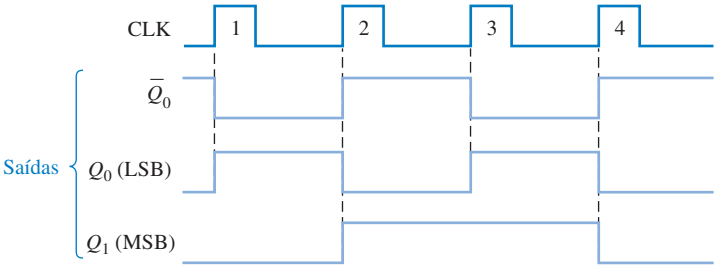
Um contador binário de 2 bits assíncrono. Abra o arquivo F08-01 para verificar a operação.



O Diagrama de Temporização Vamos examinar a operação básica do contador assíncrono apresentado na Figura 8–1 aplicando quatro pulsos de clock em FF0 e observando a saída *Q* de cada flip-flop. A Figura 8–2 ilustra as mudanças de estado nas saídas dos flip-flops em resposta aos pulsos de clock. Os dois flip-flops são conectados para operação toggle ($J = 1$, $K = 1$) e considera-se que estejam inicialmente resetados (*Q* em nível BAIXO).

► FIGURA 8–2

Diagrama de temporização para o contador visto na Figura 8–1. Assim como em capítulos anteriores, as formas de onda de saída são mostradas em laranja claro.



A borda positiva de CLK1 (pulso de clock 1) faz com que a saída Q_0 do FF0 vá para o nível ALTO, conforme mostra a Figura 8–2. No mesmo instante a saída $\overline{Q}_0$ vai para nível BAIXO, porém isso não tem efeito no FF1 porque o disparo do flip-flop é na borda positiva. Após a borda de subida de CLK1, $Q_0 = 1$ e $Q_1 = 0$. A borda positiva de CLK2 faz Q_0 comutar para o nível BAIXO. A saída $\overline{Q}_0$ vai para o nível ALTO e dispara FF1, fazendo Q_1 comutar para o nível ALTO. Após a borda de subida de CLK2, $Q_0 = 0$ e $Q_1 = 1$. A borda positiva de CLK3 faz Q_0 comutar para o nível ALTO novamente. A saída Q_0 comuta para o nível BAIXO, não afetando FF1. Portanto, após a borda de subida de CLK3, $Q_0 = 1$ e $Q_1 = 1$. A borda positiva de CLK4 faz Q_0 comutar para o nível BAIXO, enquanto que $\overline{Q}_0$ comuta para o nível ALTO e dispara FF1, fazendo Q_1 comutar para o nível BAIXO. Após a borda de subida de CLK4, $Q_0 = 0$ e $Q_1 = 0$. Agora o contador reciclou para o seu estado original (os dois flip-flops estavam resetados).

Contadores assíncronos também são conhecidos como contadores ondulantes (*ripple counter*).

O diagrama de temporização mostra as formas de onda das saídas Q_0 e Q_1 em relação aos pulsos de clock conforme ilustrado na Figura 8–2. Por questão de simplicidade, as transições de Q_0, Q_1 e dos pulsos de clock são mostradas como eventos simultâneos nesse contador assíncrono. Existe, é claro, algum pequeno atraso entre as transições de CLK e Q_0 e entre as transições de $\overline{Q}_0$ e Q_1.

Observe na Figura 8–2 que o contador de 2 bits exibe quatro estados diferentes, conforme se poderia esperar com dois flip-flops ($2^2 = 4$). Observe também que se Q_0 representa o bit menos significativo (LSB) e Q_1 representa o bit mais significativo (MSB), a seqüência dos estados do contador representa uma seqüência de números binários conforme listado na Tabela 8–1.

Em lógica digital, Q_0 sempre é o LSB, exceto se especificado de outra forma.

TABELA 8–1

Seqüência de estados binários para o contador apresentado na Figura 8–1

PULSO DE CLOCK	Q_1	Q_0
Valor inicial	0	0
1	0	1
2	1	0
3	1	1
4 (recicla)	0	0

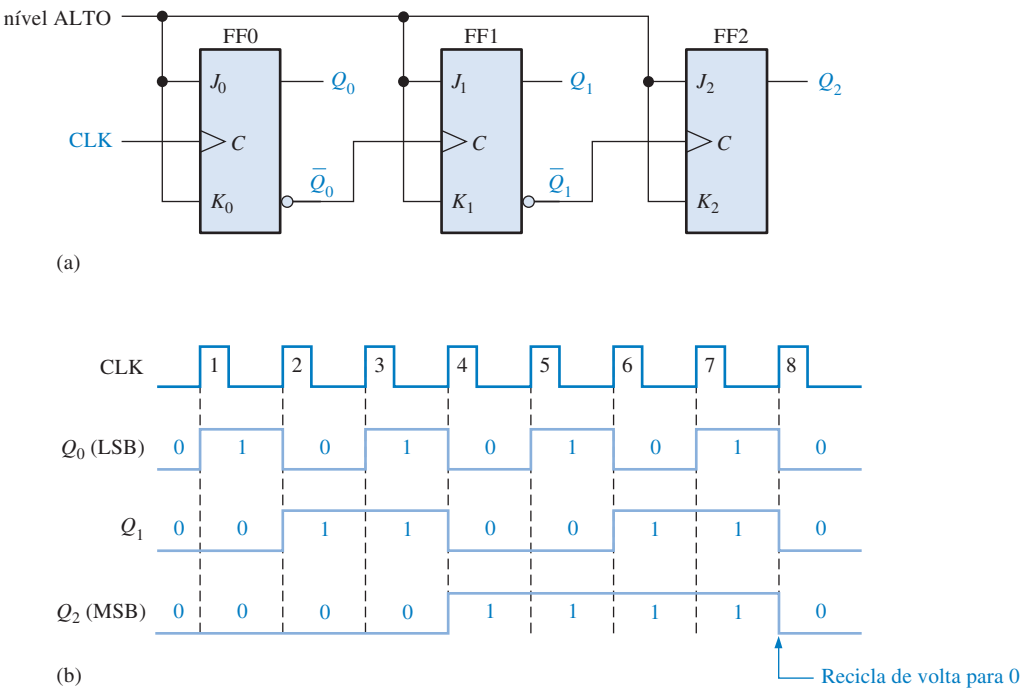
Como a seqüência é binária, o contador visto na Figura 8–1 é um contador binário. Ele conta realmente o número de pulsos de clock até três e no quarto pulso recicla para o seu estado original ($Q_0 = 0$, $Q_1 = 0$). O termo **reciclagem** é normalmente aplicado à operação de contadores; esse termo se refere à transição do contador do seu estado final de volta para o seu estado original.

Um Contador Binário Assíncrono de 3 Bits A seqüência de estados para um contador binário de 3 bits é mostrada na Tabela 8–2 e um contador binário de 3 bits é mostrado na Figura 8–3(a). A operação básica é a mesma que a do contador de 2 bits exceto que o contador de 3 bits tem oito es-

TABELA 8–2

Seqüência de estados para um contador binário de 3 bits

PULSO DE CLOCK	Q_2	Q_1	Q_0
Valor inicial	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (reciclagem)	0	0	0



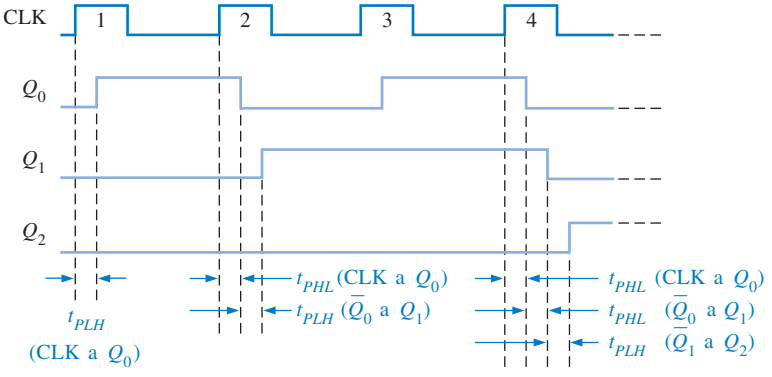
► FIGURA 8–3

Contador binário de 3 bits e seu diagrama de temporização para um ciclo. Abra o arquivo F08-03 para verificar a operação.

tados, pois tem três flip-flops. Um diagrama de temporização é mostrado na Figura 8–3(b) para oito pulsos de clock. Observe que o contador avança pela contagem binária de zero até sete e então recicla para o estado zero. Esse contador pode ser facilmente expandido para contagens maiores, acrescentando flip-flops *T*.

Atraso de Propagação Os contadores assíncronos são normalmente referidos como **contadores ondulantes** (*ripple counters*) pela seguinte razão: o efeito do pulso de clock na entrada é "sentido" primeiro pelo FF0. Esse efeito não chega em FF1 imediatamente devido ao atraso de propagação através de FF0. Então existe um atraso de propagação através de FF1 antes que FF2 possa ser disparado. Portanto, o efeito do pulso de clock na entrada "ondula" através do contador, durante algum tempo, devido aos atrasos de propagação, até alcançar o último flip-flop.

Para ilustrar, observe que todos os três flip-flops no contador visto na Figura 8–3 mudam de estado na borda de subida de CLK4. Esse efeito de clock ondulante é mostrado na Figura 8–4 para os primeiros quatro pulsos de clock, com os atrasos de propagação indicados. A transição do nível BAIXO para o nível ALTO de Q_0 ocorre um tempo de atraso (t_{PLH}) após a transição positiva do



► FIGURA 8–4

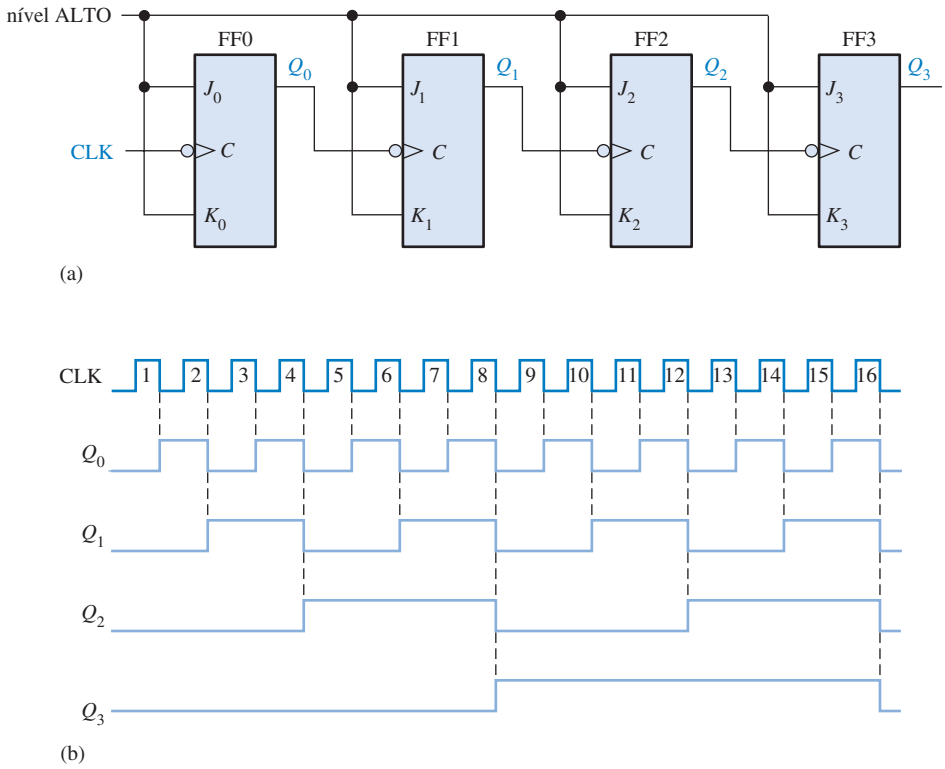
Atrasos de propagação num contador binário assíncrono (clock ondulante) de 3 bits.

pulso de clock. A transição de nível BAIXO para nível ALTO de Q_1 ocorre um tempo de atraso (t_{PLH}) após a transição positiva de $\overline{Q}_0$. A transição de nível BAIXO para nível ALTO de Q_2 ocorre um tempo de atraso (t_{PLH}) após a transição positiva de $\overline{Q}_1$. Como podemos ver, FF2 não é disparado antes de dois tempos de atraso após a borda positiva do pulso de clock CLK4. Portanto, decorre três tempos de atraso de propagação para o efeito do pulso de clock, CLK4, para ondular através do contador e mudar Q_2 de nível BAIXO para nível ALTO.

Esse atraso cumulativo de um contador assíncrono é a principal desvantagem em muitas aplicações porque limita a taxa na qual o contador pode receber pulsos de clock e cria problemas de decodificação. O atraso cumulativo máximo num contador tem que ser menor que o período da forma de onda do clock.

EXEMPLO 8–1

Um contador binário assíncrono de 4 bits é mostrado na Figura 8–5(a). Cada flip-flop é disparado por borda negativa e tem um atraso de propagação de 10 nanossegundos (ns). Desenvolva um diagrama de temporização mostrando a saída Q de cada flip-flop e determine o tempo de atraso de propagação total a partir da borda de disparo de um pulso de clock até a correspondente mudança que pode ocorrer no estado de Q_3. Determine também a freqüência de clock máxima na qual o contador pode operar.



▲ **FIGURA 8–5**

Contador binário assíncrono de quatro bits e o seu diagrama de temporização. Abra o arquivo F08-05 e verifique a operação.

Solução O diagrama de temporização com os atrasos omitidos é mostrado na Figura 8–5(b). Para o tempo de atraso total, o efeito de CLK8 ou CLK16 tem que propagar através de quatro flip-flops antes da mudança de Q_3, assim

$$t_{p(tot)} = 4 \times 10 \text{ ns} = \mathbf{40 \text{ ns}}$$

A freqüência de clock máxima é

$$f_{\max} = \frac{1}{t_{p(tot)}} = \frac{1}{40 \text{ ns}} = \mathbf{25 \text{ MHz}}$$

Problema relacionado* Mostre o diagrama de temporização se todos os flip-flops na Figura 8–5(a) são disparados por borda positiva.

* As respostas estão no final do capítulo.

Contadores de Década Assíncronos

Um contador pode ter 2^n estados, onde n é o número de flip-flops.

O **módulo** de um contador é o número de estados únicos pelos quais o contador estabelece uma seqüência. O número máximo de estados possíveis (módulo máximo) de um contador é 2^n, onde n é o número de flip-flops do contador. Os contadores podem ser projetados para ter um número de estados em sua seqüência que é menor que o valor máximo de 2^n. Esse tipo de seqüência é denominada de *seqüência truncada*.

Um módulo comum para contadores com seqüências truncadas é dez (denominado MOD10). Os contadores com dez estados em sua seqüência são denominados contadores de **década**. Um contador de década com uma seqüência de contagem de zero (0000) a nove (1001) é um contador de década BCD porque a sua seqüência de dez estados produz o código BCD. Esse tipo de contador é útil em aplicações com display nas quais o código BCD é necessário para conversões com leituras em decimal.

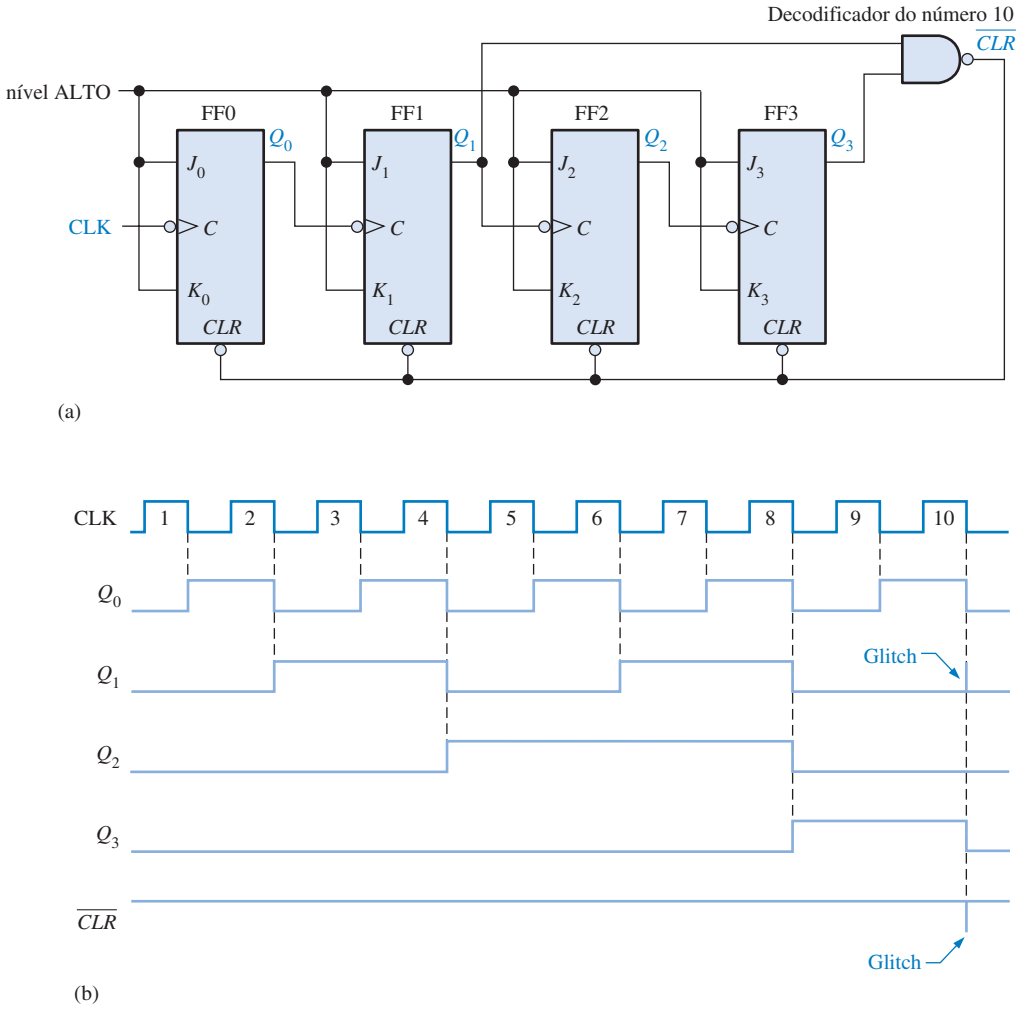
Para obter uma seqüência truncada, é necessário forçar o contador a reciclar antes que ele passe por todos os estados possíveis. Por exemplo, um contador de década BCD tem que reciclar para o estado 0000 após o estado 1001. Um contador de década requer quatro flip-flops (três flip-flops são insuficientes pois $2^3 = 8$).

Vamos usar um contador assíncrono de 4 bits como no Exemplo 8–1 e modificar sua seqüência para ilustrar o princípio dos contadores truncados. Uma forma de fazer um contador reciclar após a contagem do nove (1001) é decodificar a contagem dez (1010) com uma porta NAND e conectar a saída da porta NAND nas entradas de clear ($\overline{CLR}$) dos flip-flops, como mostra a Figura 8–6(a).

Decodificação Parcial Observe na Figura 8–6(a) que apenas Q_1 e Q_3 estão conectadas nas entradas da porta NAND. Esse arranjo é um exemplo de *decodificação parcial*, na qual os dois únicos estados ($Q_1 = 1$ e $Q_3 = 1$) são suficientes para decodificar a contagem do número dez porque nenhum dos outros estados (zero a nove) tem Q_1 e Q_3 em nível ALTO ao mesmo tempo. Quando o contador entra na contagem dez (1010), a saída da porta de decodificação vai para nível BAIXO e assincronamente reseta todos os flip-flops.

O diagrama de temporização resultante é mostrado na Figura 8–6(b). Observe que existe um glitch na forma de onda Q_1. A razão para esse glitch é que Q_1 tem que ir primeiro para nível ALTO antes que a contagem dez seja decodificada. A saída da porta de decodificação vai para nível BAIXO (as duas entradas são nível ALTO) vários nanossegundos depois de o contador atingir a contagem dez. Portanto, o contador fica no estado 1010 por um curto intervalo de tempo antes de resetar para 0000, produzindo assim o glitch em Q_1 e o glitch resultante na linha $\overline{CLR}$ que reseta o contador.

Outras seqüências truncadas podem ser implementadas de forma similar, como mostra o Exemplo 8–2.



◀ **FIGURA 8–6**

Um contador de década que recebe pulsos de clock assincronamente e que recicla também de forma assíncrona.

EXEMPLO 8–2

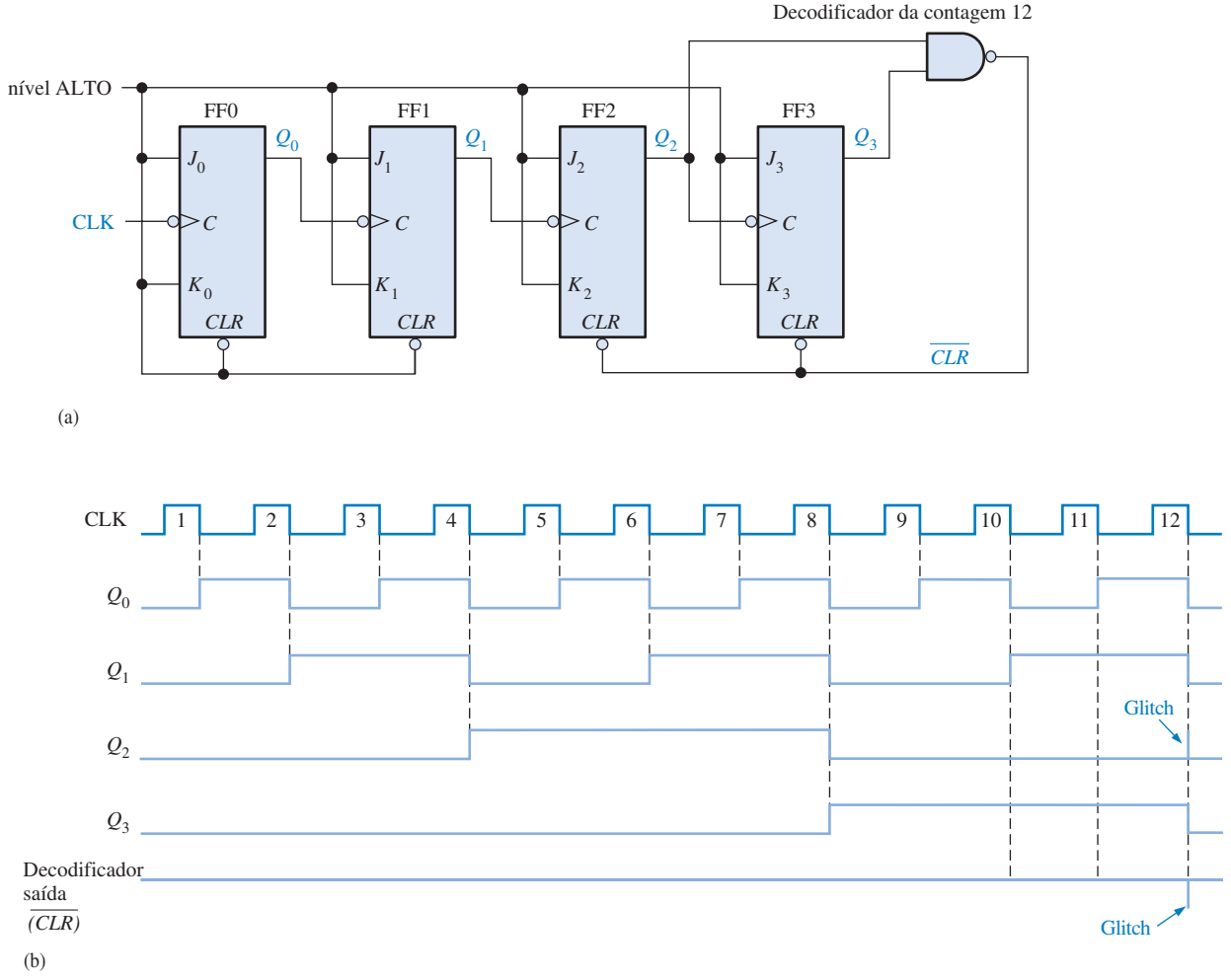
Mostre como um contador assíncrono pode ser implementado tendo um módulo de doze com uma seqüência binária de 0000 a 1011.

Solução Como três flip-flops podem produzir um máximo de oito estados, quatro flip-flops são necessários para produzir qualquer módulo maior que oito mas menor ou igual a dezesseis.

Quando o contador atinge seu ltimo estado, 1011, ele reinicializa (volta para 0000) em vez de passar para o seu próximo estado normal (1100), conforme ilustrado na seqüência a seguir:

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	←
.	.	.	.	Reciclagem
.	.	.	.	
.	.	.	.	
1	0	1	1	←
1	1	0	0	← Próximo estado normal

Observe que Q_0 e Q_1 vão para 0 de qualquer forma, porém Q_2 e Q_3 têm que ser forçadas para 0 no décimo segundo pulso de clock. A Figura 8–7(a) mostra o contador de módulo 12. A porta NAND decodifica parcialmente a contagem doze (1100) e reseta o flip-flop 2 e o flip-flop 3. Portanto, no décimo segundo pulso de clock, o contador é forçado a reciclar da contagem onze para a contagem zero, conforme mostra o diagrama de temporização dado na Figura 8–7(b). (o contador permanece na contagem doze por apenas alguns nanossegundos antes que seja resetado pelo glitch em $\overline{CLR}$).

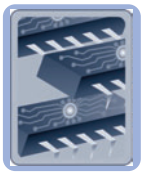


▲ **FIGURA 8–7**

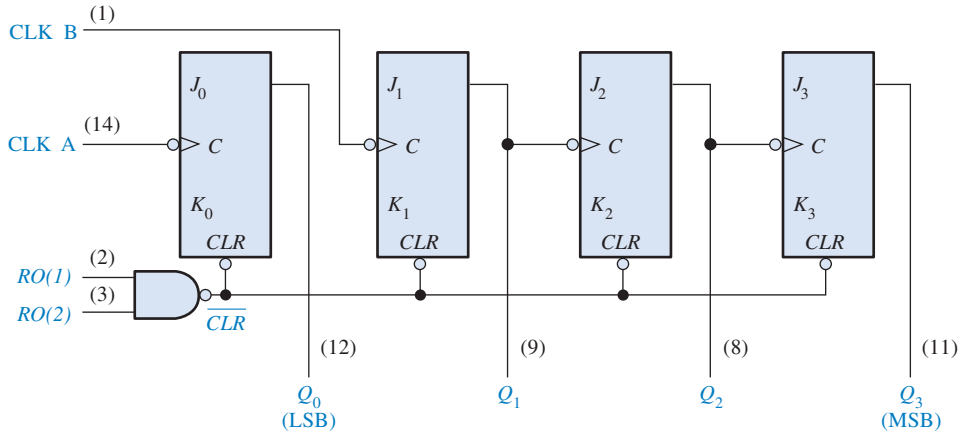
Contador de módulo 12 assíncrono com reciclagem assíncrona.

Problema relacionado Determine como pode ser a modificação no circuito contador visto na Figura 8–7(a) para torná-lo um contador de módulo 13?

CONTADOR BINÁRIO ASSÍNCRONO DE 4 BITS (74LS93)



O CI 74LS93 é um exemplo de um contador assíncrono na forma de circuito integrado específico. Assim como mostra o diagrama lógico na Figura 8–8, esse dispositivo consiste na realidade de um flip-flop e um contador assíncrono de 3 bits. Esse arranjo tem a finalidade de flexibilidade. Ele pode ser usado como um dispositivo divisor por 2 sendo usado apenas o flip-flop ou pode ser usado como um contador de módulo 8 se for usado apenas a parte do contador de 3 bits. Esse dispositi-

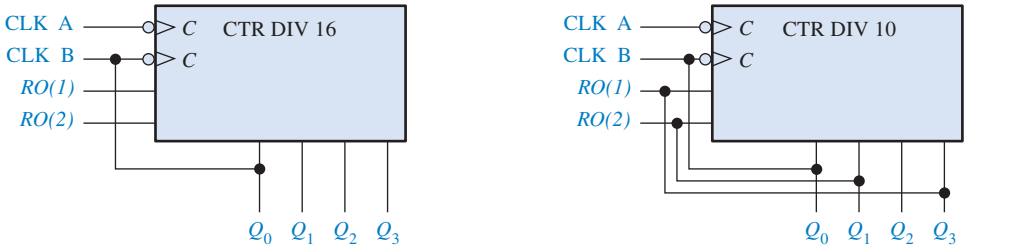


▲ **FIGURA 8–8**

Diagrama lógico do CI contador binário assíncrono de 4 bits 74LS93. (Os números dos pinos estão entre parênteses e todas as entradas *J* e *K* estão conectadas internamente no nível ALTO).

vo também provê entradas de controle de resete, *R0(1)* e *R0(2)*. Quando essas duas entradas estiverem em nível ALTO, o contador é resetado para 0000 pela linha $\overline{CLR}$.

Adicionalmente, o CI 74LS93 pode ser usado como um contador de 4 bits que conta de 0 a 15 (módulo 16) conectado a saída Q_0 na entrada CLK *B* como mostra a Figura 8–9(a). Ele também pode ser configurado como um contador de década (conta de 0 a 9) com reciclagem assíncrona usando as entradas de controle de resete para decodificação parcial da contagem dez, como mostra a Figura 8–9(b).



(a) 74LS93 conectado como um contador de módulo 16

(b) 74LS93 conectado como um contador de década

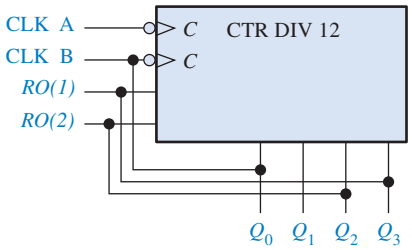
▲ **FIGURA 8–9**

Duas configurações de contador assíncrono com o 74LS93. (A identificação qualificativa, CTR DIV *n*, indica um contador com *n* estados).

EXEMPLO 8–3

Mostre de que forma o CI 74LS93 pode ser usado como um contador de módulo 12.

Solução Use as entradas de controle de resete, *R0(1)* e *R0(2)*, para decodificação parcial para a contagem 12 (lembre-se, existe uma porta NAND interna associada com essas entradas). A decodificação da contagem 12 é realizada pela conexão de Q_3 a *R0(1)* e Q_2 a *R0(2)*, como mostra a Figura 8–10. A saída Q_0 é conectada em CLK B para se obter um contador de 4 bits.



▶ **FIGURA 8–10**
O CI 74LS93 conectado como um contador de módulo 12.

Logo que o contador alcança a contagem 12 (1100), ele é resetado para 0000. Entretanto, a reciclagem resulta num glitch no estado 1100 por vários nanossegundos antes da reciclagem.

Problema relacionado Mostre como o CI 74LS93 pode ser conectado como um contador de módulo 13.

SEÇÃO 8–1 REVISÃO
As respostas estão no final do capítulo.

1. O que significa o termo *assíncrono* em se tratando de contadores?
2. Quantos estados tem um contador de módulo 14? Qual é o número mínimo necessário de flip-flops?

8-2 OPERAÇÃO DE CONTADORES SÍNCRONOS

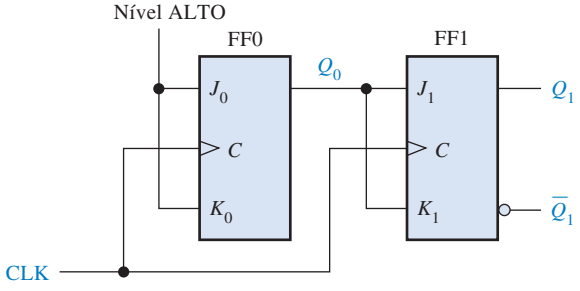
O termo **síncrono** se refere aos eventos que têm uma relação de tempo fixa de um em relação ao outro. Um **contador síncrono** é aquele no qual todos os flip-flops recebem pulsos de clock ao mesmo tempo por meio de uma linha comum.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a operação de um contador binário síncrono de 2 bits ■ Descrever a operação de um contador binário síncrono de 3 bits ■ Descrever a operação de um contador binário síncrono de 4 bits ■ Descrever a operação de um contador de década síncrono ■ Desenvolver diagramas de temporização de contadores ■ Discutir o CI contador binário de 4 bits 74HC163 e o CI contador de década BCD 74F162

Um Contador Binário Síncrono de 2 Bits

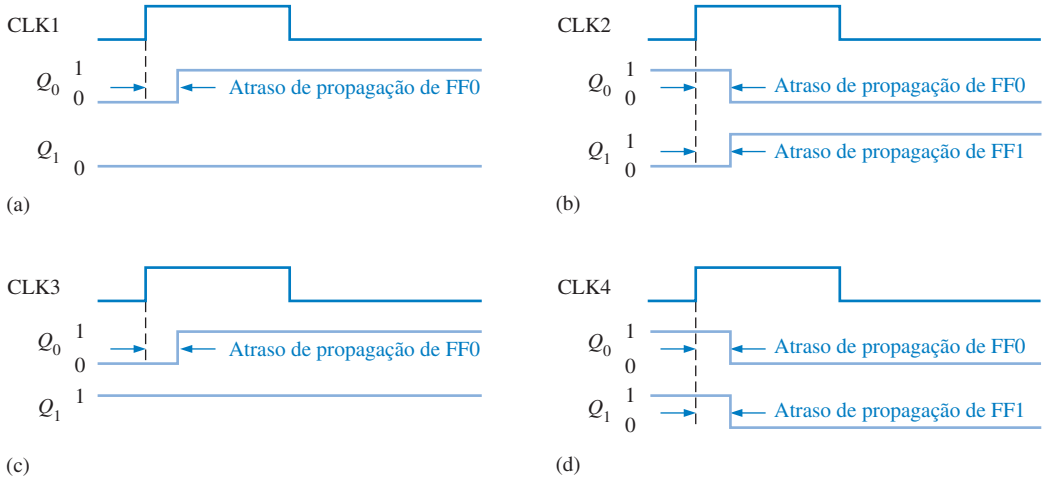
A Figura 8–11 mostra um contador binário síncrono de 2 bits. Observe que um arranjo diferente do contador assíncrono tem que ser usado para as entradas J_1 e K_1 do FF1 para conseguir uma seqüência binária.



▶ **FIGURA 8–11**
Um contador binário síncrono de 2 bits.

A entrada de clock é conectada em cada flip-flop num contador síncrono.

A operação desse contador síncrono é a seguinte: primeiro, considere que o contador esteja inicialmente no estado binário 0; ou seja, os dois flip-flops estão resetados. Quando a borda positiva do primeiro pulso de clock é aplicada, o FF0 comuta e Q_0 passa então para o nível ALTO. O que acontece com FF1 na borda positiva de CLK1? Para saber, vamos analisar as condições de entrada de FF1. A entradas J_1 e K_1 estão em nível BAIXO porque Q_0, saída na qual elas estão conectadas, ainda não foi para nível ALTO. Lembre-se, existe um atraso de propagação a partir da borda de disparo do pulso de clock até que a saída Q comute realmente. Assim, $J = 0$ e $K = 0$ quando é aplicada a borda de subida do primeiro pulso de clock. Essa é uma condição de repouso e assim FF1 não muda de estado. Uma representação detalhada da temporização dessa parte da operação do contador é mostrada na Figura 8–12(a).



▲ **FIGURA 8–12**

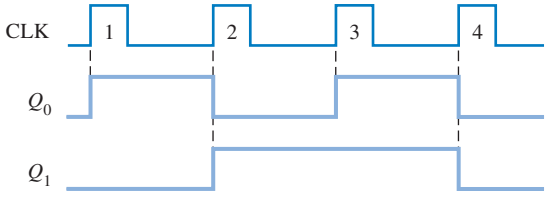
Detalhes da temporização para a operação do contador síncrono de 2 bits (os atrasos de propagação dos dois flip-flops são considerados iguais).

Após CLK1, $Q_0 = 1$ e $Q_1 = 0$ (que é o estado do binário 1). Quando a borda de subida de CLK2 ocorre, FF0 comuta (toggle) e Q_0 vai para nível BAIXO. Como FF1 tem um nível ALTO ($Q_0 = 1$) nas entradas J_1 e K_1 na borda de disparo desse pulso de clock, o flip-flop comuta e Q_1 vai para nível ALTO. Portanto, após CLK2, $Q_0 = 0$ e $Q_1 = 1$ (que é o estado do binário 2). O detalhe da temporização para essa condição é mostrado na Figura 8–12(b).

Quando a borda de subida de CLK3 ocorre, FF0 comuta novamente para o estado SET ($Q_0 = 1$) e FF1 permanece em SET ($Q_1 = 1$) porque suas entradas J_1 e K_1 estão em nível BAIXO ($Q_0 = 0$). Após essa borda de disparo, $Q_0 = 1$ e $Q_1 = 1$ (que é o estado do binário 3). O detalhe da temporização é mostrado na Figura 8–12(c).

Finalmente, na borda de subida de CLK4, Q_0 e Q_1 vão para nível BAIXO porque os dois estão na condição toggle em suas entradas J e K. O detalhe da temporização é mostrado na Figura 8–12(d). Agora o contador reciclou para o seu estado original, binário 0.

O diagrama de temporização completo para o contador visto na Figura 8–11 é mostrado na Figura 8–13. Observe que todas as transições da forma de onda aparecem coincidentes; ou seja, os atrasos de propagação não são indicados. Embora os atrasos sejam um fator importante na operação de um contador síncrono, eles normalmente são omitidos no diagrama de temporização geral



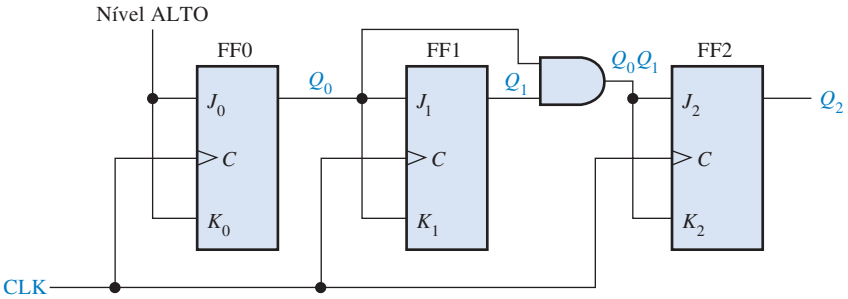
◀ **FIGURA 8–13**

Diagrama de temporização para o contador apresentado na Figura 8–11.

por questão de simplicidade de representação. As relações de formas de onda principais resultam da operação normal de um circuito que pode ser expressa completamente sem mostrar pequenos atrasos e diferenças de temporização. Entretanto, em circuitos digitais de alta velocidade, esses pequenos atrasos são considerações importantes no projeto e na análise de defeito.

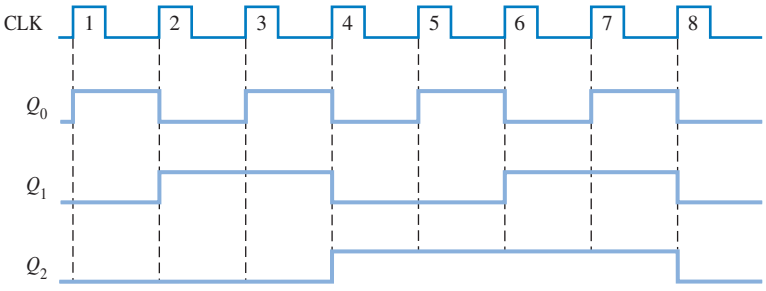
Um Contador Binário Síncrono de 3 Bits

Um contador binário síncrono de 3 bits é mostrado na Figura 8–14 e o seu diagrama de temporização é mostrado na Figura 8–15. Podemos entender essa operação do contador examinando sua seqüência de estados como mostra a Tabela 8–3.



▶ **FIGURA 8–14**

Um contador binário síncrono de 3 bits. Abra o arquivo F08-14 para verificar a operação.



▶ **FIGURA 8–15**

Diagrama de temporização para o contador mostrado na Figura 8–14.

NOTA: COMPUTAÇÃO

O TSC ou *time stamp counter* (contador de tempo de referência) no Pentium é usado para monitorar o desempenho, o que permite que diversos parâmetros importantes do desempenho de um sistema Pentium sejam determinados com exatidão. Lendo o TSC antes e depois da execução de um procedimento, o tempo preciso necessário para o procedimento pode ser determinado com base no tempo de ciclo do processador. Dessa forma, o TSC representa a base para toda a avaliação de tempo em conexão com a operação de sistema otimizado. Por exemplo, pode-se determinar com precisão qual dentre duas ou mais seqüências de programação é mais eficiente. Essa é uma ferramenta útil para desenvolvedores de compiladores e programadores de sistemas no desenvolvimento de um código mais efetivo para o Pentium.

▶ **TABELA 8–3**

Seqüência de estados binários para um contador binário de 3 bits

CLOCK DE PULSO	Q_2	Q_1	Q_0
Estado inicial	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (reciclagem)	0	0	0

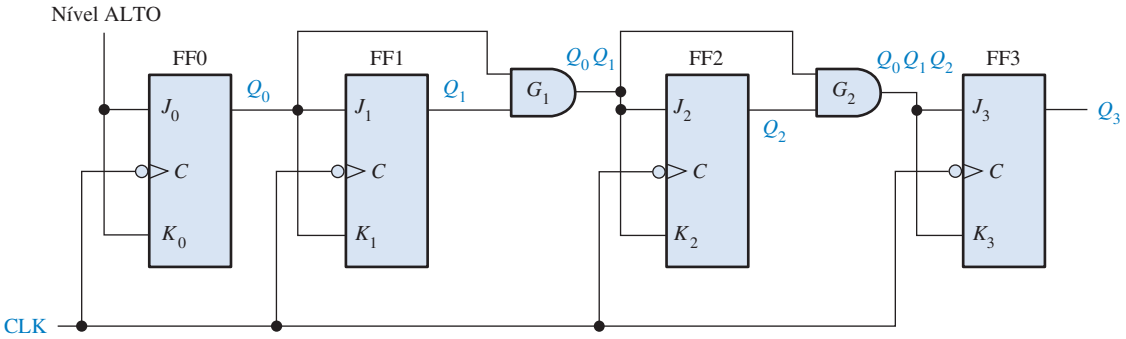
Primeiro vamos analisar Q_0. Observe que Q_0 muda a cada pulso de clock conforme o contador avança do seu estado original para o seu estado final e em seguida retorna para o seu estado original. Para produzir essa operação, FF0 tem que ser mantido no modo toggle (comutar) mantendo constantemente o nível ALTO nas suas entradas J_0 e K_0. Observe que Q_1 vai para o estado oposto cada vez que Q_0 for nível 1. Essa mudança ocorre em CLK2, CLK4, CLK6 e CLK8. O pulso CLK8 faz com que o contador recicle. Para produzir essa operação, Q_0 é conectada nas entradas J_1 e K_1 de FF1. Quando Q_0 for nível 1 e ocorrer um pulso de clock, FF1 estará no modo toggle

e assim mudará de estado. As outras vezes, quando Q_0 estiver em 0, FF1 estará no modo de repouso e permanecerá no seu estado atual.

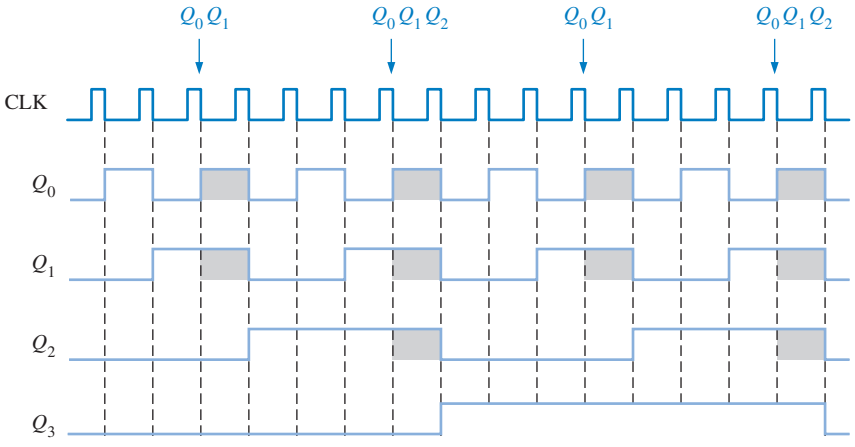
Em seguida, vamos analisar como é feito para FF2 mudar nos momentos apropriados de acordo com a seqüência binária. Observe que as duas vezes em que Q_2 muda de estado, é precedido por uma condição única na qual Q_0 e Q_1 são nível ALTO. Essa condição é detectada pela porta AND e aplicada nas entradas J_2 e K_2 de FF2. Sempre que Q_0 e Q_1 forem nível ALTO, a saída da porta AND faz com que as entradas J_2 e K_2 de FF2 sejam nível ALTO, sendo que FF2 comuta no pulso de clock seguinte. Em todos os outros instantes, as entradas J_2 e K_2 de FF2 são mantidas em nível BAIXO pela saída da porta AND, sendo que FF2 não muda de estado.

Contador Binário Síncrono de 4 Bits

A Figura 8–16(a) mostra um contador binário síncrono de 4 bits e a Figura 8–16(b) mostra o diagrama de temporização. Esse contador em particular é implementado com flip-flops disparados por borda negativa. O raciocínio por trás do controle das entradas *J* e *K* para os primeiros três flip-flops é o mesmo discutido anteriormente para o contador de 3 bits. O quarto estágio (FF3) muda apenas duas vezes nessa seqüência. Observe que essas duas transições ocorrem apenas no momento seguinte em que Q_0, Q_1 e Q_2 são todas nível ALTO. Essa condição é decodificada pela porta AND G_2 de forma que quando um pulso de clock ocorrer, FF3 mudará de estado. Para todos os outros momentos, as entradas J_3 e K_3 de FF3 estarão em nível BAIXO e na condição de repouso.



(a)



(b)

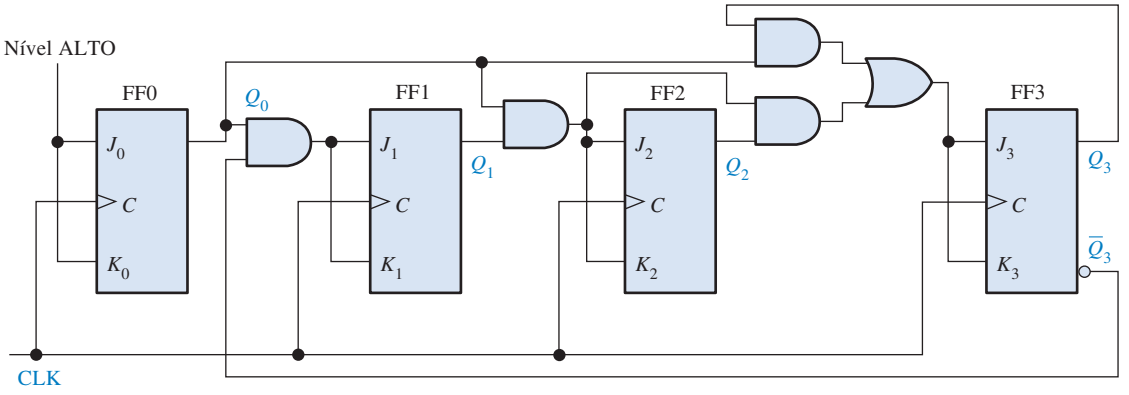
▲ **FIGURA 8–16**

Um contador binário síncrono de 4 bits e o seu diagrama de temporização. Os pontos em que as entradas das portas AND são nível ALTO estão indicados por áreas sombreadas.

Contador de Década Sícrono de 4 Bits

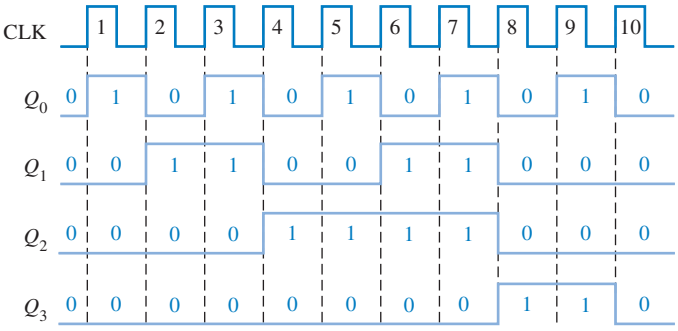
Um contador de década tem dez estados.

Como sabemos, um contador de década BCD exibe uma seqüência binária truncada passando pelos estados de 0000 a 1001. Em vez de passar do estado 1001 para o estado 1010, ele recicla para o estado 0000. Um contador de década BCD síncrono é mostrado na Figura 8–17. O diagrama de temporização do contador de década é mostrado na Figura 8–18.



▲ **FIGURA 8–17**

Um contador de década BCD síncrono. Abra o arquivo F08-17 para verificar a operação.



▶ **FIGURA 8–18**

Diagrama de temporização para o contador de década BCD (Q_0 é o LSB).

Podemos entender a operação do contador examinando a seqüência de estados na Tabela 8–4 e seguindo a implementação na Figura 8–17. Primeiro, observe que FF0 (Q_0) comuta a cada pulso de clock, assim a equação lógica para as entradas J_0 e K_0 é

$$J_0 = K_0 = 1$$

Essa equação é implementada conectando J_0 e K_0 constantemente ao nível ALTO.

Em seguida, observe na Tabela 8–4 que FF1 (Q_1) muda de estado no próximo pulso de clock cada vez que $Q_0 = 1$ e $Q_3 = 0$, assim a equação lógica para as entradas J_1 e K_1 é

$$J_1 = K_1 = Q_0\overline{Q}_3$$

Essa equação é implementada fazendo uma AND entre Q_0 e $\overline{Q}_3$ e conectando a saída da porta nas entradas J_1 e K_1 de FF1.

O FF2 (Q_2) muda de estado no próximo pulso de clock cada vez que $Q_0 = 1$ e $Q_1 = 1$. Isso requer como equação lógica a seguinte equação:

$$J_2 = K_2 = Q_0Q_1$$

Essa equação é implementada fazendo uma AND entre Q_0 e Q_1 e conectando a saída da porta nas entradas J_2 e K_2 de FF2.

PULSO DE CLOCK	Q_3	Q_2	Q_1	Q_0
Estado inicial	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recicla)	0	0	0	0

◀ **TABELA 8–4**

Estados de um contador de década BCD

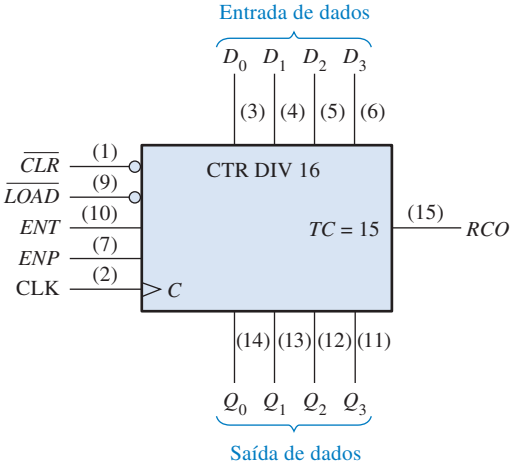
Finalmente, FF3 (Q3) muda para o estado oposto no próximo pulso de clock cada vez que $Q_0 = 1$, $Q_1 = 1$ e $Q_2 = 1$ (estado 7), ou quando $Q_0 = 1$ e $Q_3 = 1$ (estado 9). A equação para essa situação é:

$$J_3 = K_3 = Q_0Q_1Q_2 + Q_0Q_3$$

Essa função é implementada com lógica AND/OR conectada às entradas J_3 e K_3 de FF3 como mostra o diagrama lógico visto na Figura 8–17. Observe que as diferenças entre esse contador de década e o contador binário de módulo 16 são a porta AND $Q_0\overline{Q}_3$, e a porta OR; esse arranjo detecta a ocorrência do estado 1001 e faz com que o contador recicle corretamente no próximo pulso de clock.

O CONTADOR BINÁRIO SÍNCRONO DE 4 BITS (74HC163)

O CI 74HC163 é um exemplo de um contador binário síncrono de 4 bits. Um símbolo lógico é mostrado na Figura 8–19 com os números dos pinos entre parênteses. Esse contador tem vários aspectos além das funções básicas anteriormente discutidas para os contadores binários síncronos.



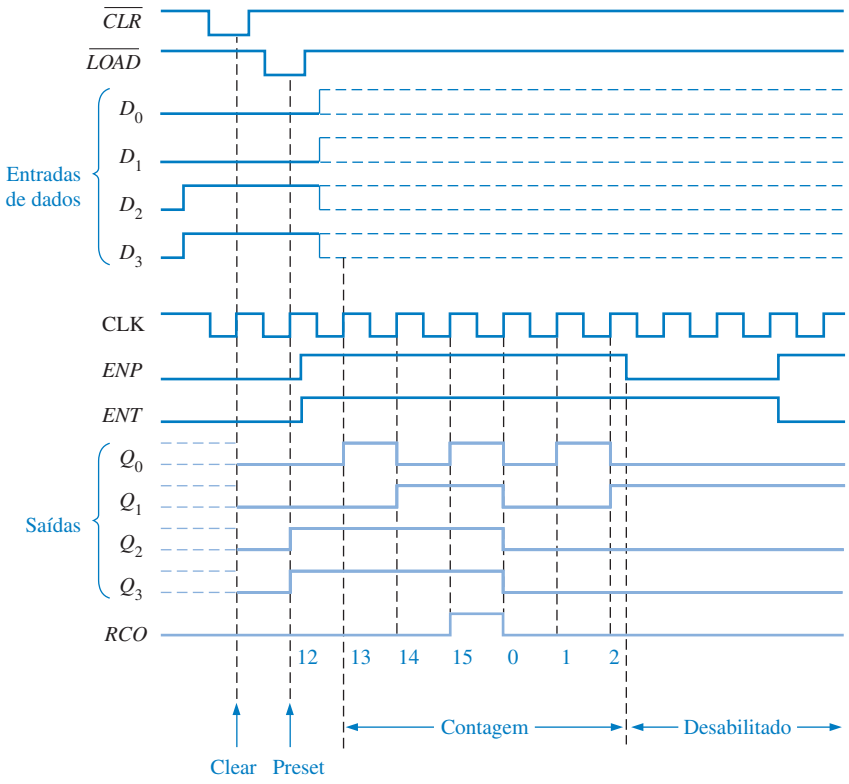
◀ **FIGURA 8–19**

O CI contador binário síncrono de 4 bits 74HC163. (A indicação qualificativa CTR DIV 16 indica um contador com dezesseis estados.)

Primeiro, o contador pode ser presetado sincronamente para qualquer número binário de 4 bits aplicando os níveis apropriados nas entradas de dados em paralelo. Quando um nível BAIXO é aplicado na entrada $\overline{LOAD}$, o contador assume os estados das entradas de dados no próximo pulso de clock. Portanto, a seqüência de contagem pode ser iniciada com qualquer número binário de 4 bits.

Além disso, existe uma entrada clear ($\overline{CLR}$) ativa em nível BAIXO, que reseta sincronamente todos os flip-flops do contador. Existe duas entradas de habilitação, *ENP* e *ENT*. Essas entradas têm que estar em nível ALTO para que o contador percorra os estados binários da seqüência. Quando pelo menos uma dessas entradas for nível BAIXO, o contador está desabilitado. A saída de clock ondulante (*RCO*) vai para nível ALTO quando alcança o último estado em sua seqüência de quinze, denominada de **contagem final** ($TC = 15$). Essa saída, juntamente com as entradas de habilitação, permite que contadores desse tipo sejam conectados em cascata para se obter seqüências de contagem maiores.

A Figura 8–20 mostra um diagrama de temporização desse contador sendo presetado para doze (1100) e então contando de forma crescente até a sua contagem final, quinze (1111). A entrada D_0 é o bit de entrada menos significativo e Q_0 é o bit de saída menos significativo.



▲ **FIGURA 8–20**

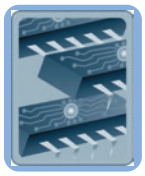
Exemplo de temporização para o CI 74HC163.

Vamos examinar esse diagrama de temporização em detalhes. Isso nos ajudará a interpretar diagramas de temporização nesse capítulo ou em folhas de dados de fabricantes. Para começar, o pulso de nível BAIXO na entrada $\overline{CLR}$ faz com que todas as saída (Q_0, Q_1, Q_2 e Q_3) sejam nível BAIXO.

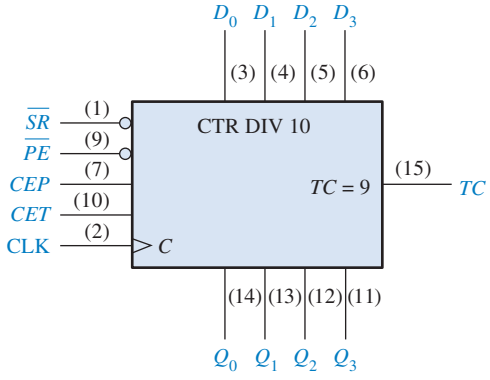
Em seguida, o pulso de nível BAIXO na entrada $\overline{LOAD}$ insere os dados que estão nas entradas de dados (D_0, D_1, D_2 e D_3) para dentro do contador. Esses dados aparecem nas saídas Q no instante da primeira borda de clock positiva após $\overline{LOAD}$ ter ido para nível BAIXO. Essa é a operação de *presete*. Nesse exemplo em particular, Q_0 é nível BAIXO, Q_1 é nível BAIXO, Q_2 é nível BAIXO e Q_3 é nível ALTO. Essa saída corresponde, é claro, ao binário 12 (Q_0 é o LSB).

O contador agora avança percorrendo os estados 13, 14 e 15 nas próximas três bordas positivas do clock. Em seguida o contador recicla para 0, 1, 2 nos pulsos de clock seguintes. Observe que as entradas *ENP* e *ENT* estão em nível ALTO durante a seqüência de estados. Quando *ENP* vai para nível BAIXO, o contador é desabilitado e permanece no estado do binário 2.

O CONTADOR DE DÉCADA BCD SÍNCRONO (74F162)

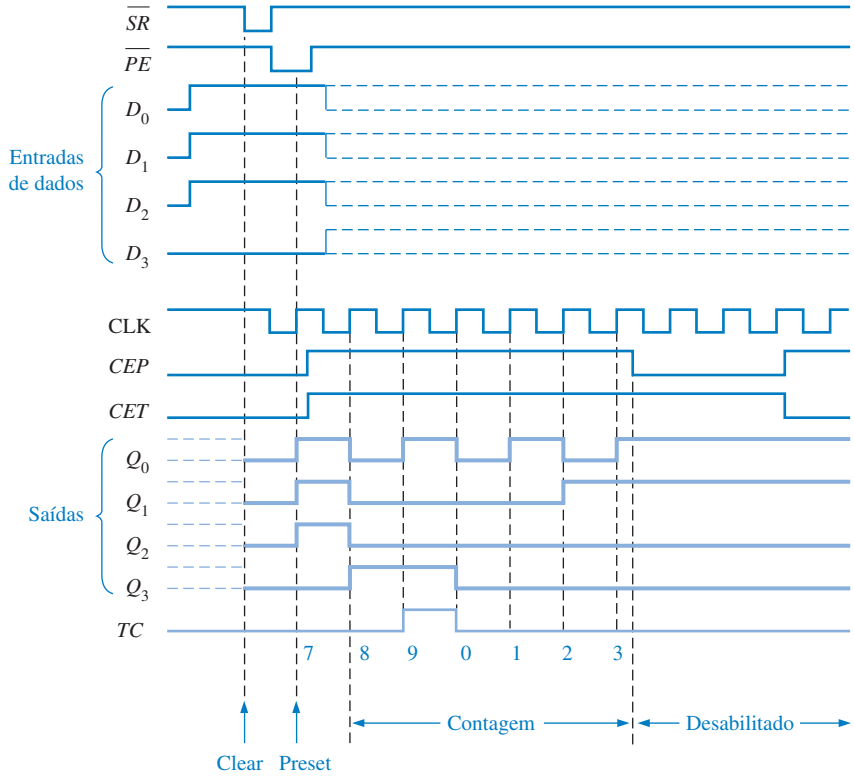


O CI 74F162 é um exemplo de um contador de década. Ele pode ser presetado para qualquer contagem BCD usando as entradas de dados e um nível BAIXO na entrada $\overline{PE}$. Um nível BAIXO na entrada $\overline{SR}$ reseta o contador. As entradas de habilitação *CEP* e *CET* têm que estar em nível ALTO para o contador poder avançar através da seqüência de estados em resposta a transições positivas na entrada CLK. As entradas de habilitação em conjunto com a saída de contagem final, *TC* (1001), provê possibilidades de conexão em cascata de diversos contadores de década. A Figura 8–21 mostra o símbolo lógico para o CI contador 74F162 e a Figura 8–22 é um diagrama de temporização mostrando o contador sendo presetado para a contagem 7 (0111). O contador em cascata será discutido na Seção 8–5.



◄ FIGURA 8–21

O CI contador de década BCD síncrono 74F162. (A identificação de qualificação CTR DIV 10 indica um contador com dez



▲ FIGURA 8–22

Exemplo de temporização para um CI 74F162.

SEÇÃO 8–2 REVISÃO

1. Em que um contador síncrono difere de um contador assíncrono?
2. Explique a função da característica de *presete* de contadores tal como o CI 74HC163.
3. Descreva a finalidade das entradas *ENP* e *ENT* e da saída *RCO* para o CI contador 74HC163.

8-3 CONTADORES SÍNCRONOS CRESCENTE/DECRESCENTE

Um **contador crescente/decrescente** (*up/down*) é aquele que é capaz de avançar nas duas direções uma determinada seqüência. Um contador crescente/decrescente, algumas vezes denominado de contador bidirecional, pode ter qualquer seqüência especificada de estados. Um contador binário de 3 bits que avança de forma crescente através de sua seqüência (0, 1, 2, 3, 4, 5, 6, 7) podendo então ser invertida de forma que ele segue a seqüência na direção oposta (7, 6, 5, 4, 3, 2, 1, 0), é uma ilustração de uma operação crescente/decrescente

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a operação básica de um contador crescente/decrescente ■ Discutir o CI contador de década crescente/decrescente 74HC190

Em geral, a maioria dos contadores crescente/decrescente (up/down) pode ter o sentido da contagem invertida em qualquer ponto de sua seqüência. Por exemplo, o contador binário de 3 bits pode ser controlado para percorrer a seguinte seqüência:

CRESCENTE CRESCENTE

0, 1, 2, 3, 4, 5, 4, 3, 2, 3, 4, 5, 6, 7, 6, 5, etc.

DECRESCENTE DECRESCENTE

A Tabela 8–5 mostra a seqüência crescente/decrescente completa para um contador binário de 3 bits. As setas indicam o movimento estado-por-estado do contador nos modos de operação CRESCENTE e DECRESCENTE. O exame de Q_0 para as seqüências crescente e decrescente mostra que FF0 comuta a cada pulso de clock. Portanto, as entradas J_0 e K_0 de FF0 são

$$J_0 = K_0 = 1$$

Para a seqüência crescente, Q_1 muda de estado no próximo pulso de clock quando $Q_0 = 1$. Para a seqüência decrescente, Q_1 muda de estado no próximo pulso de clock quando $Q_0 = 0$. Portanto, as entradas J_1 e K_1 de FF1 tem que ser igual a 1 sob as condições expressas pela seguinte equação:

$$J_1 = K_1 = (Q_0 \cdot \text{CRESCENTE} + \overline{Q}_0 \cdot (\text{DECRESCENTE})$$

▶ **TABELA 8–5**

Seqüência crescente/decrescente para um contador binário de 3 bits

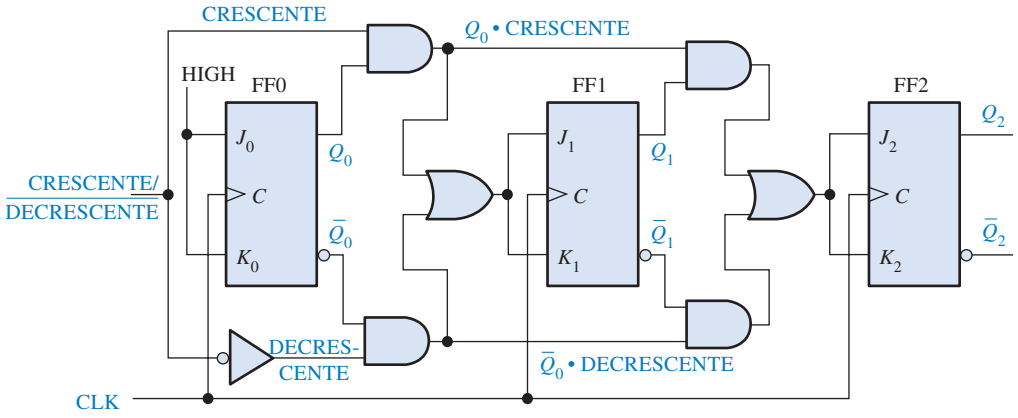
PULSO DE CLOCK	CRESC.	Q_2	Q_1	Q_0	DECR.
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

Para a seqüência crescente, Q_2 muda de estado no próximo pulso de clock quando $Q_0 = Q_1 = 1$. Para a seqüência decrescente, Q_2 muda de estado no próximo pulso de clock quando $Q_0 = Q_1 = 0$. Portanto, as entradas J_2 e K_2 de FF2 têm que ser igual a 1 sob as condições expressas pela seguinte equação:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \text{CRESCENTE}) + \overline{Q}_0 \cdot \overline{Q}_1 \cdot (\text{DECRESCENTE})$$

Cada uma das condições para as entradas *J* e *K* de cada flip-flop produz uma comutação no ponto apropriado da seqüência de contagem.

A Figura 8–23 mostra uma implementação básica de um contador binário crescente/decrescente usando as equações lógicos agora desenvolvidas para as entradas *J* e *K* de cada flip-flop. Observe que a entrada de controle *CRESCENTE*/*DECRESCENTE* é nível ALTO para contagem CRESCENTE e nível BAIXO para DECRESCENTE.

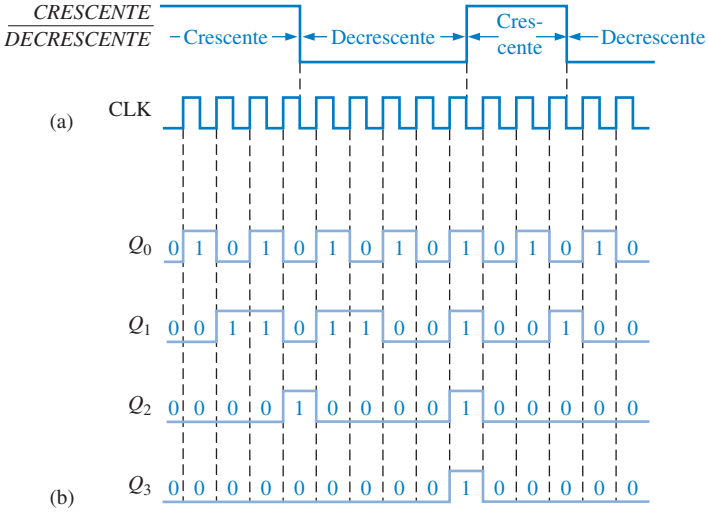


◀ **FIGURA 8–23**

Um contador crescente/decrescente básico de 3 bits. Abra o arquivo F08-23 para verificar a operação.

EXEMPLO 8–4

Mostre o diagrama de temporização e determine a seqüência de um contador binário crescente/decrescente síncrono de 4 bits se a entrada de clock e a entrada de controle *CRESCENTE*/*DECRESCENTE* têm formas de onda como as mostradas na Figura 8–24(a). O contador começa com todos os estados em 0 e é disparado pela borda positiva.



▲ **FIGURA 8–24**

Solução O diagrama de temporização mostrando as saídas *Q* é mostrado na Figura 8–24(b). A partir dessas formas de onda, a seqüência do contador é como mostra a Tabela 8–6.

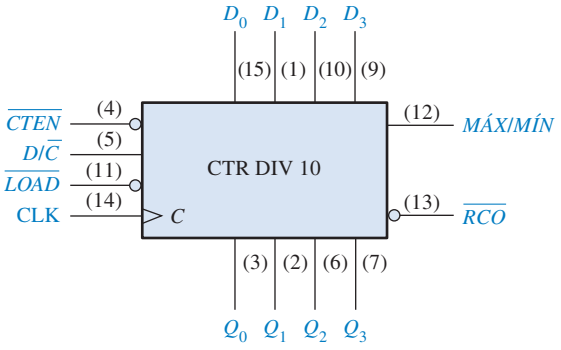
▶ **TABELA 8–6**

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	
0	0	0	1	
0	0	1	0	CRESC.
0	0	1	1	
0	1	0	0	
0	0	1	1	
0	0	1	0	
0	0	0	1	DECRES.
0	0	0	0	
1	1	1	1	
0	0	0	0	
0	0	0	1	CRESC.
0	0	1	0	
0	0	0	1	DECRES.
0	0	0	0	

Problema relacionado Mostre o diagrama de temporização se a forma de onda de controle na Figura 8–24(a) for invertida.

CONTADOR DE DÉCADA CRESCENTE/DECRESCENTE (74HC190)

A Figura 8–25 mostra um diagrama lógico para o CI 74HC190, um exemplo de um circuito integrado contador síncrono crescente/decrescente. A direção da contagem é determinada pelo nível na entrada crescente/decrescente $(D/\overline{C})$. Quando essa entrada for nível ALTO, o contador conta decrescente; quando ela for nível BAIXO, o contador conta crescente. Além disso, esse dispositivo pode ser presetado para qualquer dígito BCD desejado conforme determinado pelos estados das entradas de dados quando a entrada $\overline{LOAD}$ for nível BAIXO.

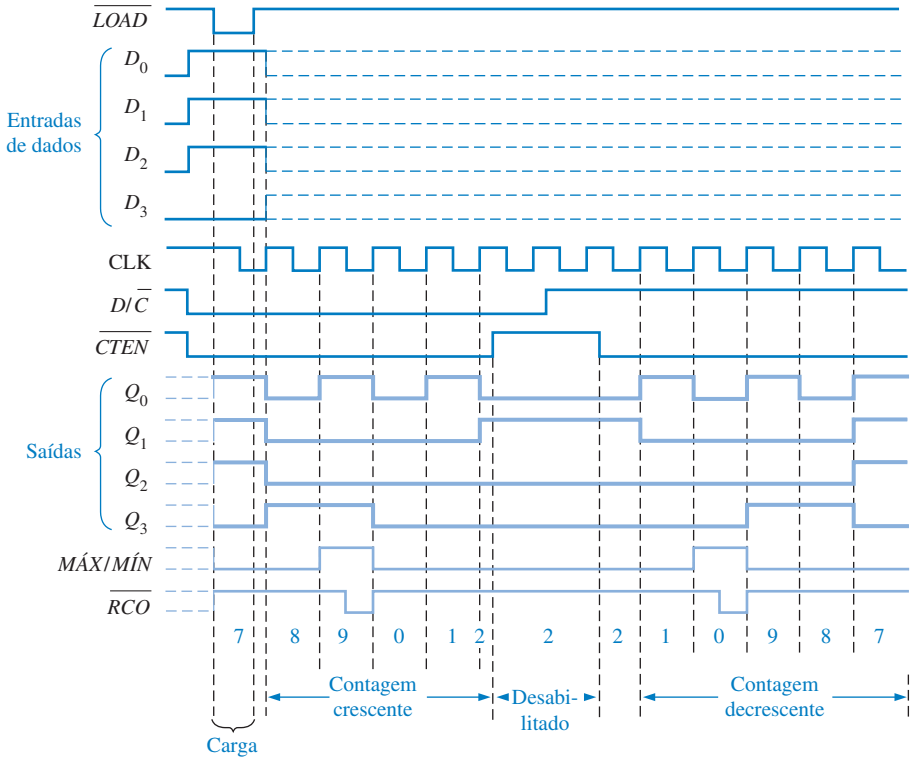


▶ **FIGURA 8–25**

O CI contador de década síncrono crescente/decrescente 74HC190.

A saída MÁX/MÍN produz um pulso de nível ALTO quando a contagem final nove (1001) é alcançada no modo CRESCENTE ou quando a contagem final zero (0000) é alcançada no modo DECRESCENTE. Essa saída MÁX/MÍN, juntamente com a saída de clock ondulante $(\overline{RCO})$ e a entrada de habilitação de contagem $(\overline{CTEN})$, é usada para conexão em cascata de contadores. (A conexão em cascata de contadores é discutida na Seção 8–5.)

A Figura 8–26 é um diagrama de temporização que mostra o CI contador 74HC190 preseta-do para sete (0111) e em seguida percorrendo uma seqüência de contagem crescente seguida de uma seqüência decrescente. A saída MÁX/MÍN é nível ALTO quando o contador estiver com todos os estados 0 (MÍN) ou o estado 1001 (MÁX).



◀ **FIGURA 8–26**
Exemplo de temporização para um CI 74HC190.

SEÇÃO 8–3 REVISÃO

1. Um contador binário crescente/decrescente de 4 bits está no modo DECRESCENTE e no estado 1010. No próximo pulso de clock, para qual estado irá o contador?
2. Qual será a contagem final de um contador binário de 4 bits no modo CRESCENTE? E no modo DECRESCENTE? Qual será o próximo estado após a contagem final no modo DECRESCENTE?

8-4 PROJETO DE CONTADORES SÍNCRONOS

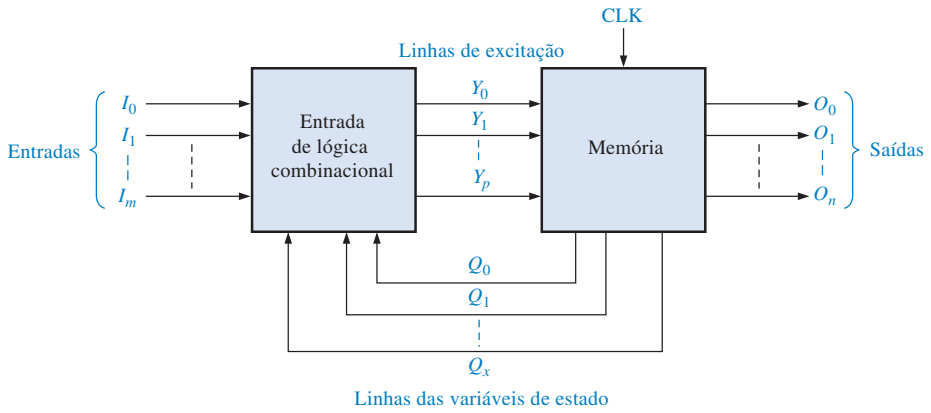
Nesta seção, veremos como as técnicas de projeto de circuitos seqüenciais podem ser aplicadas especialmente no projeto de contadores. Em geral, os circuitos seqüenciais podem ser classificados em dois tipos: (1) aquele no qual a(s) saída(s) depende(m) apenas do estado atual interno (denominado *circuitos Moore*) e (2) aquele no qual a(s) saída(s) depende(m) do estado atual e da(s) entrada(s) (denominados de *circuitos Mealy*). Essa seção é recomendada para aqueles que desejam uma introdução ao projeto de contadores ou o projeto de máquinas de estado em geral. Essa seção não é pré-requisito para qualquer outro material.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever um circuito seqüencial em termos das partes básicas e das entradas e saídas ■ Desenvolver um diagrama de estado para uma seqüência dada ■ Desenvolver uma tabela do próximo estado para uma seqüência de contador especificada ■ Criar uma tabela de transição de flip-flop ■ Usar o método do mapa de Karnaugh para deduzir os requisitos lógicos para um contador síncrono ■ Implementar um contador para produzir uma seqüência especificada de estados

Modelo Geral de um Circuito Seqüencial

Antes de abordar as técnicas de projeto de contadores específicos, vamos começar com uma definição geral de um **circuito seqüencial** ou **máquina de estados**: um circuito seqüencial geral consiste de uma seção de lógica combinacional e uma seção de memória (flip-flops), como mostra a Figura 8–27. Num circuito seqüencial com clock, existe uma entrada de clock para a seção de memória como indicado.



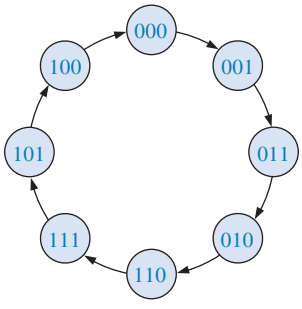
▶ **FIGURA 8–27**
Circuito seqüencial geral com clock.

A informação armazenada na seção da memória, bem como as entradas da lógica combinacional (I_0, I_1,..., I_m), são necessárias para a operação adequada do circuito. Para qualquer instante dado, a memória está num estado denominado *estado atual* e avança para o próximo estado num pulso de clock conforme determinado pelas condições das linhas de excitação (Y_0, Y_1,..., Y_p). O estado atual da memória é representado pelas variáveis de estado (Q_0, Q_1,..., Q_x). Essas variáveis de estado, juntamente com as entradas (I_0, I_1,..., I_m), determinam as saída do sistema (O_0, O_1,..., O_n).

Note que todos os circuitos seqüenciais têm variáveis de entrada e saída conforme o modelo geral discutido. Entretanto, todos têm variáveis de excitação e variáveis de estado. Os contadores são um caso especial de circuitos seqüenciais com clock. Nessa seção, aplicamos, em contadores síncronos, um procedimento geral de projeto para circuitos seqüenciais numa série de passos.

Passo 1: Diagrama de Estados

O primeiro passo no projeto de um contador é criar um diagrama de estados. Um **diagrama de estados** mostra progressão de estados através dos quais o contador avança quando recebe clock. Como exemplo, a Figura 8–28 é um diagrama de estados para um contador de código Gray de 3 bits. Esse circuito em particular não tem outras entradas além do clock nem outras saídas além das saídas obtidas de cada flip-flop do contador. Talvez seja necessário nesse momento revisar a abordagem sobre código Gray no Capítulo 2.



▶ **FIGURA 8–28**
Diagrama de estados para um contador de código Gray.

Passo 2: Tabela do Próximo Estado

Uma vez definido o circuito seqüencial pelo diagrama de estados, o segundo passo é deduzir a tabela do próximo estado, a qual apresenta cada estado do contador (estado atual) juntamente com o próximo estado correspondente. *O próximo estado é o estado para o qual o contador passa a*

partir do estado atual com a aplicação de um pulso de clock. A tabela do próximo estado é deduzida a partir do diagrama de estados e é mostrada na Tabela 8–7 para o contador de código Gray de 3 bits. Q_0 é o bit menos significativo.

◀ TABELA 8–7

Tabela do próximo estado para um contador de código Gray de 3 bits

ESTADO ATUAL			PRÓXIMO ESTADO		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

Passo 3: Tabela de Transição de Flip-Flop

A Tabela 8–8 é uma tabela de transição para o flip-flop J-K. Todas as transições de saída são relacionadas mostrando a saída Q do flip-flop indo dos estados atuais para os próximos estados. Q_N é o estado atual do flip-flop (antes do pulso de clock) e Q_{N+1} é o próximo estado (após o pulso de clock). Para cada transição de saída, as entrada J e K que fazem com que ocorram a transição são relacionadas. Um X indica um "don't care" (a entrada que pode ser 1 ou 0).

◀ TABELA 8–8

Tabela de transição para um flip-flop J-K

TRANSIÇÕES DE SAÍDA			ENTRADAS DO FLIP-FLOP	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Q_N: estado atual

Q_{N+1}: próximo estado

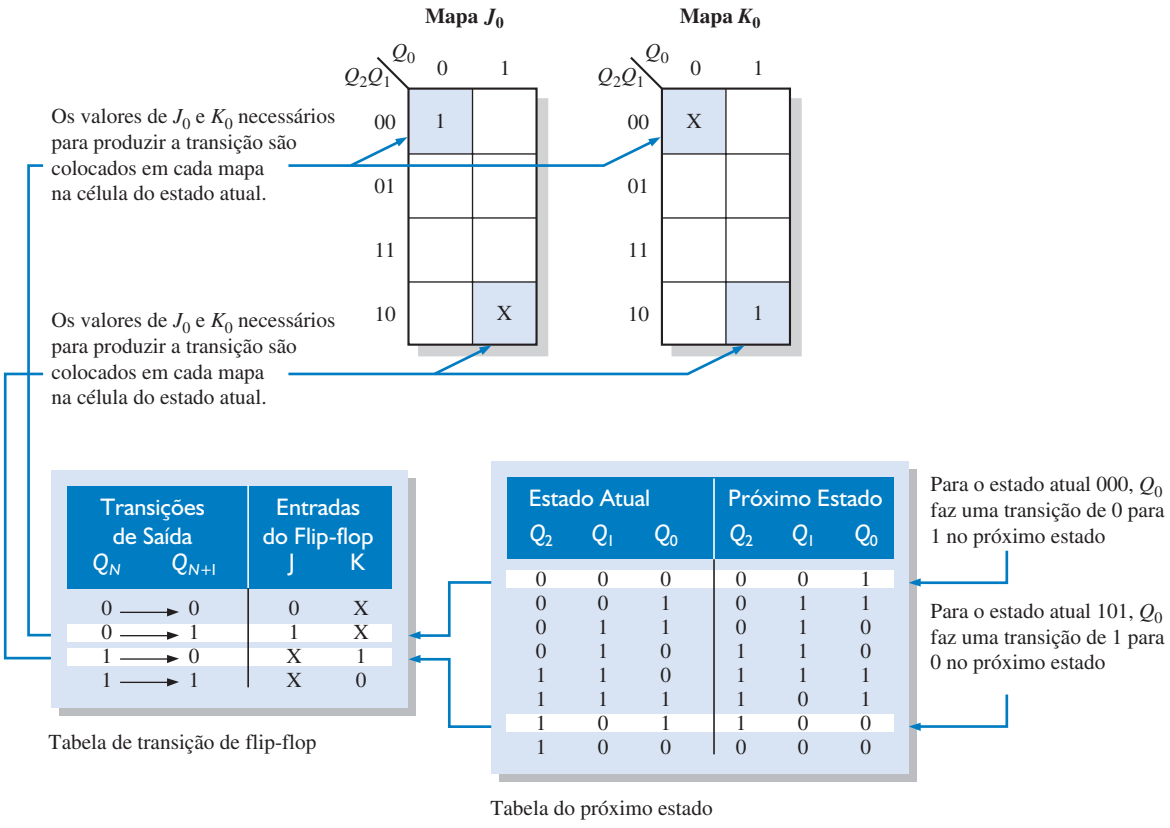
X: "don't care"

Para projetar o contador, a tabela de transição é aplicada para cada um dos flip-flops do contador, baseado na tabela do próximo estado (Tabela 8–7). Por exemplo, para o estado atual 000, Q_0 passa do estado atual que é 0 para o próximo estado que é 1. Para isso acontecer, J_0 tem que ser nível 1 e não importa o estado de K_0 ($J_0 = 1$, $K_0 = X$), como podemos ver na tabela de transição (Tabela 8–8). Em seguida, Q_1 é nível 0 no estado atual e permanece em 0 no próximo estado. Para essa transição, $J_1 = 0$ e $K_1 = X$. Finalmente, Q_2 é nível 0 no estado atual e permanece em 0 no próximo estado. Portanto, $J_2 = 0$ e $K_2 = X$. Essa análise se repete para cada um dos estados atuais na Tabela 8–7.

Passo 4: Mapas de Karnaugh

Os mapas de Karnaugh podem ser usados para determinar a lógica necessária para as entradas J e K de cada flip-flop no contador. Existe um mapa de Karnaugh para a entrada J e outra para a entrada K de cada flip-flop. Nesse procedimento de projeto, cada célula no mapa de Karnaugh representa um dos estados atuais na seqüência do contador relacionada na Tabela 8–7.

A partir dos estados de J e K na tabela de transição (Tabela 8–8) um nível 1, nível 0 ou um X é colocado em cada célula de estado atual no mapa dependendo da transição da saída Q para um

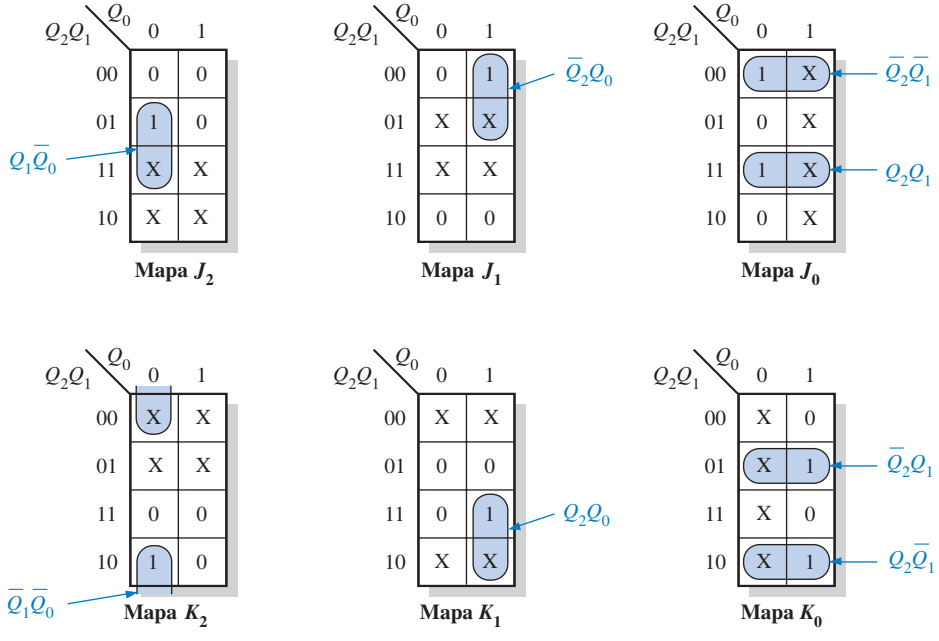


▲ **FIGURA 8–29**

Exemplos de procedimento de mapeamento para a seqüência do contador representado na Tabela 8–7 e Tabela 8–8.

flip-flop em particular. Para ilustrar esse procedimento, duas amostras inseridas são mostradas para as entradas J_0 e K_0 para o flip-flop menos significativo (Q_0) na Figura 8–29.

Os mapas de Karnaugh completos para todos os três flip-flops no contador são mostrados na Figura 8–30. As células são agrupadas como indicado e as expressões Booleanas correspondentes para cada grupo são deduzidas.



▶ **FIGURA 8–30**

Mapas de Karnaugh para o estado atual das entradas *J* e *K*.

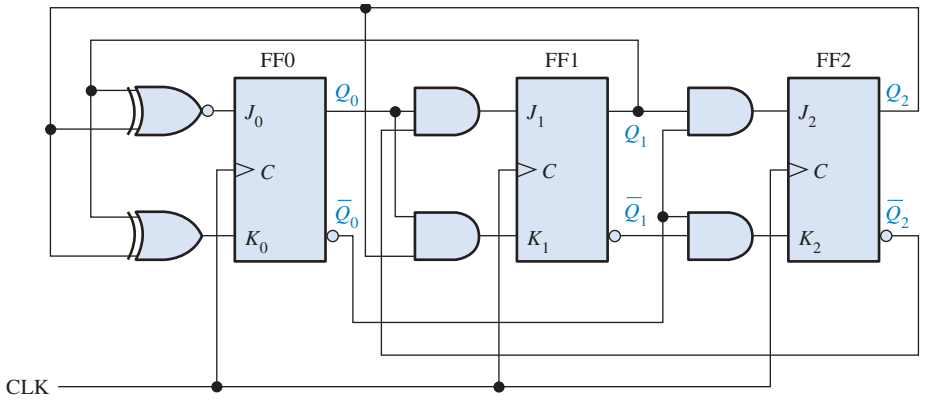
Passo 5: Expressões Lógicas para as Entradas dos Flip-flops

A partir dos mapas de Karnaugh da Figura 8–30 obtemos as seguintes expressões para as entradas *J* e *K* de cada flip-flop:

$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2Q_0$$
$$K_1 = Q_2Q_0$$
$$J_2 = Q_1\overline{Q}_0$$
$$K_2 = \overline{Q}_1\overline{Q}_0$$

Passo 6: Implementação do Contador

O passo final é a implementação da lógica combinacional a partir das expressões para as entradas *J* e *K* e a conexão dos flip-flops para formar o contador de código Gray completo de 3 bits como mostra a Figura 8–31.



▲ **FIGURA 8–31**

Contador de código Gray de três bits. Abra o arquivo F08-31 para verificar a operação.

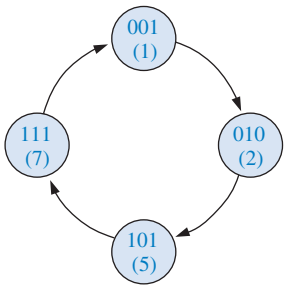
A seguir é mostrado um resumo dos passos seguidos no projeto desse contador. Em geral, esses passos podem ser aplicados a qualquer circuito seqüencial.

1. Especifique a seqüência do contador e desenhe o diagrama de estados.
2. Deduza a tabela do próximo estado a partir do diagrama de estados.
3. Desenvolva uma tabela de transição mostrando as entradas dos flip-flops necessárias para cada transição. A tabela de transição é sempre a mesma para um dado tipo de flip-flop.
4. Transfira os estados de *J* e *K* a partir da tabela de transição para os mapas de Karnaugh. Existe um mapa de Karnaugh para cada entrada de cada flip-flop.
5. Agrupe as células dos mapas de Karnaugh para garantir e deduzir a expressão lógica para cada entrada de flip-flop.
6. Implemente as expressões com lógica combinacional e combine com os flip-flops para criar o contador.

Esse procedimento é aplicado agora ao projeto de outros contadores síncronos nos Exemplos 8–5 e 8–6.

EXEMPLO 8–5

Projete um contador com a seqüência de contagem binária irregular mostrada no diagrama de estados visto na Figura 8–32. Use flip-flops J-K.



▶ FIGURA 8–32

Solução

Passo 1: O diagrama de estados é mostrado. Embora existam apenas quatro estados, um contador de 3 bits é necessário para implementar seis seqüências porque a contagem máxima é sete. Como a seqüência necessária não inclui todos os estados binários possíveis, os estados inválidos (0, 3, 4 e 6) podem ser tratados como "don't cares" no projeto. Entretanto, caso o contador passe erroneamente por um estado inválido, temos que ter certeza de que ele volta para um estado válido.

Passo 2: A tabela do próximo estado é desenvolvida a partir do diagrama de estados e é dada na Tabela 8–9.

▶ TABELA 8–9
Tabela do próximo estado

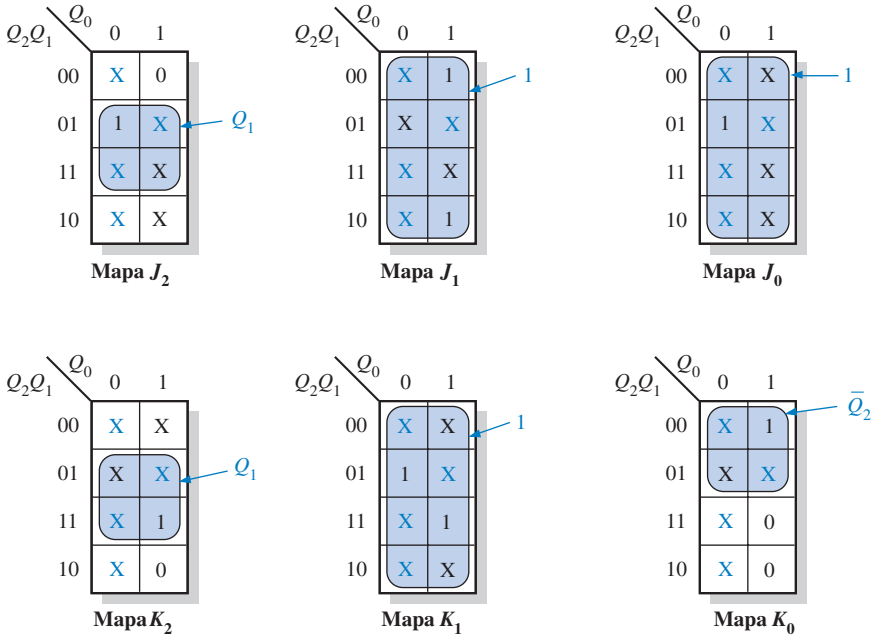
ESTADO ATUAL			PRÓXIMO ESTADO		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	0	1	0
0	1	0	1	0	1
1	0	1	1	1	1
1	1	1	0	0	1

Passo 3: A tabela de transição para o flip-flop J-K é repetida na Tabela 8–10.

▶ TABELA 8–10
Tabela de transição para um flip-flop J-K

TRANSIÇÕES DE SAÍDA			ENTRADAS DO FLIP-FLOP	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Passo 4: As entradas *J* e *K* são inseridas nos mapas de Karnaugh do estado atual mostrados na Figura 8–33. Além disso, condições "don't care" podem ser colocadas nas células correspondentes aos estados inválidos de 000, 011, 100 e 110, conforme indicado pelos Xs em laranja.

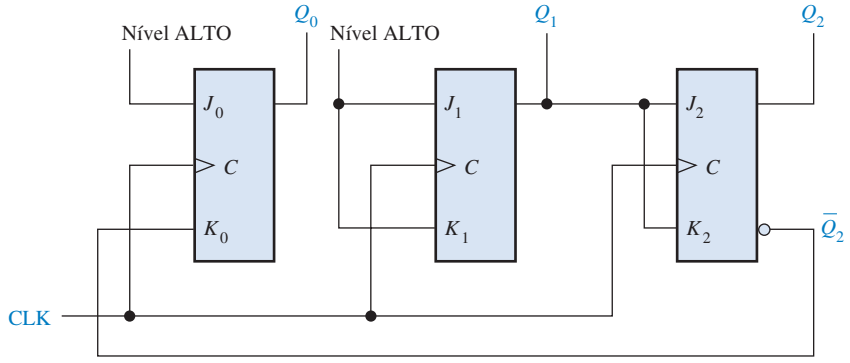


► FIGURA 8–33

Passo 5: Os grupos de 1s são formados com os estados "don't care" possíveis para se obter simplificação máxima, conforme mostra a Figura 8–33. Observe que quando *todas* as células do mapa são agrupadas, a expressão é simplesmente igual a 1. A expressão para cada entrada *J* e *K* obtida dos mapas são as seguintes:

$$J_0 = 1, K_0 = \overline{Q}_2$$
$$J_1 = K_1 = 1$$
$$J_2 = K_2 = Q_1$$

Passo 6: A implementação do contador é mostrada na Figura 8–34.



► FIGURA 8–34

Uma análise mostra que se o contador, acidentalmente, entrar em estados inválidos (0, 3, 4, 6), ele sempre retorna para um estado válido de acordo com as seqüências a seguir: 0 → 3 → 4 → 7 e 6 → 1.

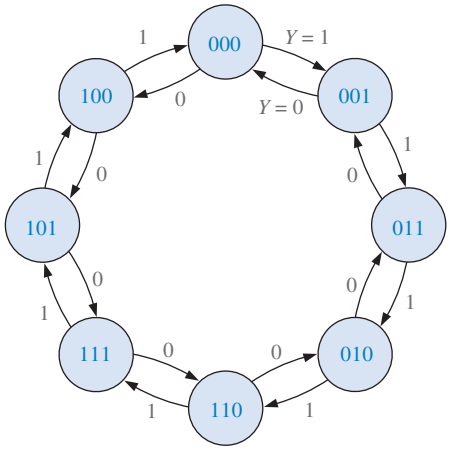
Problema relacionado Verifique a análise que prova que o contador sempre voltará (eventualmente) para um estado válido a partir de um estado inválido.

EXEMPLO 8–6

Desenvolva um contador crescente/decrescente de 3 bits síncrono com uma seqüência em código Gray. O contador deve contar de forma crescente quando a entrada de controle CRESCENTE/DECRESCENTE é nível 1 e conta de forma decrescente quando a entrada de controle é nível 0.

Solução

Passo 1 O diagrama de estados é mostrado na Figura 8–35. O nível 1 ou 0 ao lado de cada seta indica o estado da entrada CRESCENTE/DECRESCENTE de controle, *Y*.



▶ **FIGURA 8–35**

Diagrama de estados para um contador de código Gray crescente/decrescente de 3 bits.

Passo 2 A tabela do próximo estado é deduzida a partir do diagrama de estados e é mostrada na Tabela 8–11. Observe que para cada estado atual existem dois possíveis estados próximos, dependo da variável CRESCENTE/DECRESCENTE de controle, *Y*.

▼ **TABELA 8–11**

Tabela do próximo estado para um contador de código Gray crescente/decrescente de 3 bits

ESTADO ATUAL			PRÓXIMO ESTADO Y = 0 (DECRESCENTE)			PRÓXIMO ESTADO Y = 1 (CRESCENTE)		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

Y = entrada de controle CRESCENTE/$\overline{\text{DECRESCENTE}}$.

Passo 3 A tabela de transição para os flip-flops J-K é repetida na tabela 8–12.

TABELA 8–12

Tabela de transição para um flip-flop J-K

TRANSIÇÕES DE SAÍDA			ENTRADAS DO FLIP-FLOP	
Q_N		Q_{N+1}	J	K
0	⟶	0	0	X
0	⟶	1	1	X
1	⟶	0	X	1
1	⟶	1	X	0

Passo 4 Os mapas de Karnaugh para as entradas J-K dos flip-flops são mostradas na Figura 8–36. A entrada de controle CRESCENTE/DECRESCENTE (Y) é considerada uma das variáveis juntamente com Q_0, Q_1 e Q_2. Usando a tabela do próximo estado, a informação na coluna "Entradas do Flip-flop" da Tabela 8–12 é transferida para os mapas como indicado para cada estado atual do contador.

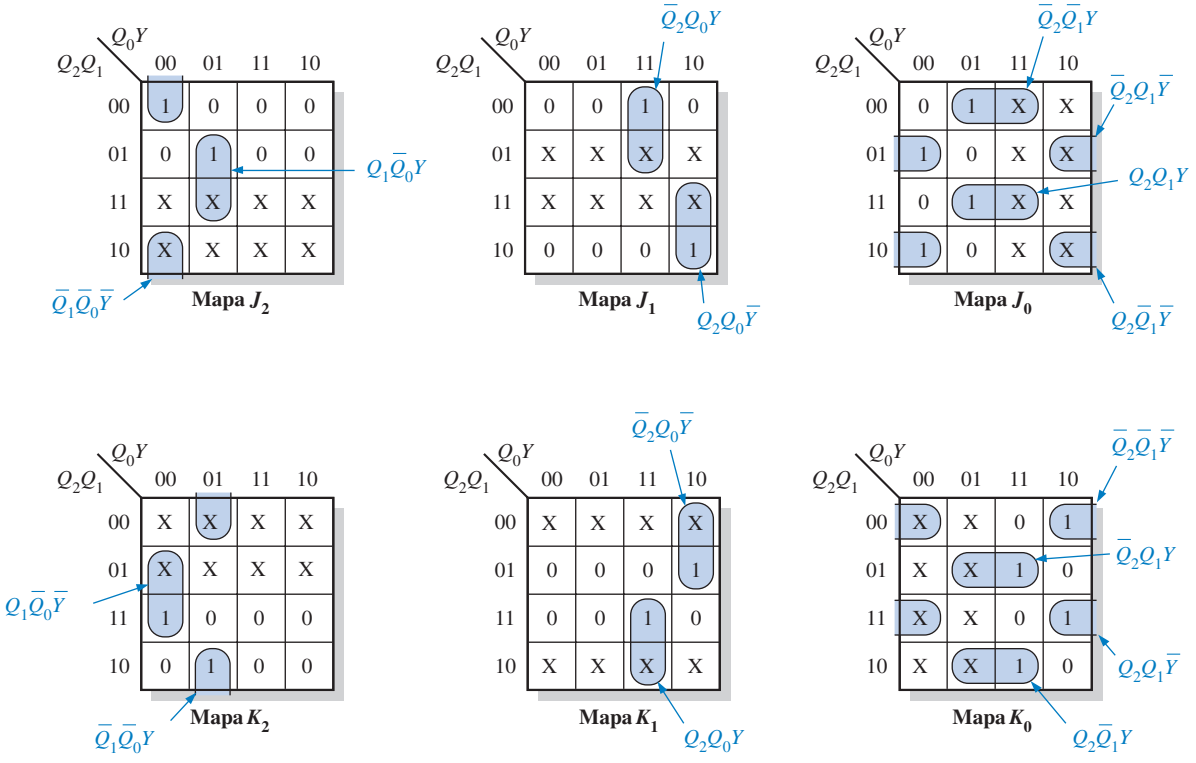


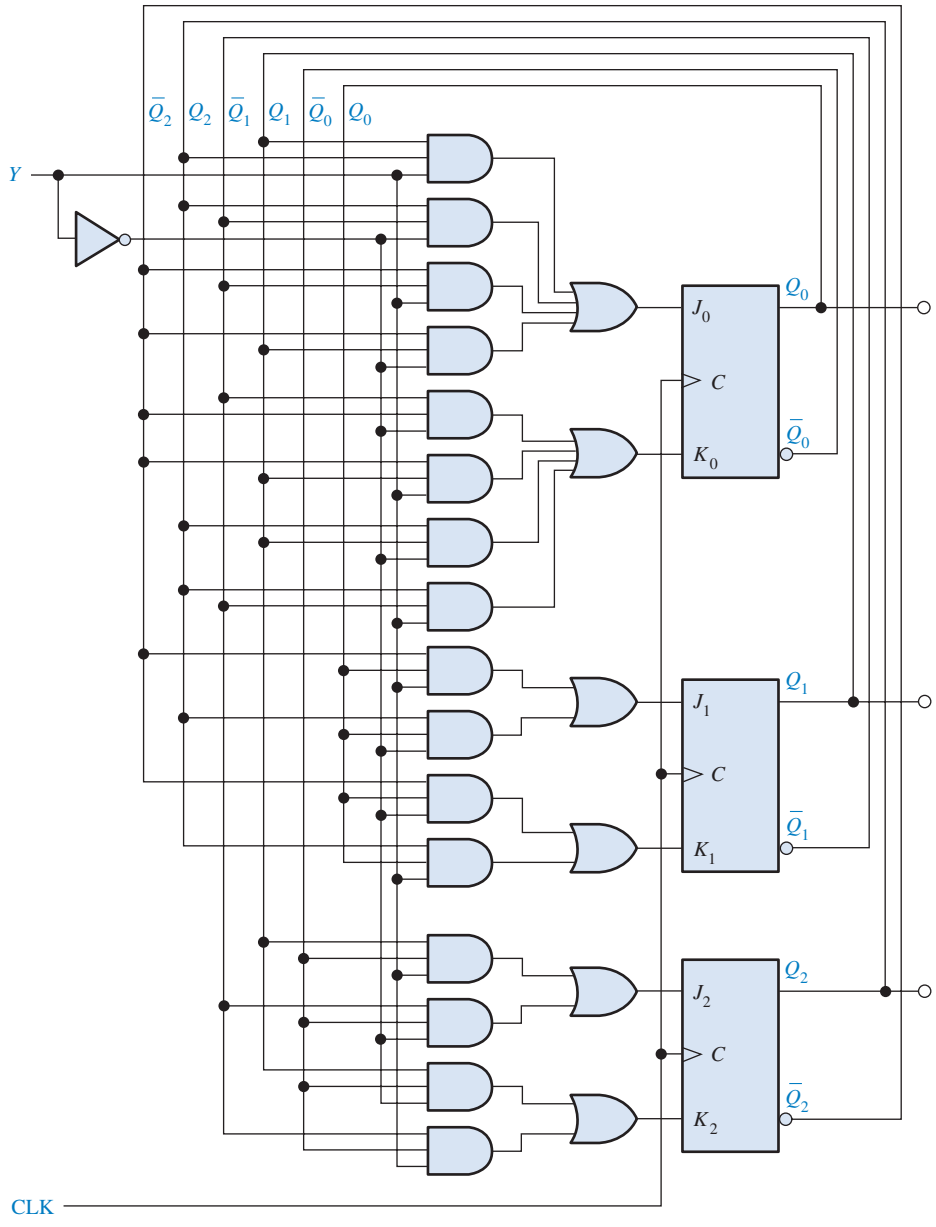
FIGURA 8–36

Mapas J e K para a Tabela 8–11. A entrada de controle CRESCENTE/DECRESCENTE (Y) é tratada como uma quarta variável.

Passo 5: Os 1s são combinados nos maiores agrupamentos possíveis, com "don't cares" (Xs) usados onde possível. Os grupos são fatorados e as expressões para as entradas J e K são:

$$J_0 = Q_2Q_1Y + Q_2\overline{Q}_1\overline{Y} + \overline{Q}_2\overline{Q}_1Y + \overline{Q}_2Q_1\overline{Y} \qquad K_0 = \overline{Q}_2\overline{Q}_1\overline{Y} + \overline{Q}_2Q_1Y + Q_2\overline{Q}_1Y + Q_2Q_1\overline{Y}$$
$$J_1 = \overline{Q}_2Q_0Y + Q_2Q_0\overline{Y} \qquad K_1 = \overline{Q}_2Q_0\overline{Y} + Q_2Q_0Y$$
$$J_2 = Q_1\overline{Q}_0Y + \overline{Q}_1\overline{Q}_0\overline{Y} \qquad K_2 = Q_1\overline{Q}_0\overline{Y} + \overline{Q}_1\overline{Q}_0Y$$

Passo 6: As equações de J e K são implementadas com lógica combinacional e o contador completo é mostrado na Figura 8–37.



▲ **FIGURA 8–37**

Contador de código Gray crescente/decrescente de três bits.

Problema relacionado Verifique que a lógica dada na Figura 8–37 concorda com as expressões no Passo 5.

SEÇÃO 8–4 REVISÃO

1. Um flip-flop está atualmente no estado RESET e tem que ir para o estado SET no próximo pulso de clock. Quais níveis lógicos têm que estar nas entradas *J* e *K*?
2. Um flip-flop está atualmente no estado SET e tem que ser mantido no estado SET no próximo pulso de clock. Quais níveis lógicos têm que estar nas entradas *J* e *K*?
3. Um contador binário está no estado $Q_3Q_2Q_1Q_0 = 1010$.
   (a) Qual é o próximo estado?
   (b) Qual condição tem que existir em cada entrada de flip-flop para garantir que ele passe para o estado correto no pulso de clock?

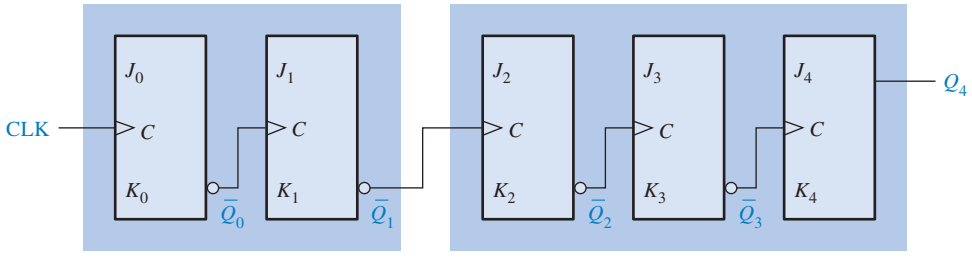
8-5 CONTADORES EM CASCATA

Os contadores podem ser associados por uma conexão em cascata para se conseguir operações com módulos maiores. Em essência, a **conexão em cascata** significa que a saída do último estágio de um contador aciona a entrada do próximo contador.

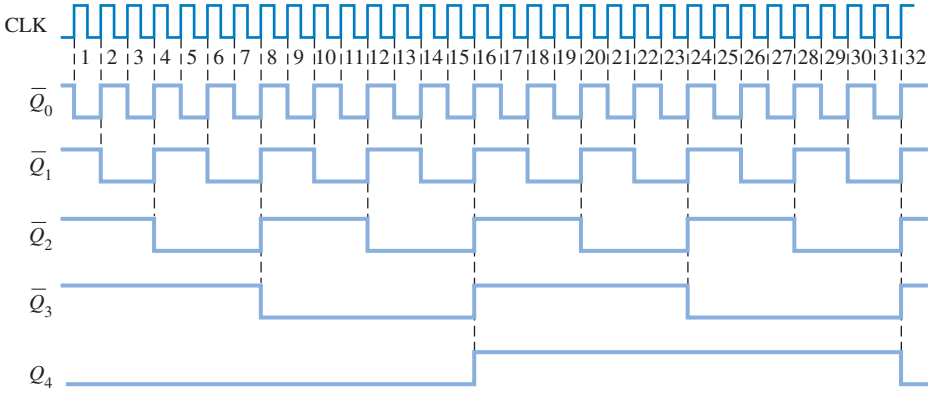
Ao final do estudo desta seção você deverá ser capaz de:

■ Determinar o módulo total de contadores em cascata ■ Analisar o diagrama de temporização de uma configuração de contadores em cascata ■ Usar contadores em cascata como um divisor de freqüência ■ Usar contadores em cascata para obter seqüências truncadas especificadas

Um exemplo de dois contadores conectados em cascata é mostrado na Figura 8–38 para um contador ondulante de 2 e 3 bits. O diagrama de temporização é mostrado na Figura 8–39. Observe que a saída final de um contador de módulo 8 (Q_4) ocorre uma vez a cada 32 pulsos de clock de



◀ **FIGURA 8–38**
Dois contadores em cascata (todas as entradas *J* e *K* são nível ALTO).



◀ **FIGURA 8–39**
Diagrama de temporização para a configuração de um contador em cascata dada na Figura 8–38.

O módulo total de contadores em cascata é igual ao produto dos módulos individuais.

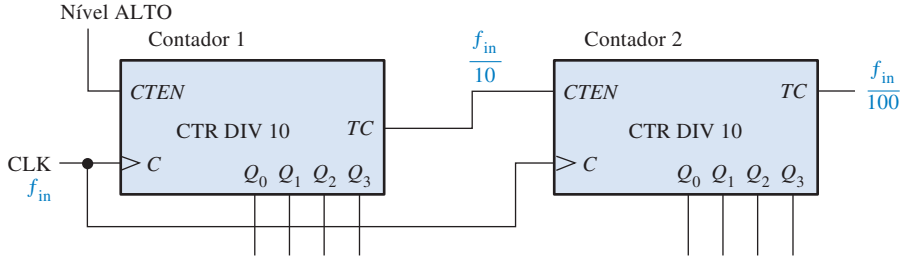
entrada. O módulo total dos contadores em cascata é 32; ou seja, ele atua como um contador/divisor por 32.

Quando se opera contadores síncronos numa configuração em cascata, é necessário usar as funções de habilitação de contagem e de contagem final para conseguir operações com módulos maiores. Em alguns dispositivos, a habilitação de contagem é identificada simplesmente por *CTEN* (ou alguma outra designação tal como *G*) e a contagem final (*TC*) é análogo à saída de clock ondulante (*RCO*) em alguns CIs contadores.

A Figura 8–40 mostra dois contadores de década conectados em cascata. A saída de contagem final (*TC*) do contador 1 é conectada na entrada de habilitação de contagem (*CTEN*) do contador 2. O contador 2 é desabilitado por um nível BAIXO na entrada *CTEN* até que o contador 1 alcance seu último estado, ou contagem final, e a sua saída de contagem final vai para nível ALTO. Esse nível ALTO agora habilita o contador 2, de forma que quando o primeiro pulso de clock após o contador 1 alcançar sua contagem final (CLK10), o contador 2 passa do seu estado inicial para o segundo estado. Ao final do segundo ciclo do contador 1 (quando o contador 1 alcança a contagem final pela segunda vez), o contador 2 é habilitado novamente e avança para o seu próximo estado. Essa seqüência continua. Como esses são contadores de década, o contador 1 tem que completar dez ciclos antes que o contador 2 complete seu primeiro ciclo. Em outras palavras, para cada dez ciclos do contador 1, o contador 2 avança um ciclo. Portanto, o contador 2 completará um ciclo após cem pulsos de clock. O módulo total desses dois contadores em cascata é $10 \times 10 = 100$.

NOTA: COMPUTAÇÃO

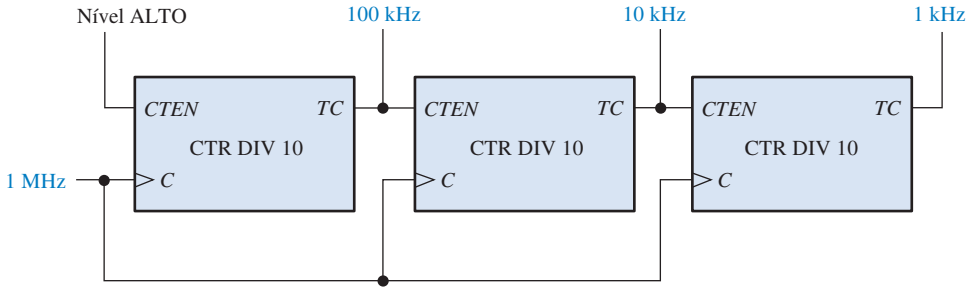
O *time stamp counter* (*TSC*), mencionado na nota de computação anterior, é um contador de 64 bits. É interessante observar que se esse contador (ou qualquer contador de 64 bits de módulo completo) que recebe pulsos de clock a uma freqüência de 100 MHz, leva 5849 anos para passar por todos os estados e alcançar sua contagem final. Em contraste, um contador de 32 bits de módulo completo passa por todos os estados em aproximadamente 43 segundos quando recebe clocks a 100MHz. A diferença é espantosa.



▲ **FIGURA 8–40**

Um contador de módulo 100 usando dois contadores de década em cascata.

Quando visto como um divisor de freqüência, o circuito mostrado na Figura 8–40 divide a freqüência do clock de entrada por 100. Os contadores em cascata são freqüentemente usados para dividir um sinal de clock de freqüência alta para obter freqüências de pulsos de maior precisão. As configurações de contadores em cascata usadas para tais finalidades são algumas vezes denominadas de *cadeias de contadores decrescentes*. Por exemplo, suponha que temos uma freqüência básica de clock de 1 MHz e desejamos obter 100 kHz, 10 kHz e 1 kHz; podem ser usados contadores de década em série. Se o sinal de 1 MHz for dividido por 10, a saída será 100 kHz. Em seguida, se o sinal de 100 kHz for dividido por 10, a saída será 10 kHz. Mais uma divisão por dez produz uma freqüência de 1 kHz. A implementação geral dessa cadeia de contadores decrescentes (em relação à freqüência) é mostrada na Figura 8–41.

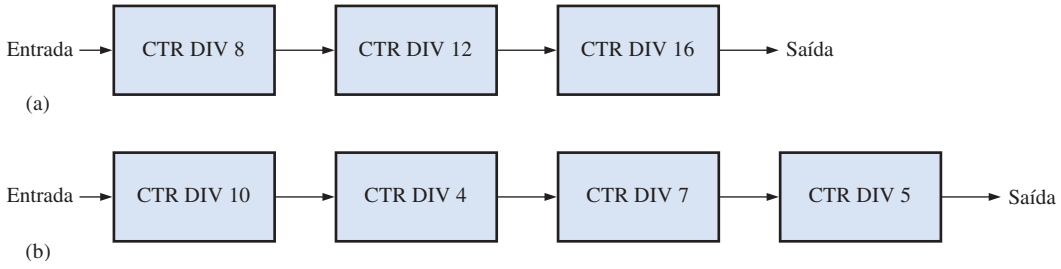


▶ **FIGURA 8–41**

Três contadores de década em cascata formando um divisor de freqüência por 1000 com saídas intermediárias de divisão por 10 e por 100.

EXEMPLO 8–7

Determine o módulo total da configuração de dois contadores em cascata mostrados na Figura 8–42.



▲ **FIGURA 8–42**

Solução Na Figura 8–42(a), o módulo total da configuração com 3 contadores é

$$8 \times 12 \times 16 = \mathbf{1536}$$

Na Figura 8–42(b), o módulo total da configuração com 4 contadores é

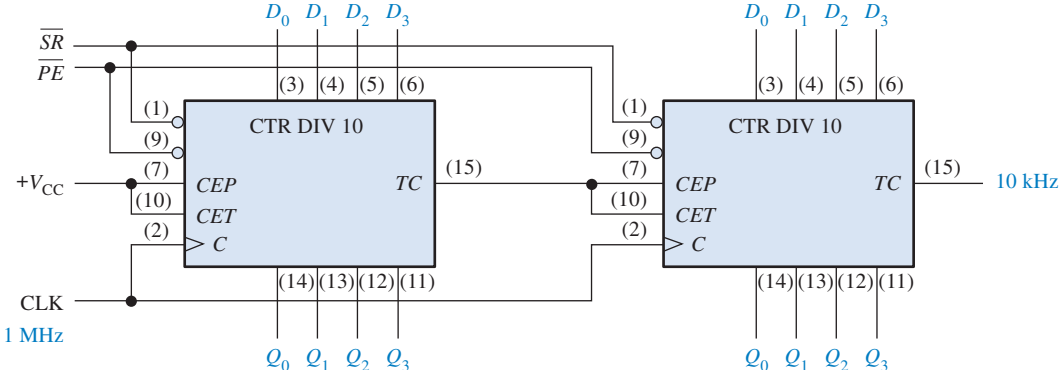
$$0 \times 4 \times 7 \times 5 = \mathbf{1400}$$

Problema relacionado Quantos contadores de década em cascata são necessários para dividir uma freqüência de clock por 100.000?

EXEMPLO 8–8

Use CIs contadores de década para obter uma forma de onda de 10 kHz a partir de um clock de 1 MHz. Mostre o diagrama lógico.

Solução Para obter 10 kHz a partir de 1 MHz é necessário um fator de divisão por 100. Dois CIs 74F162 têm que ser conectados em cascata conforme mostra a Figura 8–43. O contador da esquerda produz um pulso *TC* a cada 10 pulsos de clock. O contador da direita produz um pulso *TC* para cada 100 pulsos de clock.



▲ **FIGURA 8–43**

Um contador/divisor por 100 usando dois CIs contadores de década 74F162.

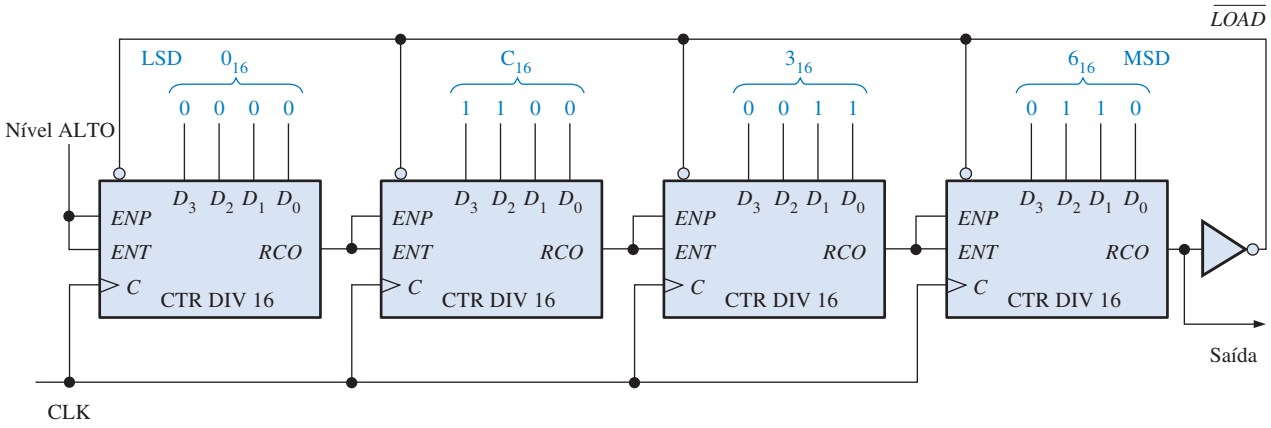
Problema relacionado Determine a freqüência da forma de onda na saída Q_0 do segundo contador (à direita) visto na Figura 8–43.

Contadores em Cascata com Seqüências Truncadas

A discussão anterior mostrou como obter um módulo total (fator de divisão) que é o produto dos módulos individuais de todos os contadores em cascata. Isso pode ser considerado uma *conexão em cascata de módulo total*.

Frequentemente uma aplicação necessita de um módulo total que é menor que a alcançada pela conexão em cascata de módulo total. Ou seja, uma seqüência truncada tem que ser implementada com contadores em cascata. Pra ilustrar esse método, usaremos a configuração de contador em cascata mostrada na Figura 8–44. Esse circuito em particular usa 4 CIs contadores binários síncronos de 4 bits 74HC161. Se esses quatro contadores (total de 16 bits) fossem conectados em cascata num arranjo de módulo total, o módulo seria

$$2^{16} = 65.536$$



▲ **FIGURA 8–44**

Um contador/divisor por 40.000 usando CIs contadores binários de 4 bits 74HC161. Observe que as entradas de cada dado em paralelo são mostradas em ordem binária (o bit mais à direita, D_0, é o LSB em cada contador).

Vamos considerar que uma certa aplicação necessita de um contador divisor por 40.000 (módulo 40.000). A diferença entre 65.536 e 40.000 é 25.536, que é o número de estados que tem que ser "apagado" da seqüência de módulo total. A técnica usada no circuito visto na Figura 8–44 é presetar o contador em cascata em 25.536 (63C0 em hexadecimal) a cada vez que ele for reciclado, de forma que ele conte de 25.536 até 65.535 em cada ciclo total. Portanto, cada ciclo total do contador consistirá de 40.000 estados.

Observe na Figura 8–44 que a saída *RCO* do contador mais à direita é invertida e aplicada na entrada $\overline{LOAD}$ de cada contador de 4 bits. Cada vez que o contador alcança o valor final de 65.535, que é 1111111111111111_2, *RCO* vai para nível ALTO e faz com que o número na entrada de dados em paralelo ($63C0_{16}$) seja carregado de forma síncrona no contador com o pulso de clock. Portanto, existe um pulso *RCO* do contador de 4 bits mais à direita a cada 40.000 pulsos de clock.

Com essa técnica qualquer módulo pode ser conseguido carregando de forma síncrona o contador com o estado inicial apropriado a cada ciclo.

SEÇÃO 8–5 REVISÃO

1. Quantos contadores de década são necessários para implementar um contador/divisor por 1000 (módulo 1000)? E um contador/divisor por 10.000?
2. Mostre com diagramas em bloco como obter os seguintes contadores usando um flip-flop, um contador de década e um contador binário de 4 bits ou qualquer combinação desses:
   (a) Contador/divisor por 20 **(b)** Contador/divisor por 32
   (c) Contador/divisor por 160 **(d)** Contador/divisor por 320

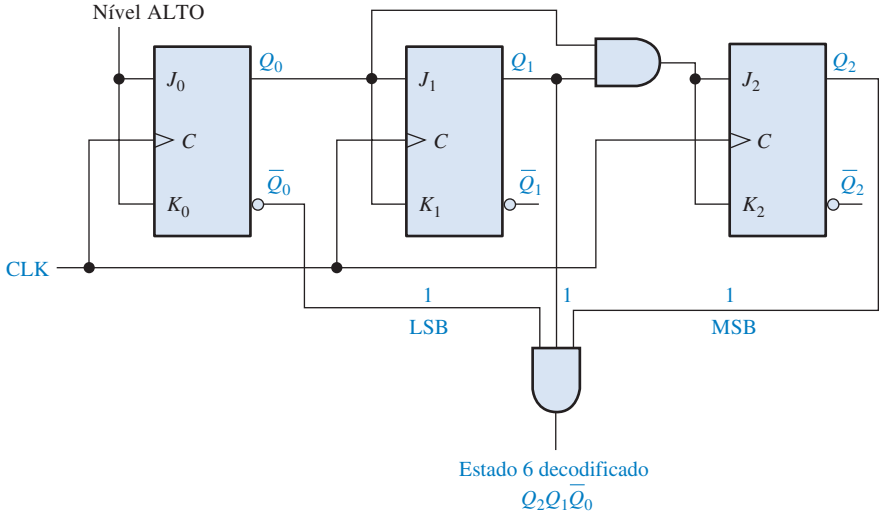
8-6 DECODIFICAÇÃO DE CONTADOR

Em muitas aplicações, é necessário que alguns ou todos os estados do contador sejam decodificados. A decodificação de um contador envolve o uso de decodificadores ou portas lógicas para determinar quando o contador estiver num determinado estado binário de sua seqüência. Por exemplo, a função de contagem final discutida anteriormente é um estado de decodificação único (o último estado) na seqüência do contador.

Ao final do estudo desta seção você deverá ser capaz de:

■ Implementar a lógica de decodificação para qualquer estado dado na seqüência de um contador ■ Explicar porque ocorrem glitches na lógica de decodificação de contadores ■ Usar o método de *strobing* para eliminar glitches de decodificação

Suponha que queremos decodificar o estado binário 6 (110) de um contador binário de 3 bits. Quando $Q_2 = 1$, $Q_1 = 1$ e $Q_0 = 0$, um nível ALTO aparece na saída da porta de decodificação indicando que o contador está no estado 6. Isso pode ser feito como mostra a Figura 8–45. Essa decodificação é denominada *decodificação ativa em nível ALTO*. A substituição da porta AND por uma porta NAND resulta numa decodificação ativa em nível BAIXO.

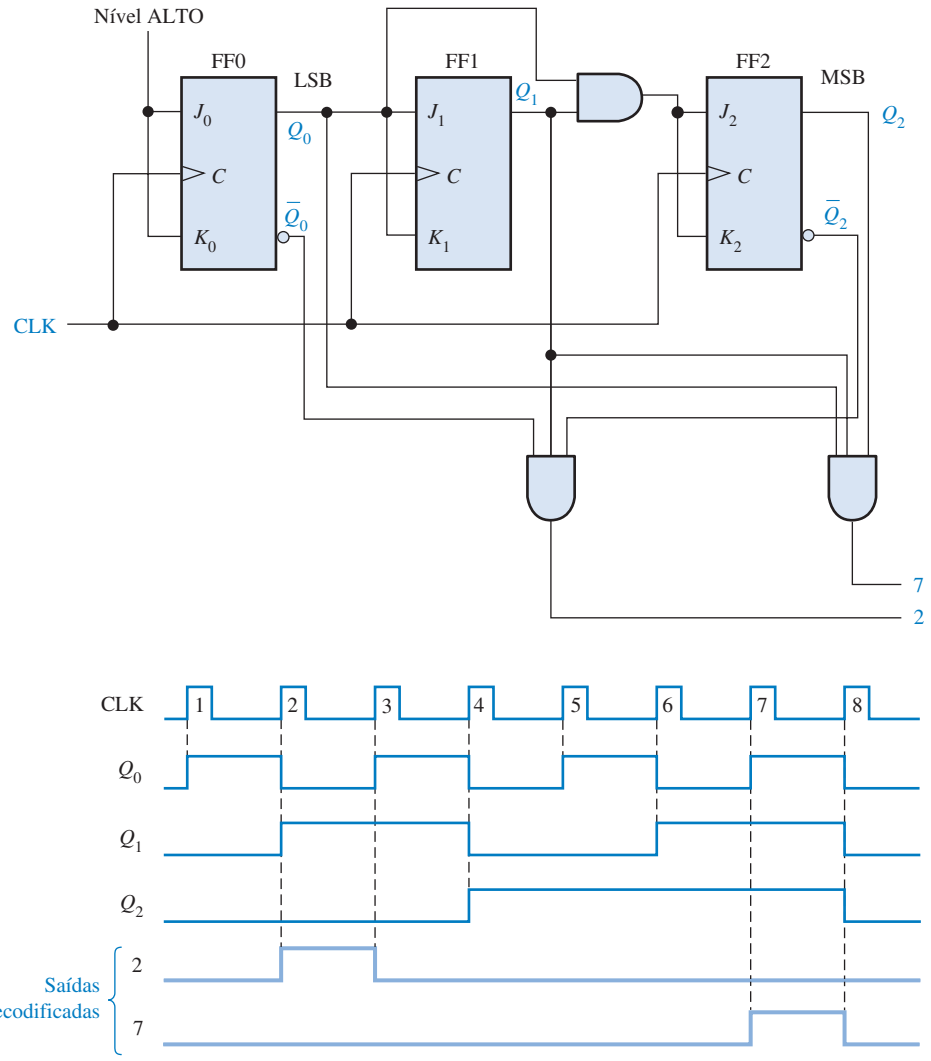


▲ **FIGURA 8–45**

Decodificação do estado 6 (110). Abra o arquivo F08-45 para verificar a operação.

EXEMPLO 8–9

Implemente a decodificação do estado binário 2 e do estado binário 7 de um contador síncrono de 3 bits. Mostre o diagrama de temporização completo do contador e as formas de onda de saída das portas de decodificação. O binário 2 = $\overline{Q}_2Q_1\overline{Q}_0$ e o binário 7 = $Q_2Q_1Q_0$.



▶ **FIGURA 8–46**

Um contador de 3 bits com decodificação ativa em nível ALTO das contagens 2 e 7. Abra o arquivo F08-46 para verificar a operação.

Solução Veja a Figura 8–46. O contador de 3 bits foi originalmente discutido na Seção 8–2 (Figura 8–14).

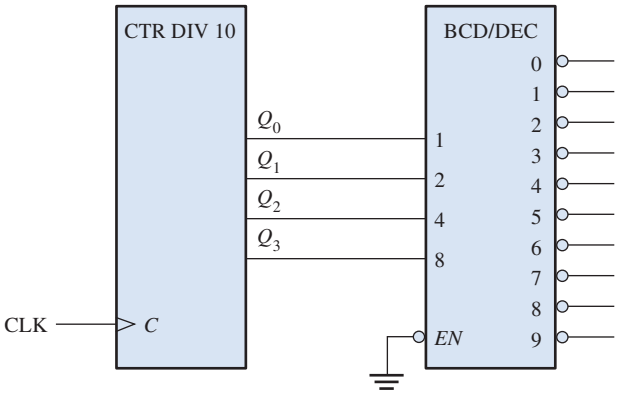
Problema relacionado Mostre a lógica para a decodificação do estado 5 num contador de 3 bits.

Glitches de Decodificação

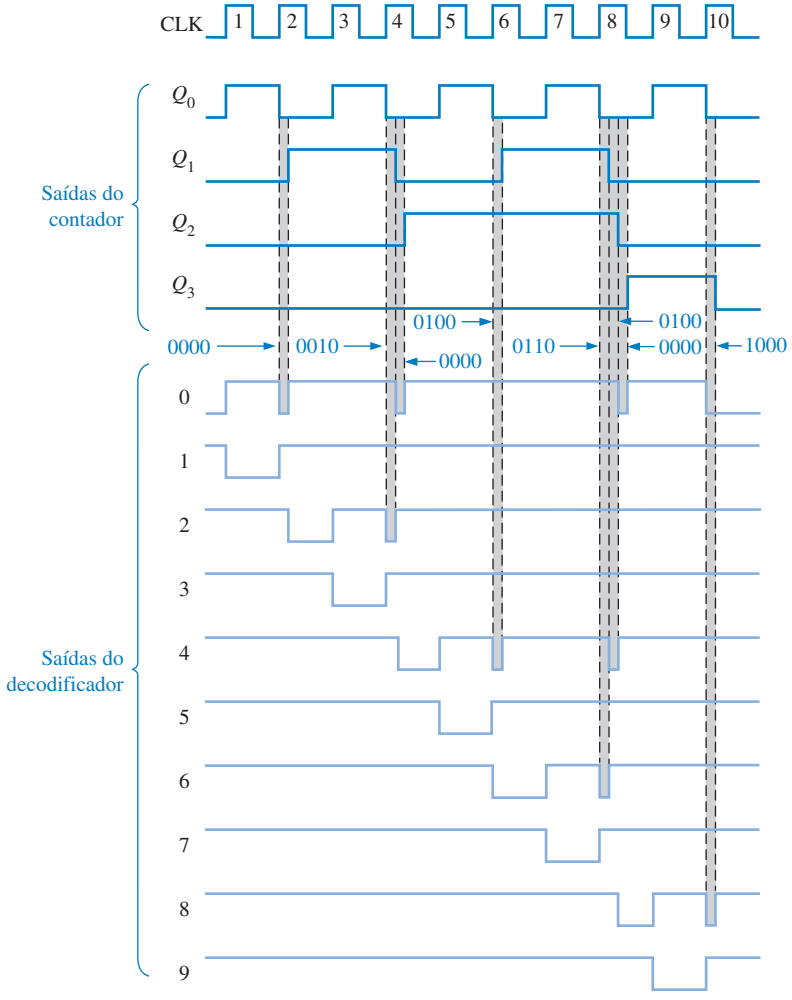
Um glitch é um spike de tensão não desejado.

O problema de glitches produzidos no processo de decodificação foi discutido no Capítulo 6. Conforme estudado, os atrasos de propagação devido ao efeito de ondulação (ripple) em contadores assíncronos cria estados de transição nos quais as saídas do contador mudam de estado em instantes ligeiramente diferentes. Esses estados de transição produzem spikes de tensão indesejados de curta duração (glitches) nas saídas do decodificador conectado ao contador. O problema do glitch também pode ocorrer em algum grau com contadores assíncronos porque os atrasos de propagação a partir do clock para as saídas Q de cada flip-flop num contador pode variar ligeiramente.

A Figura 8–47 mostra um contador de década BCD assíncrono básico conectado a um decodificador de BCD para decimal. Para ver o que acontece nesse caso, vamos analisar o diagrama de temporização no qual os atrasos de propagação são levados em consideração, como mostra a Figura 8–48. Observe que esses atrasos causam estados falsos de duração curta. O valor do estado



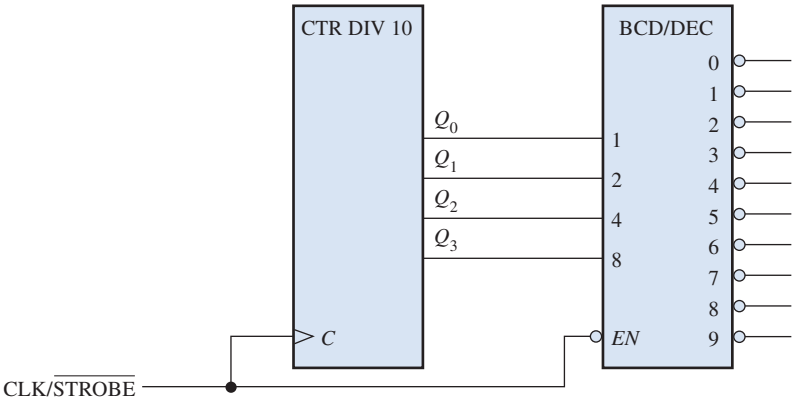
◀ **FIGURA 8–47**
Um contador de década (BCD) básico e decodificador.



◀ **FIGURA 8–48**
Saídas com glitches do decodificador mostrado na Figura 8–47. A largura dos glitches foram ampliadas por questão de ilustração sendo normalmente de alguns nanossegundos de largura.

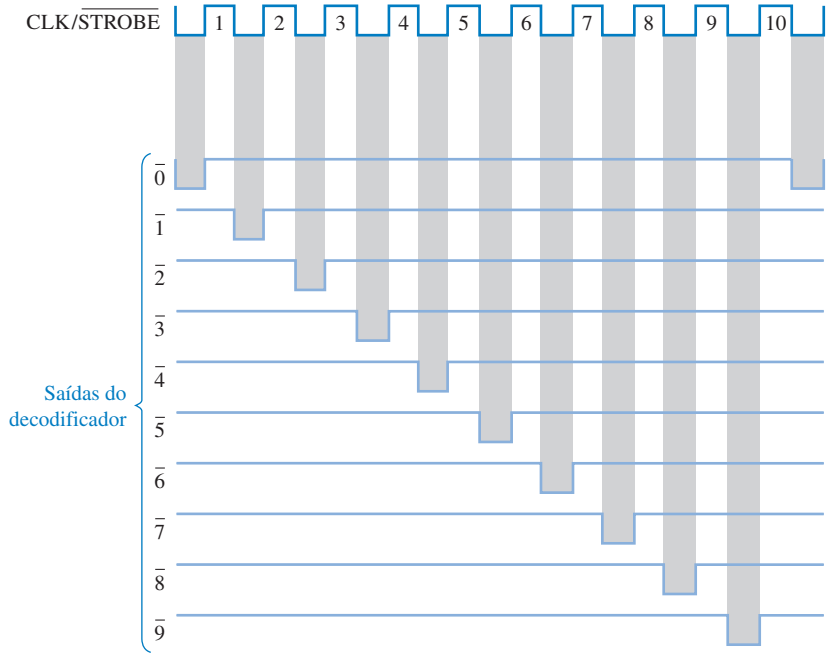
binário falso de cada transição crítica é indicado no diagrama. Os glitches podem ser vistos nas saídas do decodificador.

Uma forma de eliminar os glitches é habilitar as saídas do decodificador um tempo depois do glitch ter desaparecido. Esse método é denominado *strobing* e pode ser acompanhado, no caso de um clock ativo em nível ALTO, pelo nível BAIXO do clock para habilitar o decodificador, como mostra a Figura 8–49. O diagrama de temporização resultante melhorado é mostrado na Figura 8–50.



▶ **FIGURA 8–49**

Um contador de década básico e o decodificador com *strobing* para eliminar glitches.



▶ **FIGURA 8–50**

Saídas do decodificador com *strobing* para o circuito da Figura 8–49.

SEÇÃO 8–6 REVISÃO

1. Quais são os possíveis estados de transição quando um contador binário assíncrono de 4 bits muda
(a) da contagem 2 para a 3 **(b)** da contagem 3 para a 4
(c) da contagem 10 para a 11 **(d)** da contagem 15 para a 0.

8-7 APLICAÇÕES DE CONTADORES

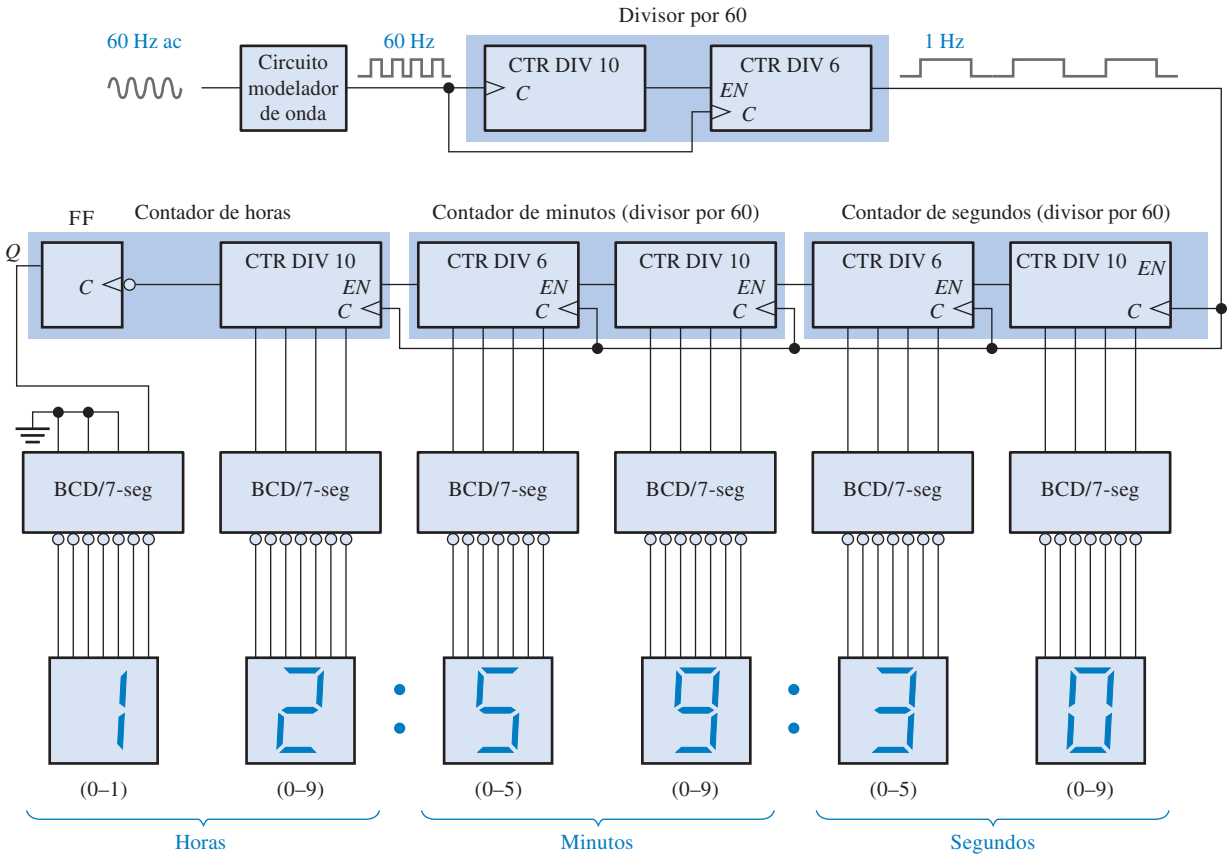
O contador digital é um dispositivo útil e versátil que é encontrado em muitas aplicações. Nesta seção, algumas aplicações representativas de contadores são apresentadas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever como os contadores são usados em sistemas digitais com clock ■ Explicar como um contador/divisor por 60 é implementado e como ele é usado num relógio digital ■ Explicar como o contador de horas é implementado ■ Discutir a aplicação de um contador num sistema de controle de estacionamento de veículos ■ Descrever como um contador é usado no processo de conversão de dados de paralelo para série.

Um Relógio Digital

Um exemplo comum de uma aplicação de contador é em sistemas de contagem de tempo. A Figura 8–51 é um diagrama lógico simplificado de um relógio digital que mostra segundos, minutos e horas. Primeiro, uma tensão ca senoidal de 60 Hz é convertida numa forma de onda de pulsos e dividida para uma forma de onda de pulsos de 1 Hz por um contador/divisor por 60 formado por um contador/divisor por 10 seguido de um contador/divisor por 6. Os contadores de *segundos* e *minutos* também são produzidos por contadores/divisores por 60 cujos detalhes são mostrados na Figura 8–52. esses contadores contam de 0 a 59 e em seguida reciclam para 0; contadores de década síncronos são usados nessa implementação em particular. Observe que a parte que divide por 6 é formada por um contador de década com uma seqüência truncada conseguida usando um decodificador da contagem 6 para resetar o contador de forma assíncrona. A contagem final (59) também é decodificada para habilitar o próximo contador da cadeia.

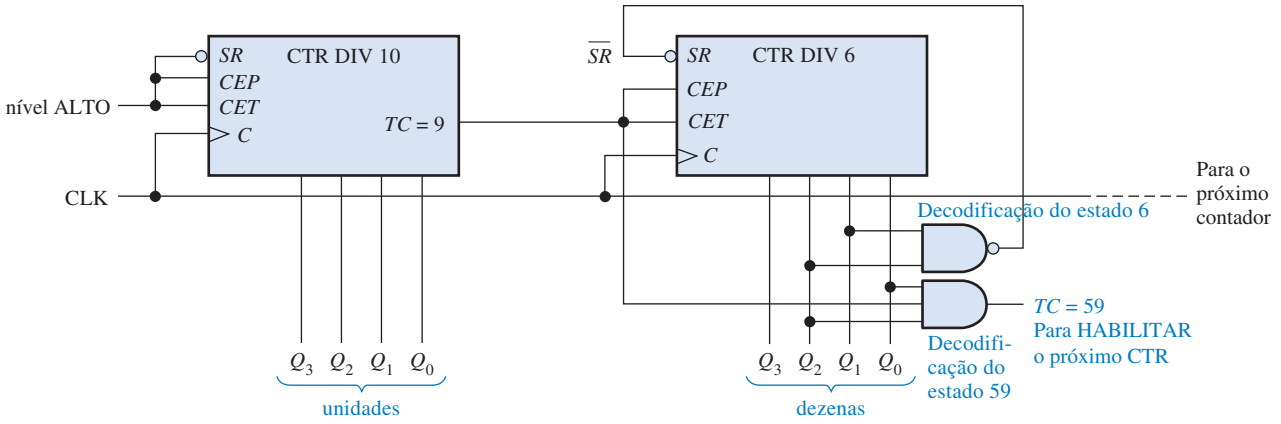


▲ **FIGURA 8–51**

Diagrama lógico simplificado para um relógio digital de 12 horas. Os detalhes lógicos usando dispositivos específicos são mostrados na Figura 8–52 e 8–53.

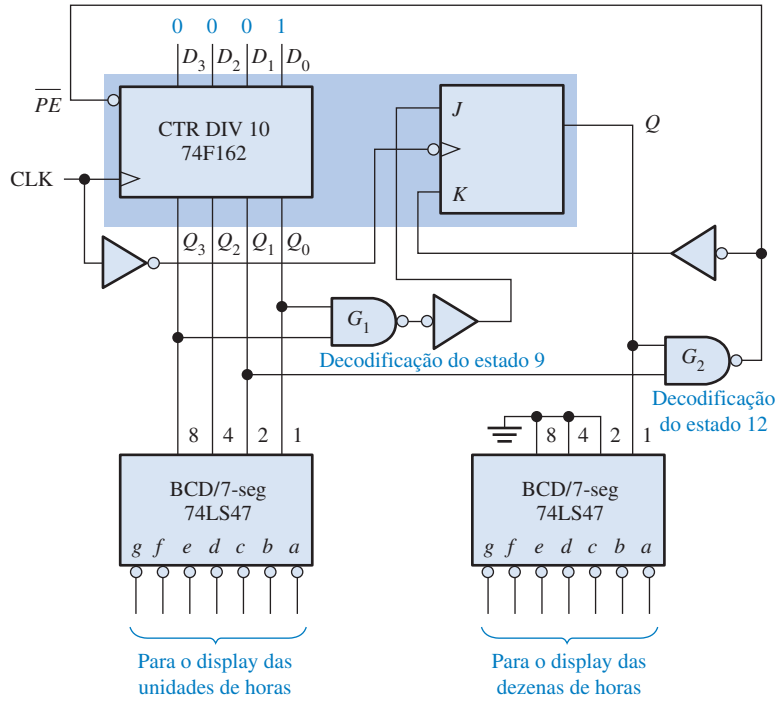
O contador de *horas* é implementado com um contador de década e um flip-flop como mostra a Figura 8–53. Considere que inicialmente os contadores de década e o flip-flop estejam resetados e as saídas das portas de decodificação dos estados 12 e 9 estejam em nível ALTO. Os contadores de década avançam passando por todos os estados de 0 a 9 e no pulso de clock que o recicla de volta para zero, o flip-flop é setado ($J = 1$, $K = 0$). Isso faz aparecer o número 1 no display das dezenas das horas. A contagem total agora é dez (o contador de década está no estado zero e o flip-flop está setado).

Em seguida, a contagem total avança para onze e em seguida para doze. No estado 12 a saída Q_2 do contador de década é nível ALTO, o flip-flop ainda está setado e assim a saída da porta de-



▲ **FIGURA 8–52**

Diagrama lógico de um contador/divisor por 60 usando CIs contadores síncronos de década 74F162. Observe que as saídas estão em ordem binária (o bit mais à direita é o LSB).



▶ **FIGURA 8–53**

Diagrama lógico para o contador de horas e decodificadores. Observe que nas entradas e saídas do contador o bit mais à direita é o LSB.

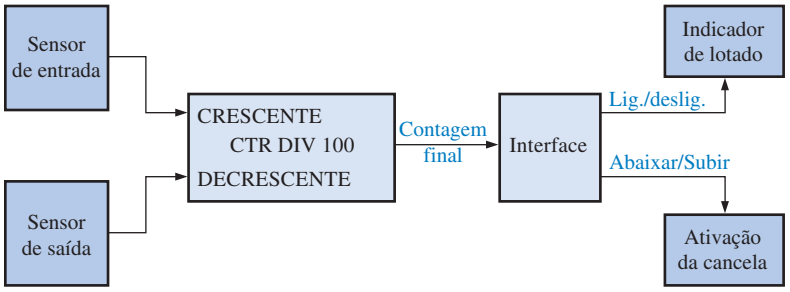
codificadora do estado 12 é nível BAIXO. Isso ativa a entrada $\overline{PE}$ do contador de década. No próximo pulso de clock, o contador de década é presetado para o estado 1 através das entradas de dados e o flip-flop é resetado ($J = 0$, $K = 1$). Como podemos ver, essa lógica sempre faz com que o contador recicle de doze para um em vez de retornar para zero.

Controle de Estacionamento de Veículos

Esse exemplo na aplicação de contadores ilustra o uso de um contador crescente/decrescente para resolver um problema do dia a dia. O problema é fazer um projeto de monitoração dos espaços disponíveis numa centena de vagas num estacionamento e provê uma indicação de uma condição de lotado através de uma indicação luminosa e abaixando a cancela na entrada.

Um sistema que resolve esse problema consiste de (1) um sensor optoeletrônico na entrada e outro na saída do estacionamento, (2) um contador crescente/decrescente com um circuito associado e (3) e um circuito de interface que usa a saída do contador para ligar ou desligar o sinal de

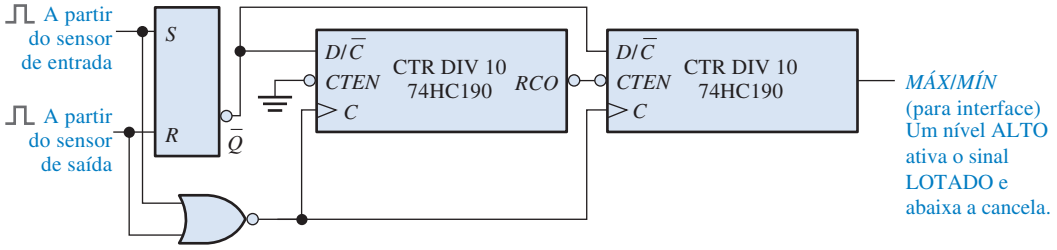
LOTADO e baixar ou subir a cancela na entrada. A Figura 8–54 mostra o diagrama em bloco geral desse sistema.



◀ **FIGURA 8–54**

Diagrama em bloco funcional para o controle de vagas num estacionamento.

Um diagrama lógico do contador crescente/decrescente (C/D) é mostrado na Figura 8–55. Ele consiste de dois CIs contadores de década 74HC190 conectados em cascata. A operação é descrita nos seguintes parágrafos.



▲ **FIGURA 8–55**

Diagrama lógico para um contador C/D de módulo 100 para o controle de um estacionamento de veículos.

O contador é inicialmente presetado em 0 usando as entradas de dados em paralelo, as quais não são mostradas. Cada veículo que entra no estacionamento interrompe o feixe de luz ativando um sensor que produz um pulso elétrico. Esse pulso positivo seta o latch S-R na borda de subida. O nível BAIXO na saída $\overline{Q}$ do latch coloca o contador no modo CRESCENTE. Além disso, o pulso do sensor vai para uma porta NOR e aciona o clock do contador na transição de nível BAIXO para nível ALTO (borda de subida). Cada vez que um veículo entra no estacionamento, o contador avança (**incrementa**) uma unidade. Quando uma centena de veículos entrarem, o contador vai para o seu último estado (100_{10}). A saída MÁX/MÍN vai para nível ALTO e ativa o circuito de interface (sem detalhamento), que ativa o sinal LOTADO e abaixa a cancela para evitar que mais veículos entrem.

O incremento de um contador aumenta a contagem dele em uma unidade.

Quando um veículo sai, um sensor optoeletrônico produz um pulso positivo, o qual reseta o latch S-R e coloca o contador em modo DECRESCENTE. A borda de descida do clock diminui (**decrementa**) a contagem em uma unidade. Se o estacionamento estiver lotado e um veículo sair, a saída MÁX/MÍN vai para nível BAIXO desligando o sinal LOTADO e subindo a cancela.

O decremento de um contador diminui a contagem dele em uma unidade.

Conversão de Dados de Paralelo para Série (Multiplexação)

Um exemplo simplificado de transmissão de dados usando técnicas de multiplexação e demultiplexação foi apresentado no Capítulo 6. Essencialmente, os bits de dados em paralelo nas entradas do multiplexador são convertidos para bits de dados em série numa única linha de transmissão. Os bits de um grupo que aparecem simultaneamente nas linhas em paralelo são denominados de *dados em paralelo*. Os bits de um grupo que aparecem numa única linha numa seqüência temporal são denominados de *dados seriais*.

A conversão de paralelo para série é normalmente acompanhada pelo uso de um contador para prover uma seqüência binária para a entrada de seleção de dados de um seletor/multiplexador, conforme ilustrado na Figura 8–56. As saídas Q do contador de módulo 8 são conectadas nas entradas de seleção de dados de um multiplexador de 8 bits.

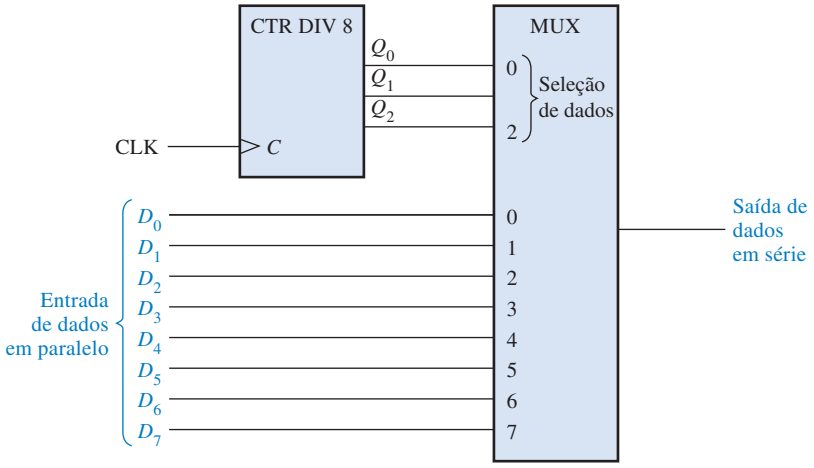


FIGURA 8–56

Lógica de conversão de dados em paralelo para série.

A Figura 8–57 é um diagrama de temporização que ilustra a operação desse circuito. O primeiro byte (grupo de oito bits) de dados em paralelo é aplicado nas entradas do multiplexador. À medida que o contador percorre uma seqüência binária de zero a sete, cada bit, começando por D_0, é selecionado seqüencialmente e passa através do multiplexador para a linha de saída. Após oito pul-

NOTA: COMPUTAÇÃO

Os computadores contêm um contador interno que pode ser programado para diversas freqüências e tons de duração, produzindo assim "música". Para selecionar um tom particular, a instrução programada seleciona um divisor que é enviado ao contador. O divisor configura o contador crescente para dividir a freqüência de clock de periférico básica para produzir um tom de áudio. A duração de um tom também pode ser configurada por uma instrução programada; portanto, um contador básico é usado para produzir melodia controlando a freqüência e a duração dos tons.

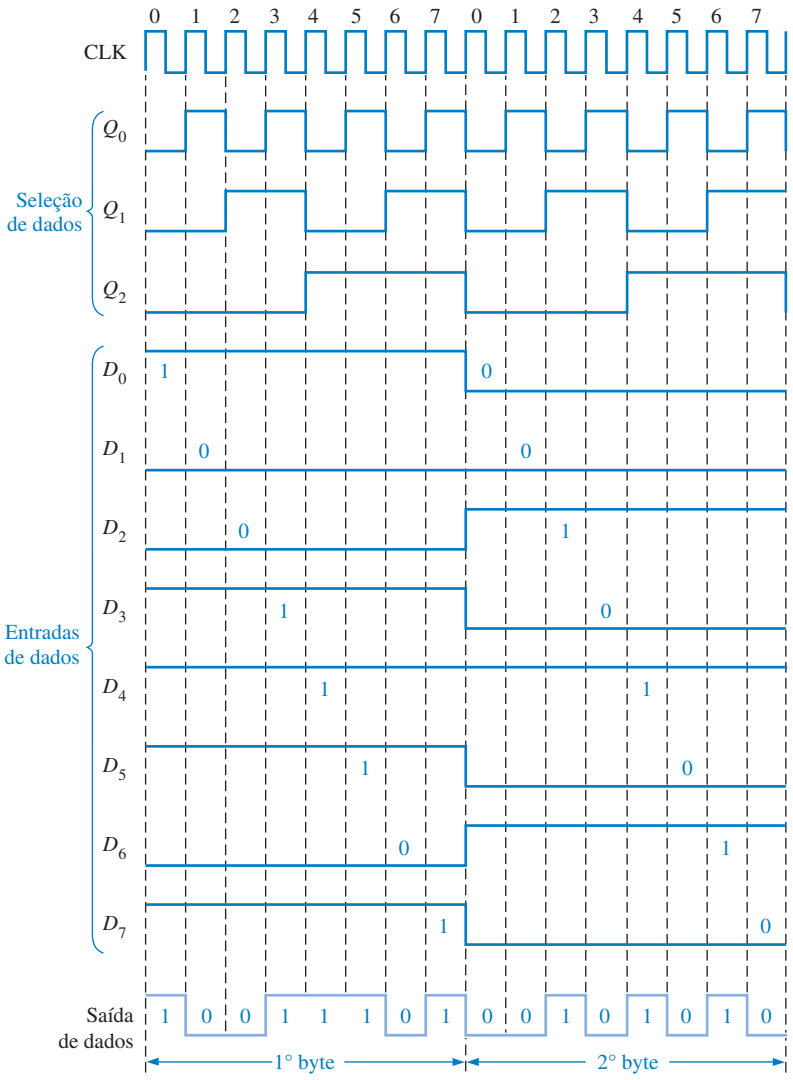


FIGURA 8–57

Exemplo de temporização da conversão de paralelo para série considerando o circuito dado na Figura 8–56.

sos de clock o byte de dados foi convertido para um formato serial e enviado pela linha de transmissão. Quando o contador recicla de volta para zero, o próximo byte é aplicado nas entradas de dados e é convertido sequencialmente para a forma serial conforme o contador percorre os seus oito estados. Esse processo se repete à medida que cada byte é convertido para um byte serial.

SEÇÃO 8–7 REVISÃO

1. Explique a finalidade de cada porta NAND na Figura 8–53.
2. Identifique as duas condições de reciclagem para o contador de horas dado na Figura 8–51 e explique o motivo de cada reciclagem.

8-8 SÍMBOLOS LÓGICOS COM NOTAÇÃO DE DEPENDÊNCIA

Até esse momento, os símbolos lógicos com notação de dependência especificados no padrão 91-1984 da ANSI/IEEE foram apresentados de forma limitada. Em muitos casos, os símbolos novos não divergem muito dos símbolos tradicionais. Entretanto, ocorre uma significativa mudança do que estamos acostumados a fazer para alguns dispositivos, incluindo contadores e outros dispositivos mais complexos. Embora continuemos a usar principalmente os símbolos mais tradicionais e familiares nesse livro, apresentamos uma abordagem resumida dos símbolos lógicos com notação de dependência. Um CI contador específico é usado como exemplo.

Ao final do estudo desta seção você deverá ser capaz de:

■ Interpretar símbolos lógicos que incluem notação de dependência ■ Identificar o bloco comum e os elementos individuais no símbolo de um contador ■ Interpretar os símbolos qualificativos ■ Discutir dependência de controle ■ Discutir dependência de modo ■ Discutir dependência AND

A notação de dependência é fundamental para o padrão ANSI/IEEE. A notação de dependência é usada em conjunto com os símbolos lógicos para especificar as relações de entradas e saídas de forma que a operação lógica de um determinado dispositivo possa ser determinada inteiramente a partir do seu símbolo lógico sem um conhecimento prévio de detalhes da sua estrutura interna e sem um diagrama lógico detalhado para referência. Essa abordagem de um símbolo lógico específico com notação de dependência tem o objetivo de ajudá-lo na interpretação de outros símbolos que você pode encontrar depois.

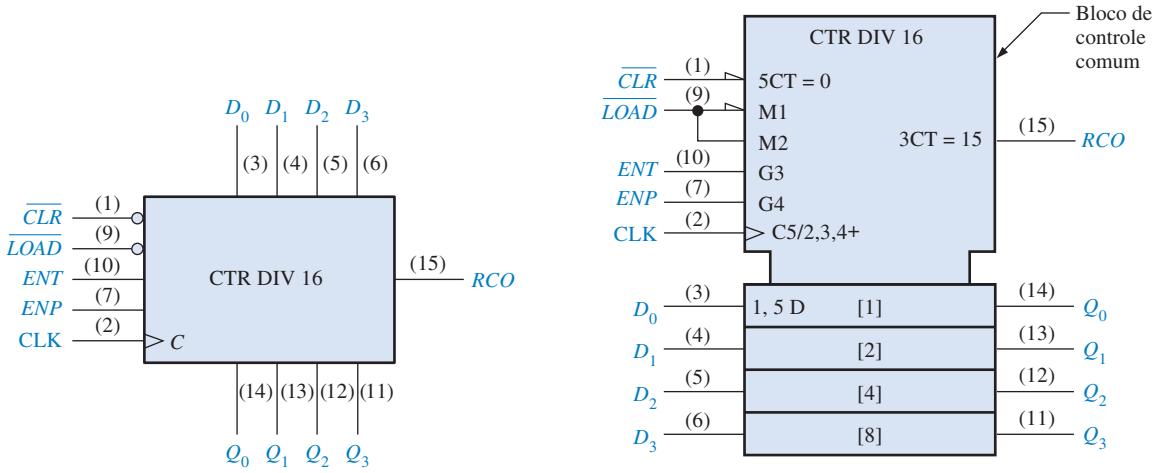
O CI contador binário síncrono de 4 bits 74HC163 é usado como ilustração. Para comparação, a Figura 8–58 mostra um símbolo de um bloco tradicional e o símbolo ANSI/IEEE com notação de dependência. Em seguida temos descrições básicas do símbolo e da notação de dependência.

Bloco de Controle Comum O bloco superior, com recortes nos cantos na Figura 8–85(b), tem entradas e uma saída que são consideradas comuns a todos os elementos no dispositivo e não unicamente para qualquer um desses elementos.

Elementos Individuais O bloco inferior na Figura 8–58(b), o qual está dividido em quatro seções limitadas, representa os quatro elementos de armazenamento (flip-flops D) no contador, com entradas D_0, D_1, D_2 e D_3 e saídas Q_0, Q_1, Q_2 e Q_3.

Símbolos Qualificativos A denominação "CTR DIV 16" na Figura 8–58(b) identifica o dispositivo como um contador (CTR) com dezesseis estados (DIV 16).

Dependência de Controle (C) Como mostra a Figura 8–58(b), a letra *C* indica dependência de controle. Entradas de controle geralmente habilitam ou desabilitam as entradas de dados (*D*, *J*, *K*, *S* e *R*) de um elemento de armazenamento. A entrada *C* é geralmente a entrada de clock. Nesse caso o dígito 5 após o *C* (C5/2,3,4+) indica que a entrada rotulada com um prefixo 5 é dependente



▲ **FIGURA 8–58**

O CI contador síncrono de 4 bits 74HC163.

do clock (sincronizada com o clock). Por exemplo, 5CT = 0 na entrada $\overline{CLR}$ indica que a função clear é dependente do clock; ou seja, é um clear sincronizado. Quando a entrada $\overline{CLR}$ for nível BAIXO (0) o contador é resetado para zero ($CT = 0$) na borda de disparo do pulso de clock. Além disso, o rótulo 5 D na entrada do elemento de armazenamento [1] indica que o armazenamento de dados é dependente do clock (sincronizado com ele). Todos os rótulos no elemento de armazenamento [1] se aplicam aos elementos [2], [4] e [8] abaixo dele desde que eles não sejam rotulados de forma diferente.

Dependência de Modo (M) Como mostra a Figura 8–58(b), a letra *M* indica dependência de modo. Esse rótulo é usado para indicar como as funções de várias entradas ou saídas dependem do modo no qual o dispositivo opera. Nesse caso o dispositivo tem dois modos de operação. Quando a entrada $\overline{CLR}$ for nível BAIXO (0), conforme indicado pelo triângulo na entrada, o contador está no modo presete (M1) no qual a entrada de dados (D_0, D_1, D_2 e D_3) são carregadas sincronamente nos quatro flip-flops. O dígito 1 que segue a letra M e M1 e o 1 no rótulo 1, 5 D mostra uma relação de dependência e indica que os dados de entrada são armazenados apenas quando o dispositivo está no modo presete (M1), no qual $\overline{LOAD} = 0$. Quando a entrada $\overline{LOAD}$ for nível ALTO (1), o contador avança através da sua seqüência binária normal, conforme indicado por M2 e o 2 em C5/2,3,4+.

Dependência AND (G) Como mostra a Figura 8–58(b), a letra *G* indica dependência AND, mostrando que uma entrada designada pela letra *G* seguida de um dígito passa por uma função AND com qualquer outra entrada ou saída que tem o mesmo dígito como um prefixo em seu rótulo. Nesse exemplo em particular, G3 na entrada *ENT* e 3CT = 15 na saída *RCO* estão relacionados, conforme indicado pelo 3, e essa relação é uma dependência AND, indicada pela letra *G*. Isso nos diz que *ENT* tem que ser nível ALTO (sem triângulo na entrada) *e* a contagem tem que ser quinze ($CT = 15$) para a saída *RCO* ser nível ALTO.

Além disso, os dígitos 2, 3 e 4 no rótulo C5/2,3,4+ indicam que o contador avança através de seus estados quando $\overline{LOAD} = 1$, conforme indicado pelos rótulos de dependência AND G3 e G4. O sinal + indica que o contador avança uma contagem quando essa condição existe.

SEÇÃO 8–8 REVISÃO

1. Em notação de dependência, o que significam as letras *C*, *M* e *G*?
2. O armazenamento de dados é indicado por qual letra?

8-9 ANÁLISE DE DEFEITO

A análise de defeito em circuitos com contadores pode ser simples ou bastante complexa, dependendo do tipo de contador e do tipo de defeito. Essa seção proporciona alguma compreensão em como abordar a análise de defeito de circuitos seqüenciais.

Ao final do estudo desta seção você deverá ser capaz de:

■ Detectar um contador com defeito ■ Isolar defeitos em contadores de módulo máximo conectados em cascata ■ Isolar defeitos em contadores com seqüências truncadas conectados em cascata ■ Determinar defeitos em contadores implementados com flip-flops individuais

Contadores

Para um contador com uma seqüência direta que não é controlado por lógica externa, quase que a única coisa a verificar (além de V_{CC} e GND) é a possibilidade de entradas e saídas abertas ou em curto-circuito. Um CI contador quase nunca altera a sua seqüência de estados devido a um defeito interno, assim precisamos apenas verificar o efeito dos pulsos nas saídas Q para detectar a existência de um ponto aberto ou em curto-circuito. A ausência de pulsos em uma das saídas Q indica uma linha interna aberta ou em curto-circuito, a qual pode ser interna ou externa ao CI. A ausência de pulsos em todas as saídas Q indica que a entrada de clock está com defeito ou a entrada de clear está fixa no seu estado ativo.

Para verificar a entrada de clear, aplique um nível ativo constante enquanto o contador recebe clock. Observaremos um nível BAIXO em cada uma das saídas Q se o contador estiver funcionando corretamente.

Uma característica de carga paralela síncrona num contador pode ser verificada ativando a entrada de carga paralela e testando cada estado como a seguir: Aplique nível BAIXO nas entradas paralelas, ative a entrada de clock uma vez e verifique se todas as saídas Q estão em nível BAIXO. Em seguida, aplique nível ALTO nas entradas de dados paralelas, ative a entrada de clock uma vez e verifique se todas as saídas Q estão em nível ALTO.

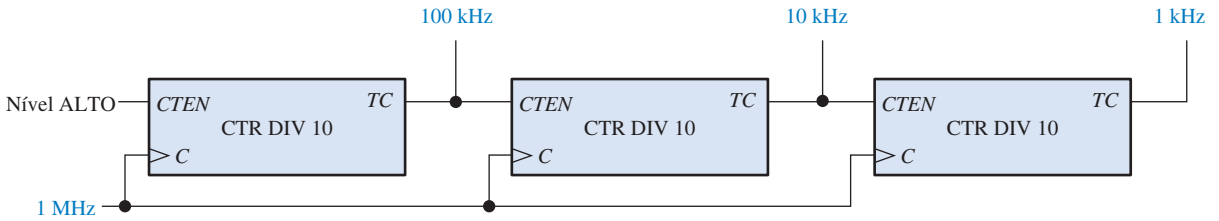
Contadores em Cascata com Módulo Máximo

Um defeito em um dos contadores numa cadeia de contadores em cascata pode afetar todos os contadores que vêm a seguir. Por exemplo, se uma entrada de habilitação de contagem abre, essa entrada funciona efetivamente como nível ALTO (para TTL) e o contador estará sempre habilitado. Esse tipo de defeito em um dos contadores fará com que ele opere a uma taxa maior que a normal. Isso está ilustrado na Figura 8–59 para um contador/divisor por 1000 conectado em cascata onde uma entrada de habilitação (*CTEN*) aberta funciona como uma entrada TTL em nível ALTO habilitando continuamente o segundo contador. Outros defeitos que podem afetar estágios de contadores de "fluxo decrescente" são entradas de clock ou saídas de contagem final abertas ou em curto-circuito. Em algumas dessas situações, podem ser observados pulsos, mas podendo estar numa freqüência errada. Devem ser feitas medidas de freqüência exata ou taxa de freqüência.

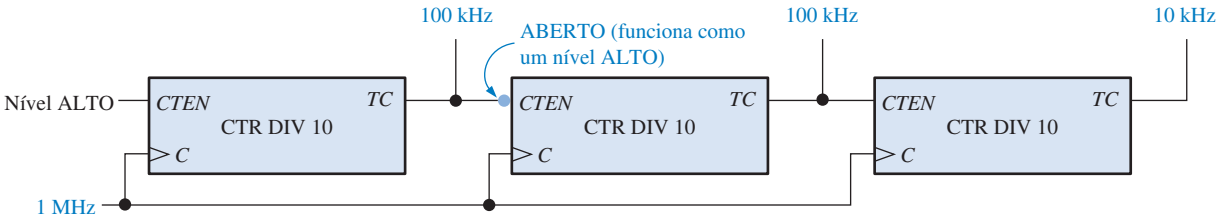
Contadores em Cascata com Seqüências Truncadas

A seqüência de contagem de um contador em cascata com seqüência truncada, tal como a que é mostrada na Figura 8–60, pode ser afetada por outros tipos de defeitos além daqueles mencionados para contadores em cascata de módulo máximo. Por exemplo, um defeito em uma das entradas de dados em paralelo, a entrada $\overline{LOAD}$, ou o inversor pode alterar a contagem presetada e assim alterar o módulo do contador.

Por exemplo, suponha que a entrada D_3 do contador mais significativo visto na Figura 8–60 esteja aberta e funcione como um nível ALTO. Em vez da contagem 6_{16} (0110) ser presetada no contador, E_{16} (1110) é presetado. Assim, em vez de começar com $63C0_{16}$ (25.536_{10}) cada vez que o contador reciclar, a seqüência começa com $E3C0_{16}$ (58.304_{10}). Isso altera o módulo do contador de 40.000 para 65.536 – 58.304 = 7273.



(a) Operação normal



(b) Entrada de habilitação de contagem (CTEN) do segundo contador aberta

▲ **FIGURA 8–59**

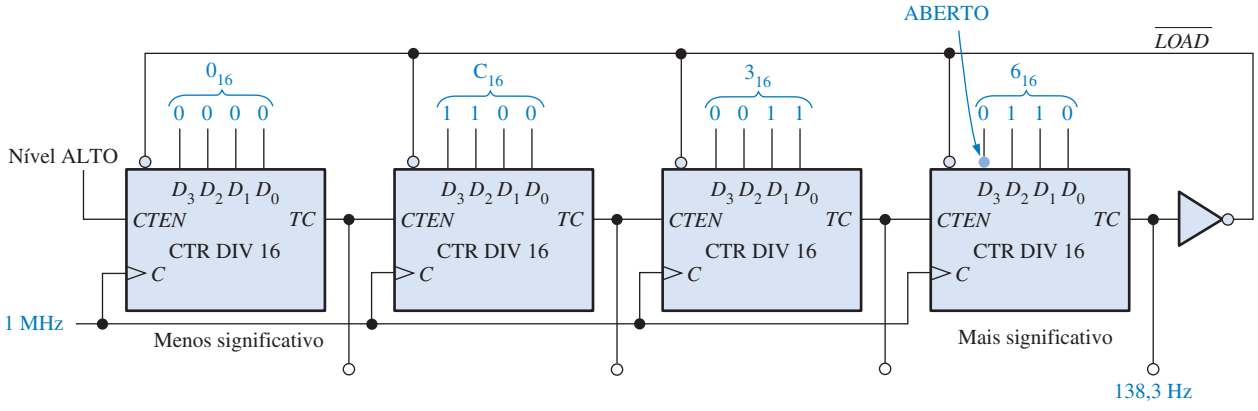
Exemplo de um defeito que afeta nós contadores seguintes num arranjo em cascata.

Para verificar esse contador, aplique uma freqüência de clock conhecida, por exemplo 1 MHz, e meça a freqüência na saída da última contagem final. Se o contador estiver operando corretamente, a freqüência de saída será

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\text{ MHz}}{40.000} = 25\text{ Hz}$$

Nesse caso, o defeito específico descrito no parágrafo anterior faz com que a freqüência de saída seja

$$f_{out} = \frac{f_{in}}{\text{módulo}} = \frac{1\text{ MHz}}{7232} = 138{,}3$$

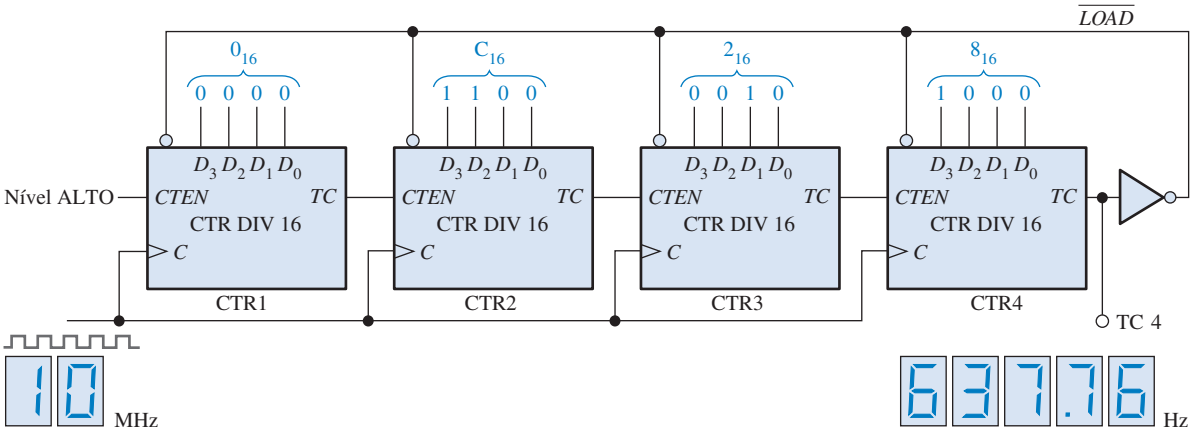


▲ **FIGURA 8–60**

Exemplo de um defeito num contador em cascata com seqüência truncada.

EXEMPLO 8–10

São realizadas medidas de freqüência no contador truncado mostrado na Figura 8–61 conforme indicado. Determine se o contador está funcionando corretamente e, em caso negativo, determine o defeito.



▲ FIGURA 8–61

Solução Verifique se a freqüência medida em TC 4 está correta. Em caso afirmativo, o contador está funcionando corretamente.

$$\begin{aligned}\text{módulo truncado} &= \text{módulo completo} - \text{contagem presetada}\\ &= 16^4 - 82C0_{16}\\ &= 65.536 - 33.472 = 32.064\end{aligned}$$

A freqüência correta em TC 4 é

$$f_4 = \frac{10\text{ MHz}}{32.064} = 311{,}88\text{ Hz}$$

Hum! Temos um problema. A freqüência de 637,76 Hz medida não está de acordo com a freqüência calculada de 318,88 Hz.

Para encontrar o contador com defeito, determine o módulo truncado real com a seguir:

$$\text{módulo} = \frac{f_{\text{in}}}{f_{\text{out}}} = \frac{10\text{ MHz}}{637{,}76\text{ Hz}} = 15.680$$

Como o módulo truncado deveria ser 32.064, é mais provável que o contador esteja sendo presetado em uma contagem errada quando recicla. A contagem presetada real do contador é determinada como a seguir:

$$\begin{aligned}\text{módulo truncado} &= \text{módulo completo} - \text{contagem presetada}\\ \text{contagem presetada} &= \text{módulo completo} - \text{módulo truncado}\\ &= 65.536 - 15.5680\\ &= 49.856\\ &= C2C0_{16}\end{aligned}$$

Isso mostra que o contador está sendo presetado com $C2C0_{16}$, em vez de $82C0_{16}$, cada vez que recicla.

Os contadores 1, 2 e 3 estão sendo presetados corretamente, porém o contador 4 não. Como $C_{16} = 1100_2$, a entrada D_2 para o contador 4 está em nível ALTO quando deveria ser nível BAIXO. Isso é causado mais provavelmente por uma **entrada aberta**. Verifique a existência de um circuito aberto externo provocado por uma solda fria, um condutor partido ou um pino dobrado no CI. Caso nada seja encontrado, substitua o CI e o contador deve funcionar corretamente.

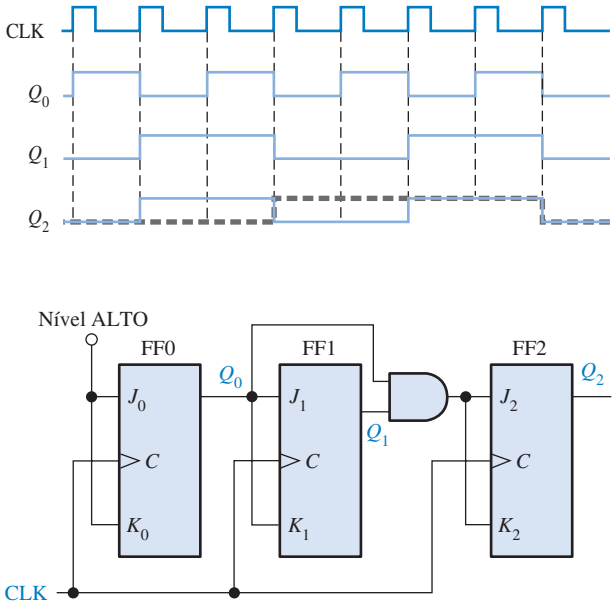
Problema relacionado Determine qual seria a freqüência de saída em TC 4 se a entrada D_3 do contador 3 estiver aberta.

Contadores Implementados com Flip-Flops Individuais

Contadores implementados com flip-flops individuais e CIs de portas lógicas são às vezes mais difíceis de se realizar uma análise de defeito porque existem muito mais entradas e saídas com conexões externas do que num CI contador. A seqüência de um contador pode ser alterada por uma única entrada ou saída aberta ou em curto-circuito, conforme ilustra o Exemplo 8–11.

EXEMPLO 8–11

Suponha que observamos as seguintes formas de onda indicadas para o contador na Figura 8–62. Determine se existe um problema com o contador.



▶ FIGURA 8–62

Solução A forma de onda Q_2 está errada. A forma de onda correta é mostrada com uma linha tracejada. Podemos ver que a forma de onda de Q_2 se parece exatamente com a forma de onda de Q_1, assim o que quer que esteja fazendo o FF1 comutar parece também estar controlando FF2.

Verificando as entradas *J* e *K* do FF2, encontramos uma forma de onda que se parece com Q_0. Esse resultado indica que Q_0 está de alguma forma passando através da porta AND. A única forma disso acontecer é se a entrada Q_1 da porta AND estiver sempre em nível ALTO. Entretanto, vimos que Q_1 tem uma forma de onda correta. Essa observação nos leva a concluir que a entrada inferior da porta AND tem que estar aberta internamente funcionando como um nível ALTO. Substitua a porta AND e resete o circuito.

Problema relacionado Descreva a saída Q_2 do contador visto na Figura 8–62 se a saída Q_1 do FF1 estiver aberta.

SEÇÃO 8–9 REVISÃO

1. Quais são os defeitos que podem fazer com que o contador visto na Figura 8–59 não tenha pulsos em qualquer uma das saídas *TC*?
2. O que acontece se o inversor no circuito mostrado na Figura 8–61 tiver a saída aberta?

Os problemas de análise de defeito abordados no CD-ROM estão disponíveis na Seção "Prática de Análise de Defeito Usando o Multisim" nos problemas no final do capítulo.

DICA PRÁTICA

Para observar a relação de tempo entre dois sinais digitais com um osciloscópio de duplo traço, a forma adequada de ajustar o trigger do osciloscópio é usando como referência o sinal mais lento dos dois. A razão para isso é que o sinal mais lento tem menos pontos possíveis de trigger que o sinal mais rápido e não haverá ambigüidade para iniciar a varredura. O trigger no modo vertical usa uma composição dos dois canais e nunca deve ser usado para determinar uma informação de tempo absoluto. Como os sinais de clock são geralmente os sinais mais rápidos num sistema digital, eles não devem ser usados para trigger.

APLICAÇÕES EM SISTEMAS DIGITAIS

O sistema de controle de semáforo que foi apresentado no Capítulo 6 e abordado também no Capítulo 7 é finalizado aqui. No Capítulo 6 foi desenvolvida a lógica combinacional.

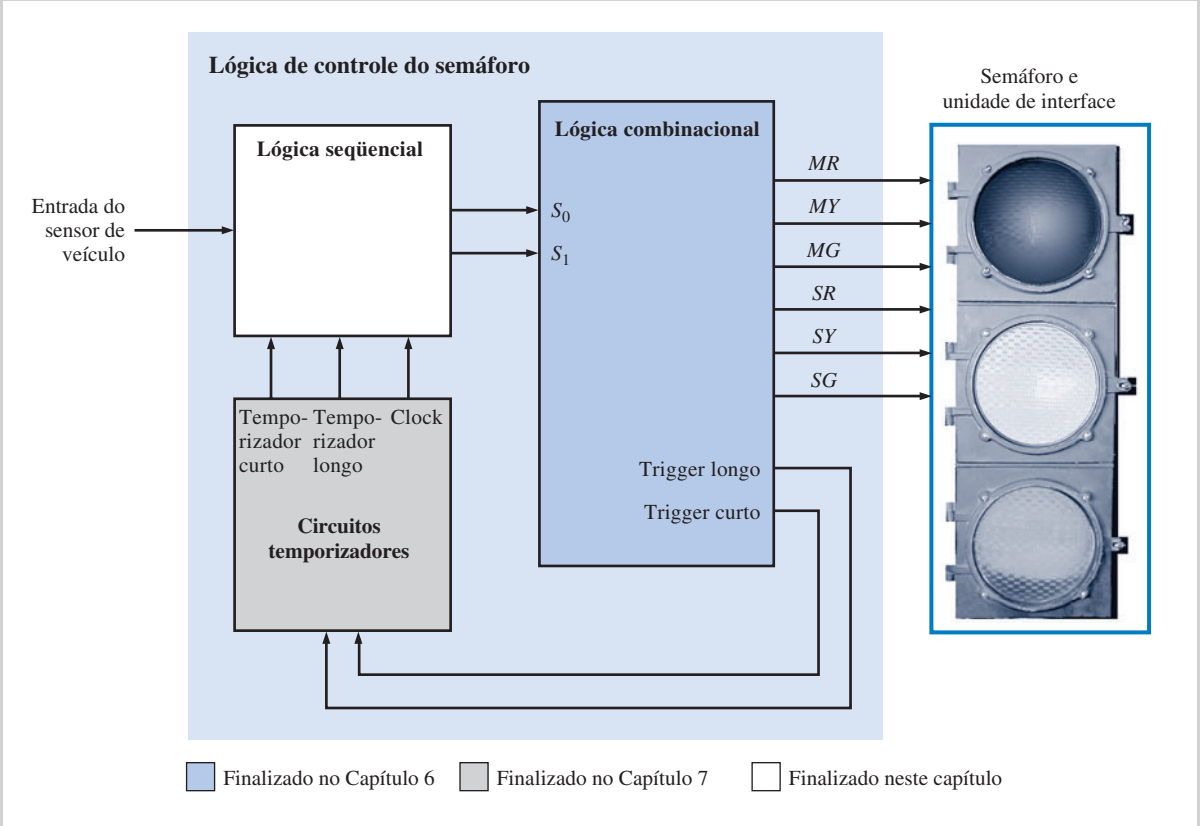
No Capítulo 7 foram desenvolvidos os circuitos de temporização.

Neste capítulo desenvolvemos a lógica seqüencial e todos os blocos que são conectados para completar o sistema de controle de semáforo. O diagrama em bloco geral do sistema é mostrado novamente na Figura 8–63.

Requisitos da Lógica Seqüencial

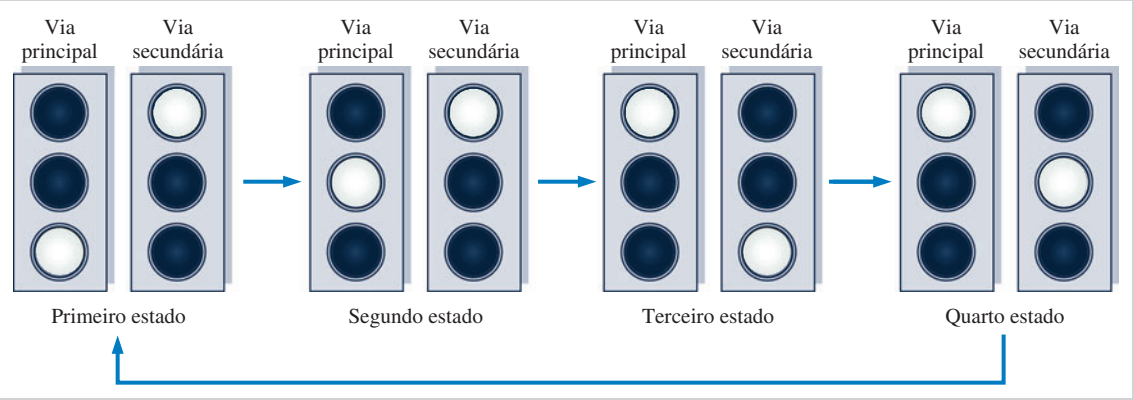
A lógica seqüencial controla a seqüência de estados do semáforo baseado nas entradas dos circuitos de temporização e do sensor de veículo. A lógica seqüencial produz uma seqüência em código Gray de 2 bits para os quatro estados do sistema os quais são indicados na Figura 8–64.

Diagrama em Bloco A lógica seqüencial consiste de um contador de código Gray de 2 bits e uma lógica de entrada associada, conforme mostra a Figura 8–65.



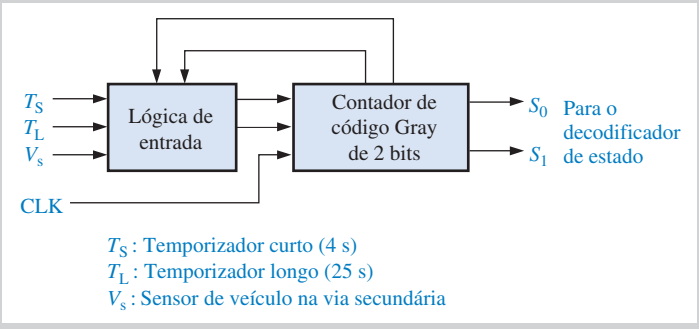
▲ **FIGURA 8–63**

Diagrama em bloco do sistema de controle de semáforo.



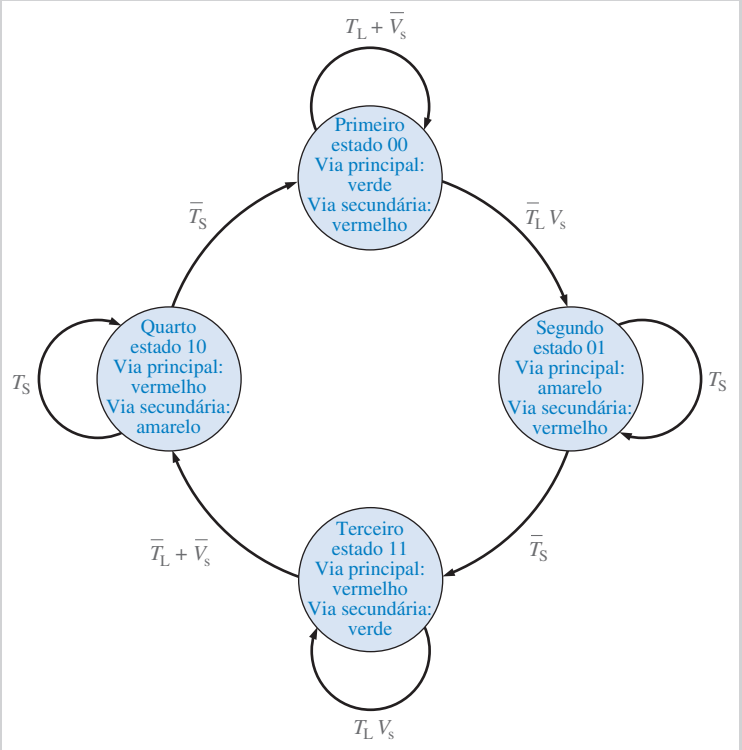
▲ **FIGURA 8–64**

Seqüência de estados do semáforo.



▶ **FIGURA 8–65**

Diagrama em bloco da lógica seqüencial.



▶ **FIGURA 8–66**

Diagrama de estados do sistema de controle do semáforo.

O contador produz uma seqüência de quatro estados. As transições de um estado para outro são determinadas pelos temporizadores de 4 s e 25 s e pela entrada do sensor de veículo. O clock para o contador é o sinal de 10 kHz produzido pelo oscilador nos circuitos de temporização.

Diagrama de Estados O diagrama de estados para o sistema de controle de semáforo foi apresentado no Capítulo 6 e é mostrado novamente na Figura 8–66. Baseado nesse diagrama de estados a operação da lógica seqüencial é descrita como a seguir:

Primeiro estado: O código Gray para esse estado é 00. A luz da via principal é verde e a da via secundária é vermelha. O sistema permanece nesse estado por pelo menos 25 s quando o temporizador longo está ligado ou enquanto não houver veículo na via secundária. Essa situação é expressa como $T_L + \overline{V}_s$. O sistema vai para o próximo estado quando o temporizador longo estiver desligado e houver um veículo na via secundária. Essa situação é expressa como $(\overline{T}_L V_s)$.

Segundo estado: O código Gray para esse estado é 01. A luz da via principal é amarela e a da via secundária é vermelha. O sistema permanece nesse estado por 4 s quando o temporizador curto está ligado (T_s) e vai para o próximo estado quando o temporizador curto se *desligar* $(\overline{T}_S)$.

Terceiro estado: O código Gray para esse estado é 11. A luz da via principal é vermelha e a da via secundária é verde. O sistema permanece nesse estado quando o temporizador longo estiver ligado e houver um veículo na via secundária. Esse estado é expresso como $T_L V_S$. O sistema vai para o próximo estado quando o temporizador longo se desligar ou quando não houver veículo na via secundária. Esse estado é expresso como $T_L + V_S$.

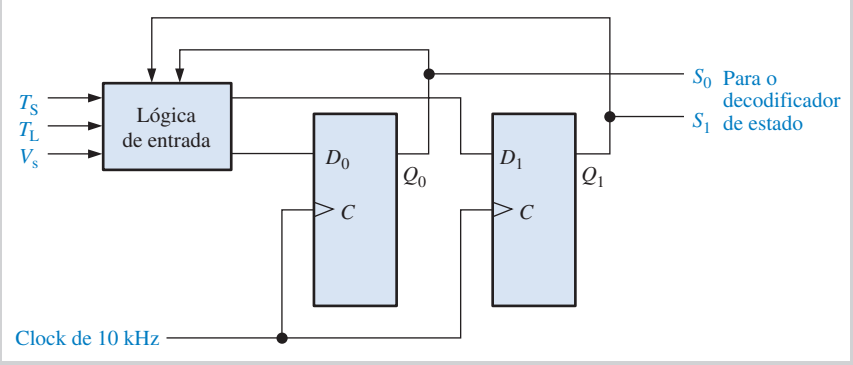
Quarto estado: O código Gray para esse estado é 10. A luz da via principal é vermelha e a da via secundária é amarela. O sistema permanece nesse estado por 4 s quando o temporizador curto estiver ligado (T_s) e retorna para o primeiro estado quando o temporizador curto se *desligar* $(\overline{T}_S)$.

Implementação da Lógica Seqüencial O diagrama visto na Figura 8–67, mostra que dois flip-flops D são usados para implementar o contador Gray. As saídas da lógica de entrada fornecem as entradas *D* dos flip-flops e o contador recebe clock a partir do oscilador. A lógica de entrada tem cinco variáveis de entrada: Q_0, Q_1, T_L, T_S e V_s.

A tabela de transição do flip-flop D é mostrada na Tabela 8–13. A partir do diagrama de estados uma tabela do próximo estado pode ser desenvolvida como mostra a Tabela 8–14. As condições de entrada para T_L, T_S e V_s para cada combinação de estado atual/próximo estado são relacionadas na tabela.

A partir da Tabela 8–13 e Tabela 8–14, as condições lógicas necessárias para cada flip-flop ir para o estado 1 podem ser determinadas. Por exemplo, Q_0 vai de 0 para 1 quando o estado atual for 00 e a condição de entrada for $\overline{T}_L + \overline{V}_S$, conforme indicado na segunda linha da Tabela 8–13. D_0 tem que ser 1 para fazer com que Q_0 passe para o nível 1 ou permaneça em 1 no próximo pulso de clock. Para D_0 ser nível 1, uma expressão lógica pode ser escrita a partir da Tabela 8–14:

$$\begin{aligned} D_0 &= \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 T_S \\ &\quad + \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 T_L V_s \\ &= \overline{Q}_1\overline{Q}_0\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_1 Q_0 T_L V_s \end{aligned}$$



▶ **FIGURA 8–67**
Diagrama lógico seqüencial.

▼ **TABELA 8–13**
Tabela de transição do flip-flop D

TRANSIÇÕES DE SAÍDA Q_N		Q_{N+1}	ENTRADA DO FLIP-FLOP *D*
0	⟶	0	0
0	⟶	1	1
1	⟶	0	0
1	⟶	1	1

Podemos usar um mapa de Karnaugh para reduzir ainda mais a expressão para D_0 para

$$D_0 = \overline{Q}_1\overline{T}_L V_s + \overline{Q}_1 Q_0 + Q_0 T_L V_s$$

Além disso, a partir da Tabela 8–14 a expressão para D_1 pode ser desenvolvida.

$$\begin{aligned} D_1 &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 T_L V_s \\ &\quad + Q_1 Q_0 \overline{T}_L + Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (T_L V_s + \overline{T}_L) \\ &\quad + Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (V_s + \overline{T}_L) \\ &\quad + Q_1 Q_0 \overline{V}_s + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 (V_s + \overline{T}_L + \overline{V}_s) \\ &\quad + Q_1 \overline{Q}_0 T_S \\ &= \overline{Q}_1 Q_0 \overline{T}_S + Q_1 Q_0 + Q_1 \overline{Q}_0 T_S \end{aligned}$$

Podemos usar um mapa de Karnaugh para reduzir ainda mais a expressão D_1 para

$$D_1 = Q_0\overline{T}_S + Q_1 T_S$$

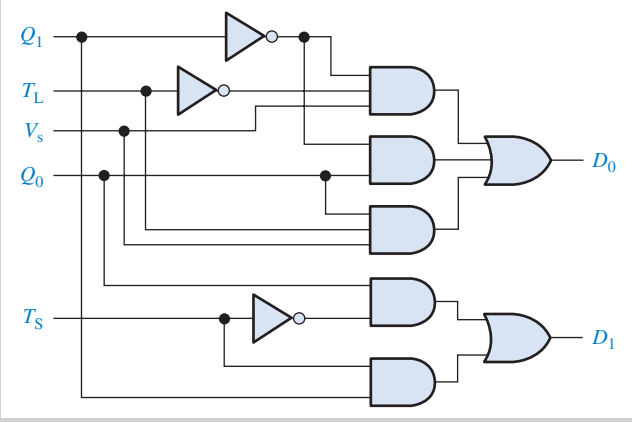
D_0 e D_1 são implementados conforme mostra a Figura 8–68.

Combinando a lógica de entrada com o contador de 2 bits, o diagrama da lógica seqüencial completa é mostrado na Figura 8–69.

▼ TABELA 8–14

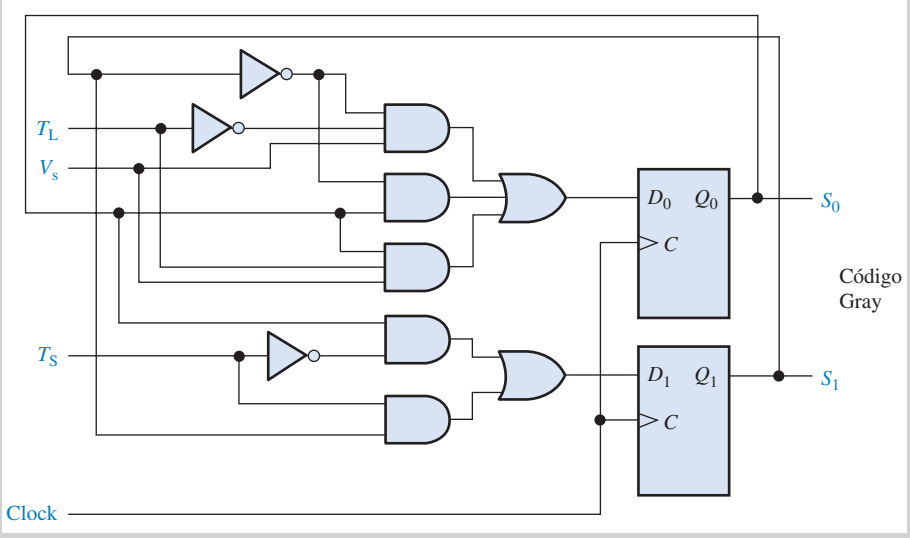
Tabela do próximo estado para as transições da lógica seqüencial

ESTADO ATUAL Q_1	ESTADO ATUAL Q_0	PRÓXIMO ESTADO Q_1	PRÓXIMO ESTADO Q_0	CONDIÇÕES DE ENTRADA	D_1	D_0
0	0	0	0	$T_L + \overline{V}_s$	0	0
0	0	0	1	$\overline{T}_L V_s$	0	1
0	1	0	1	T_S	0	1
0	1	1	1	$\overline{T}_S$	1	1
1	1	1	1	$T_L V_s$	1	1
1	1	1	0	$\overline{T}_L + \overline{V}_s$	1	0
1	0	1	0	T_S	1	0
1	0	0	0	$\overline{T}_S$	0	0



▲ FIGURA 8–68

Lógica de entrada para o contador de código Gray de 2 bits.



▶ FIGURA 8–69

A lógica seqüencial.

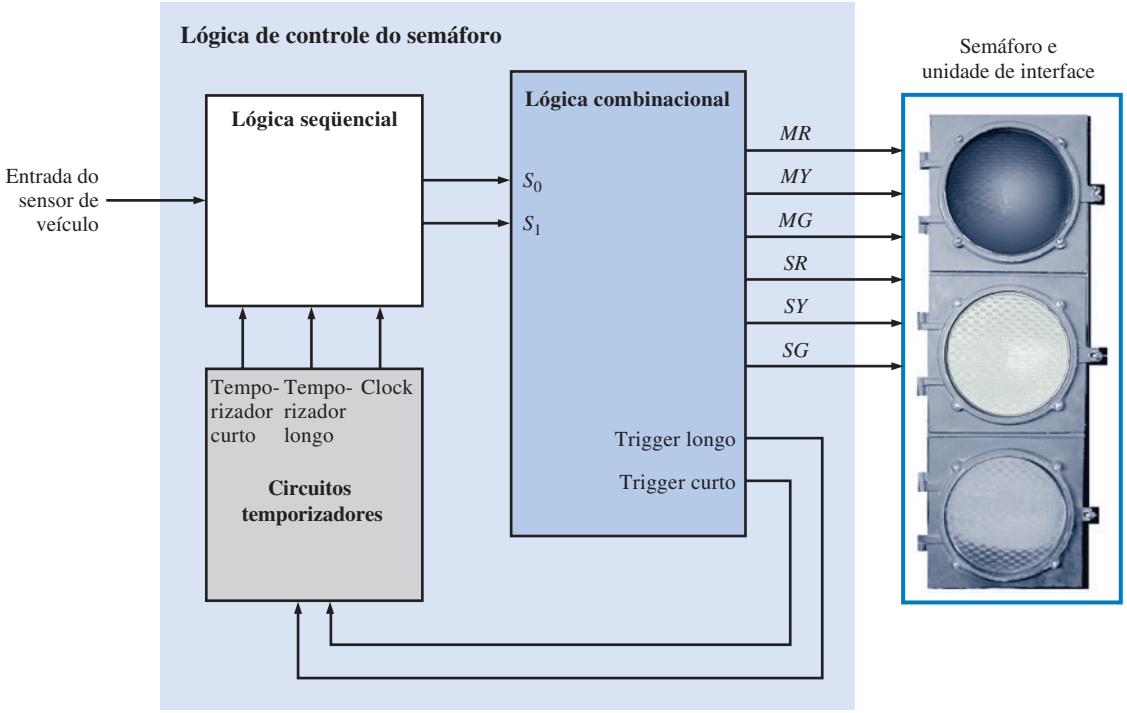
O Sistema Completo de Controle de Semáforo

Agora que temos todos os três blocos (lógica combinacional, circuitos de temporização e lógica seqüencial), os combinamos de forma a ter o sistema completo, conforme mostra o diagrama em bloco na Figura 8–70.

Os Circuitos de Interface Os circuitos de interface são necessários porque o circuito lógico não pode acionar diretamente as luzes devido aos requisitos de tensão e corrente. Existem algumas formas possíveis de desenvolver essa interface, porém o Apêndice B apresenta dois projetos possíveis.

Atribuições do Sistema

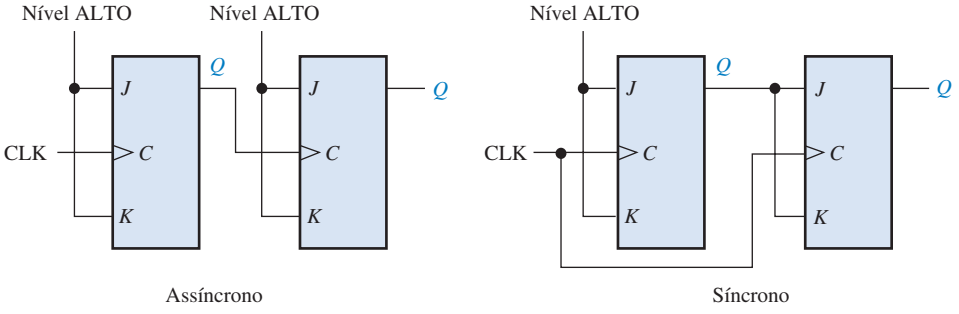
- *Atividade 1* Use um mapa de Karnaugh para confirmar que a expressão simplificada para D_0 está correta.
- *Atividade 2* Use um mapa de Karnaugh para confirmar que a expressão simplificada para D_1 está correta.



▲ **FIGURA 8–70**
Diagrama em bloco do sistema completo de controle de semáforo.

RESUMO

- Os contadores síncronos e assíncronos diferem entre si apenas na forma em que eles recebem o sinal de clock, conforme mostra a Figura 8–71. Os contadores síncronos podem operar com taxas de clock maiores que os contadores assíncronos.



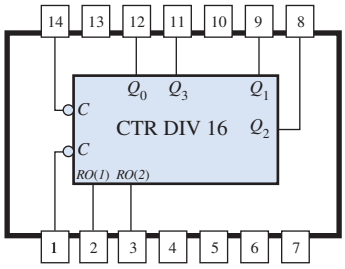
▶ **FIGURA 8–71**
Comparações entre contadores assíncronos e síncronos.

- Os diagramas de conexões para os CIs contadores apresentados neste capítulo são mostrados na Figura 8–72.
- O módulo máximo de um contador é o número máximo de estados possíveis e é uma função do número de estágios (flip-flops). Portanto,

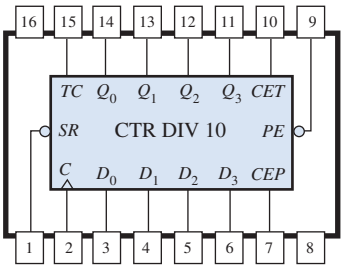
$$\text{Módulo máximo} = 2^n$$

onde n é o número de estágios no contador. O módulo de um contador é o número real de estados em sua seqüência e pode ser igual ou menor que o módulo máximo.

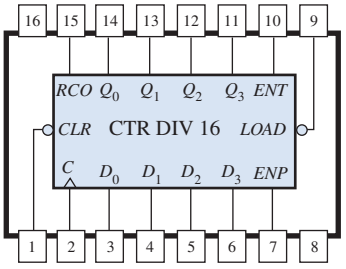
- O módulo total de contadores em cascata é igual ao produto dos módulos dos contadores individuais.



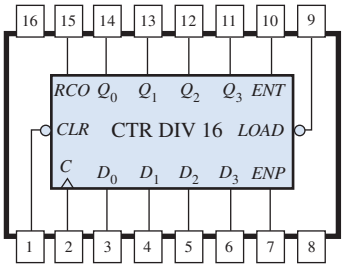
CI contador binário assíncrono de 4 bits 74LS93



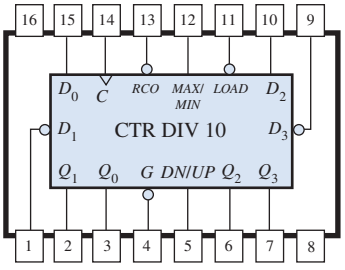
CI contador de década BCD síncrono com clear assíncrono 74F162



CI contador binário síncrono de 4 bits com clear assíncrono 74HC161



CI contador binário síncrono de 4 bits com clear síncrono 74HC163



CI contador de década crescente/decrescente síncrono 74HC190 (*G* é o habilitador de contagem)

▲ FIGURA 8–72

Observe que os rótulos (nomes das entradas e saídas) são consistentes com o texto, mas podem diferir de um formulário de um fabricante em particular. Os dispositivos mostrados são funcionalmente os mesmos, com compatibilidade de pinos, que os tipos comercializados em outras famílias TTL e CMOS.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Assíncrono Que não ocorre no mesmo instante.

Conexão em cascata Quando vários contadores são conectados a partir da saída de contagem final de um contador para a entrada de habilitação (enable) do próximo contador.

Contagem final O estado final na seqüência de um contador.

Década Caracterizado por dez estados ou valores.

Diagrama de estados Uma ilustração gráfica de uma seqüência de estados ou valores.

Máquina de estados Um sistema lógico que exibe uma seqüência de estados condicionados pela lógica interna e as entradas externas; qualquer circuito seqüencial que exibe uma seqüência específica de estados.

Módulo O número de estados únicos através dos quais o contador passa.

Reciclagem Sofrer transição (como em um contador) de um estado final de volta para o estado inicial.

Síncrono Que ocorre no mesmo instante.

AUTOTESTE

As respostas estão no final do capítulo.

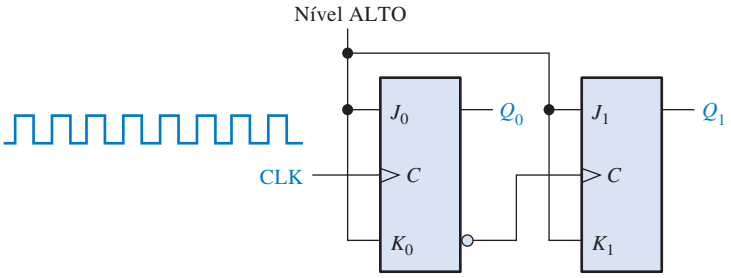
1. Contadores assíncronos são conhecidos como
   (a) contadores ondulantes **(b)** contadores de clock múltiplo
   (c) contadores de década **(d)** contadores de módulo
2. Um contador assíncrono difere de um contador síncrono
   (a) no número de estados de sua seqüência
   (b) na forma de receber pulsos de clock
   (c) no tipo de flip-flop usado
   (d) no valor do módulo
3. O módulo de um contador é
   (a) o número de flip-flops
   (b) o número atual de estados em sua seqüência
   (c) o número de vezes que ele recicla em um segundo
   (d) o número máximo possível de estados
4. Um contador binário de 3 bits tem um módulo máximo de
   (a) 3 **(b)** 6 **(c)** 8 **(d)** 16
5. Um contador binário de 4 bits tem um módulo máximo de
   (a) 16 **(b)** 32 **(c)** 8 **(d)** 4
6. Um contador de módulo 12 tem que ter
   (a) 12 flip-flops **(b)** 3 flip-flops
   (c) 4 flip-flops **(d)** pulsos de clock síncronos
7. Qual das seguintes alternativas é um exemplo de contador com um módulo truncado?
   (a) Módulo 8 **(b)** Módulo 14
   (c) Módulo 16 **(d)** Módulo 32
8. Um contador ondulante de 4 bits consiste em flip-flops em que cada um tem um atraso de propagação a partir do clock para a saída Q de 12 ns. Para esse contador reciclar de 1111 para 0000, ele gasta um total de
   (a) 12 ns **(b)** 24 ns **(c)** 48 ns **(d)** 36 ns
9. Um contador BCD é um exemplo de um
   (a) contador de módulo completo **(b)** contador de década
   (c) contador de módulo truncado **(d)** as respostas (b) e (c) estão corretas
10. Qual das seguintes alternativas é um estado inválido para um contador BCD 8421?
    (a) 1100 **(b)** 0100 **(c)** 0101 **(d)** 1000
11. Três contadores de módulo 10 conectados em cascata têm um módulo total de
    (a) 30 **(b)** 100 **(c)** 1000 **(d)** 10.000
12. Uma freqüência de clock de 10 MHz é aplicada a um contador em cascata que consiste em um contador de módulo 5, um contador de módulo 8 e dois contadores de módulo 10. A menor freqüência de saída possível é
    (a) 10 KHz **(b)** 2,5 KHz **(c)** 5 KHz **(d)** 25 KHz
13. Um contador crescente/decrescente binário de 4 bits está no estado binário zero. O próximo estado no modo DESCRESCENTE é
    (a) 0001 **(b)** 1111 **(c)** 1000 **(d)** 1110
14. A contagem final de um contador de módulo 13 é
    (a) 0000 **(b)** 1111 **(c)** 1100 **(d)** 1110

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

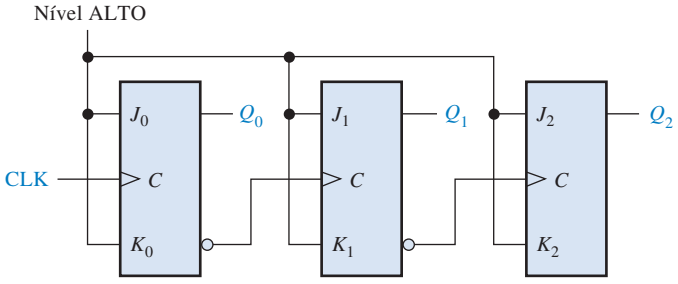
SEÇÃO 8–1 Operação de Contadores Assíncronos

1. Para o contador ondulante mostrado na Figura 8–73, mostre o diagrama de temporização completo para oito pulsos de clock, mostrando as formas de onda do clock, de Q_0 e de Q_1.



▶ **FIGURA 8–73**

2. Para o contador ondulante dado na Figura 8–74, mostre o diagrama de temporização completo para dezesseis pulsos de clock. Mostre as formas de onda do clock, de Q_0, de Q_1 e de Q_2.



▶ **FIGURA 8–74**

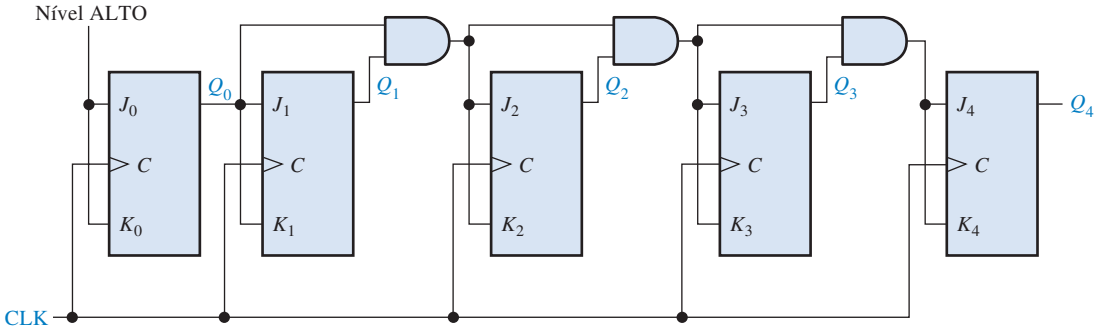
3. No contador do Problema 2, considere que cada flip-flop tem um atraso de propagação entre a borda de subida do clock e a transição na saída Q de 8 ns. Determine o tempo de atraso para o pior caso (tempo mais longo) entre o pulso de clock até que o contador chegue a um dado estado. Especifique o estado, ou estados, para os quais esse atraso de pior caso acontece.

4. Mostre como conectar um CI contador assíncrono de 4 bits 74LS93 para cada um dos seguintes módulos:
(a) 9 **(b)** 11 **(c)** 13 **(d)** 14 **(e)** 15

SEÇÃO 8–2 Operação de Contadores Síncronos

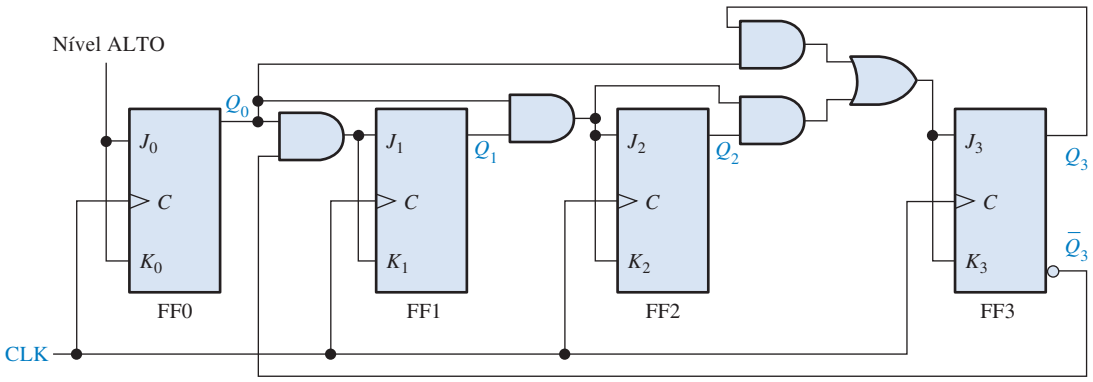
5. Se o contador do Problema 3 fosse síncrono em vez de assíncrono, qual seria o maior atraso de tempo?

6. Mostre o diagrama de temporização completo para o contador binário síncrono de 5 estágios visto na Figura 8–75. Verifique que as formas de onda das saídas Q representam o número binário correto após cada pulso de clock.



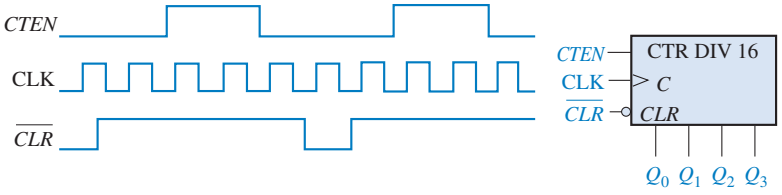
▲ **FIGURA 8–75**

7. Analisando as entradas J e K de cada flip-flop antes de cada pulso de clock, prove que o contador de década visto na Figura 8–76 avança através de uma seqüência BCD. Explique como essas condições em cada caso fazem com que o contador passe para o próximo estado correto.



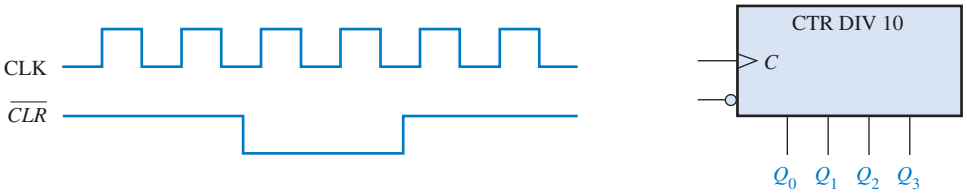
▲ **FIGURA 8–76**

8. As formas de onda vistas na Figura 8–77 são aplicadas nas entradas de habilitação do contador, de clear e de clock conforme indicado. Mostre as formas de onda de saída do contador em relação a essas entradas. A entrada clear é assíncrona.



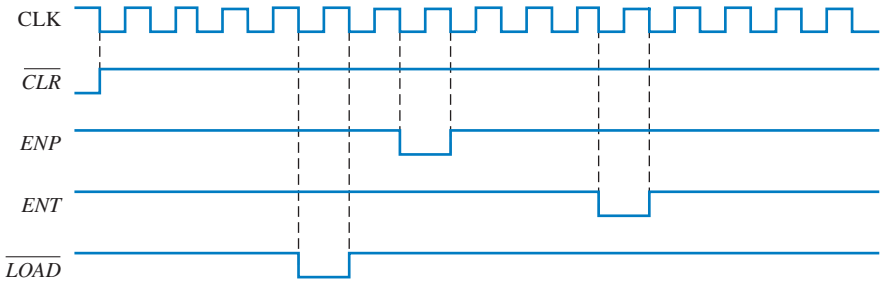
▶ **FIGURA 8–77**

9. Um contador de década BCD é mostrado na Figura 8–78. As formas de onda são aplicadas nas entradas de clear e clock conforme indicado. Determine as formas de onda para cada uma das saídas do contador (Q_0, Q_1, Q_2 e Q_3). O clear é síncrono e o contador está inicialmente no estado binário 1000.



▶ **FIGURA 8–78**

10. As formas de onda vistas na Figura 8–79 são aplicadas ao contador 74HC163. Determine as saídas Q e RCO. As entradas são $D_0 = 1$, $D_1 = 1$, $D_2 = 0$ e $D_3 = 1$.



▶ **FIGURA 8–79**

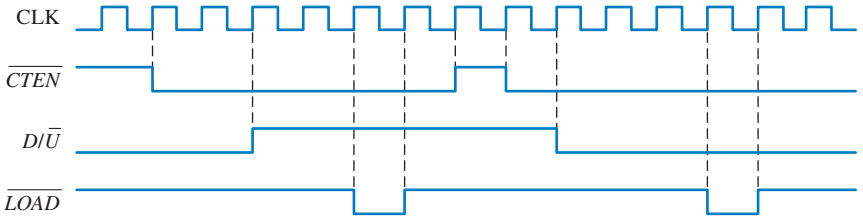
11. As formas de onda vistas na Figura 8–79 são aplicadas no CI contador 74F162. Determine as saídas Q e TC. As entradas são $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ e $D_3 = 1$.

SEÇÃO 8–3 Contadores Síncronos Crescente/Decrescente

12. Mostre um diagrama de temporização completo para um contador crescente/decrescente de 3 bits que percorre a seqüência a seguir. Indique quando o contador está no modo CRESCENTE e quando está no modo DECRESCENTE. Considere disparo na borda positiva.

0, 1, 2, 3, 2, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0

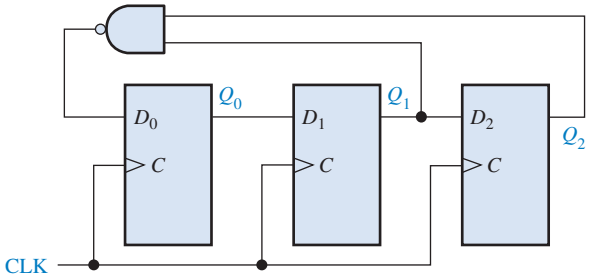
13. Desenvolva as formas de onda da saída Q para um contador crescente/decrescente com as formas de onda de entrada mostradas na Figura 8–80. Um binário 0 está nas entradas de dados. Comece com uma contagem 0000.



▶ FIGURA 8–80

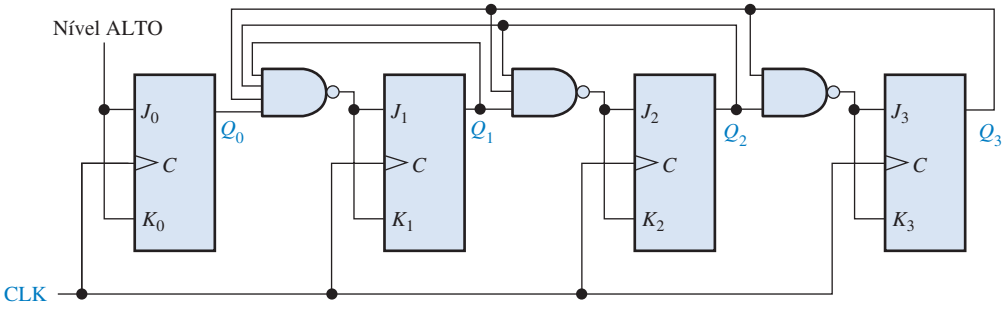
SEÇÃO 8–4 Projeto de Contadores Síncronos

14. Determine a seqüência do contador mostrado na Figura 8–81.



▶ FIGURA 8–81

15. Determine a seqüência do contador visto na Figura 8–82. Comece com o contador resetado.



▲ FIGURA 8–82

16. Projete um contador que gere a seqüência a seguir. Use flip-flops J-K.

00, 10, 01, 11, 00, . . .

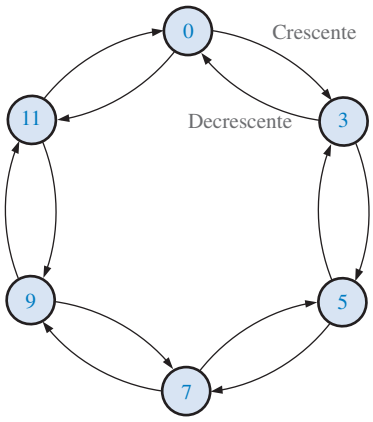
17. Projete um contador que gere a seqüência binária a seguir. Use flip-flops J-K.

1, 4, 3, 5, 7, 6, 2, 1, . . .

18. Projete um contador que gere a seqüência binária a seguir. Use flip-flops J-K.

0, 9, 1, 8, 2, 7, 3, 6, 4, 5, 0, . . .

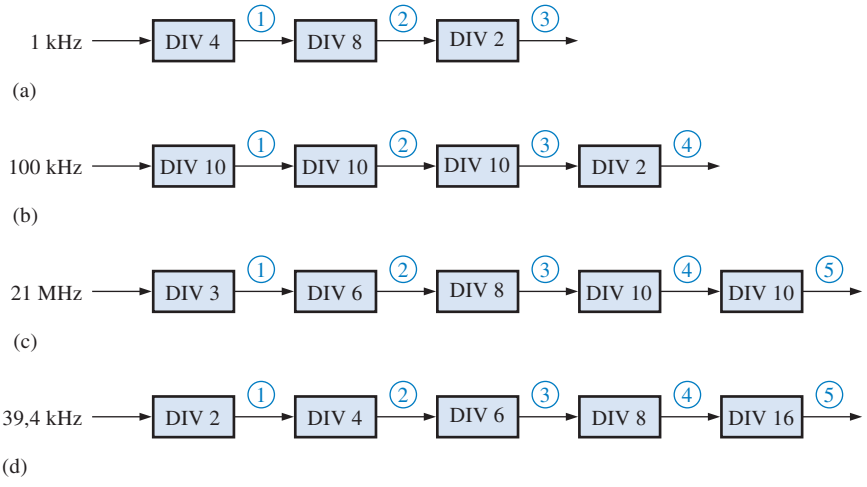
19. Projete um contador binário com a seqüência mostrada no diagrama de estados na Figura 8–83.



▶ **FIGURA 8–83**

SEÇÃO 8–5 Contadores em Cascata

20. Para cada uma das configurações de contadores em cascata dadas na Figura 8–84, determine a freqüência da forma de onda em cada ponto indicado pelo número dentro da circunferência e determine o módulo total.



▶ **FIGURA 8–84**

21. Faça a expansão do contador dado na Figura 8–41 para criar um contador/divisor por 10.000 e um contador/divisor por 100.000.

22. Fazendo uso de diagramas em bloco, mostre como obter as seguintes freqüências a partir de um clock de 10 MHz usando flip-flops individuais, contadores de módulo 5 e contadores de década:

(a) 5 MHz **(b)** 2,5 MHz **(c)** 2 MHz **(d)** 1 MHz **(e)** 500 kHz
(f) 250 kHz **(g)** 62,5 kHz **(h)** 40 kHz **(i)** 10 kHz **(j)** 1 kHz

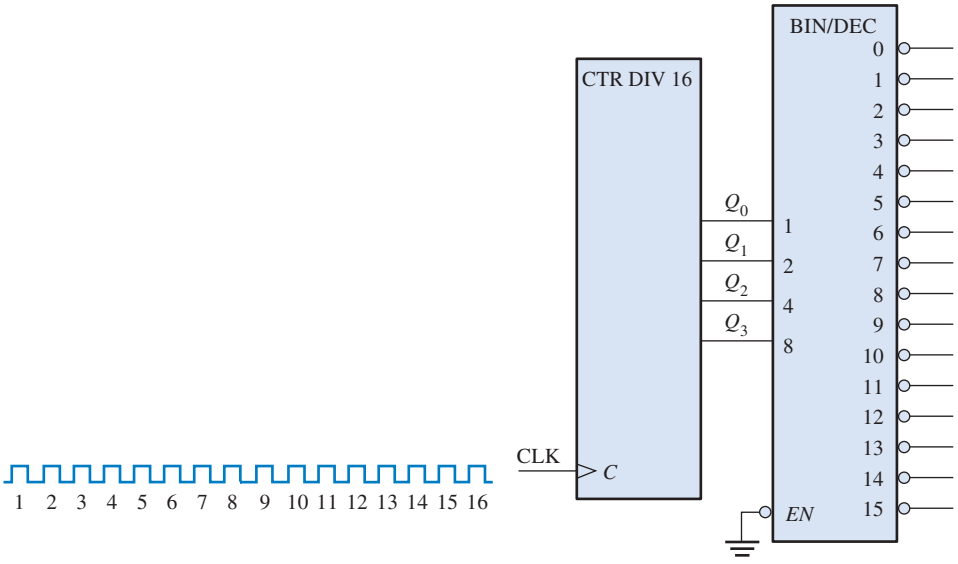
SEÇÃO 8–6 Decodificação de Contador

23. Dado um contador de década BCD com apenas as saídas *Q* disponíveis, mostre qual lógica de decodificação é necessária para decodificar cada um dos seguintes estados e como ela deve ser conectada no contador. Uma indicação de saída em nível ALTO é necessária para cada estado decodificado. O MSB é o bit mais à esquerda.

(a) 0001 **(b)** 0011 **(c)** 0101 **(d)** 0111 **(e)** 1000

24. Para o contador binário de 4 bits conectado ao decodificador na Figura 8–85, determine cada uma das formas de onda em relação aos pulsos de clock.

25. Se o contador visto na Figura 8–85 é assíncrono, determine onde os glitches de decodificação ocorrem nas formas de onda de saída do decodificador.



▶ **FIGURA 8–85**

26. Modifique o circuito dado na Figura 8–85 para eliminar os glitches de decodificação.

27. Analise o contador dado na Figura 8–45 quanto à ocorrência de glitches na saída da porta de decodificação. Caso ocorram glitches, sugira uma forma de eliminá-los.

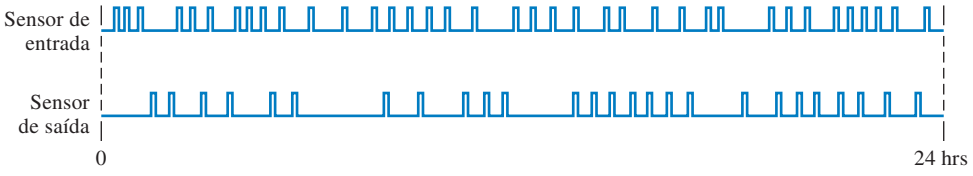
28. Analise o contador dado na Figura 8–46 quanto à ocorrência de glitches nas saídas das portas de decodificação. Caso ocorram glitches, faça uma alteração no projeto de forma a eliminá-los.

SEÇÃO 8–7 Aplicações de Contadores

29. Considere que o relógio digital mostrado na Figura 8–51 esteja inicialmente resetado para 12 horas. Determine o estado binário de cada contador após ocorrerem sessenta e dois pulsos do sinal de 60 Hz.

30. Qual a freqüência de saída de cada contador no circuito do relógio digital mostrado na Figura 8–51?

31. Para o sistema de controle de estacionamento de veículos dado na Figura 8–54, o padrão dos pulsos provenientes dos sensores de entrada e saída durante uma determinado período de 24 horas é mostrado na Figura 8–86. Se já haviam 53 carros no estacionamento no início do período mencionado, qual o estado do contador no final do período de 24 horas.



▶ **FIGURA 8–86**

32. O número binário para o decimal 57 aparece nas entradas de dados em paralelo do conversor de paralelo para série mostrado na Figura 8–56 (D_0 é o LSB). O contador contém inicialmente somente zeros e um clock de 10 kHz é aplicado. Desenvolva o diagrama e temporização mostrando o clock, as saídas do contador e a saída de dados serial.

SEÇÃO 8–9 Análise de Defeito

33. Para o contador na Figura 8–1, mostre o diagrama de temporização para as formas de onda de Q_0 e Q_1 para cada um dos seguintes defeitos (considere que Q_0 e Q_1 estejam inicialmente em nível BAIXO):

(a) entrada de clock de FF0 em curto-circuito para GND

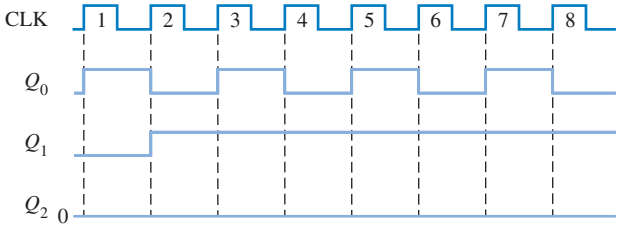
(b) saída Q_0 aberta

(c) entrada de clock de FF1 aberta
(d) entrada *J* de FF0 aberta
(e) entrada *K* de FF1 em curto-circuito para GND

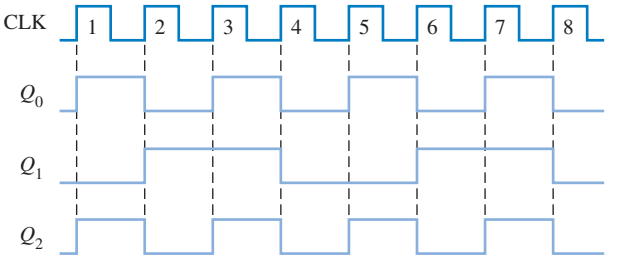
34. Resolva o Problema 33 para o contador dado na Figura 8–11.

35. Identifique o defeito no contador dado na Figura 8–3 analisando as formas de onda dadas na Figura 8–87.

36. A partir do diagrama de forma de onda visto na Figura 8–88, determine o defeito mais provável no contador mostrado na Figura 8–14.

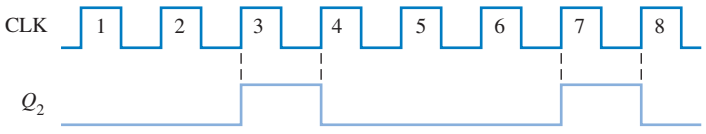


▲ **FIGURA 8–87**



▲ **FIGURA 8–88**

37. Resolva o Problema 36 se a saída Q_2 apresenta a forma de onda observada na Figura 8–89. As saídas Q_0 e Q_1 são as mesmas que as mostradas na Figura 8–88.



▶ **FIGURA 8–89**

38. Aplicamos um clock de 5 MHz no contador em cascata dado na Figura 8–44 e medimos a freqüência de 76,2939 Hz na última saída *RCO*. Isso está correto? Em caso negativo, qual o problema mais provável?

39. Desenvolva uma tabela para uso no teste do contador dado na Figura 8–44 que mostre a freqüência na saída *RCO* final para todos os possíveis defeitos de circuito aberto nas entradas de dados em paralelo (D_0, D_1, D_2 e D_3) considerando uma de cada vez. Use a freqüência de 10 MHz como freqüência de teste para o clock.

40. No display de 7 segmentos, que mostra as dezenas das horas do sistema do relógio digital visto na Figura 8–51, visualizamos continuamente o número 1. Todos os outros dígitos funcionam corretamente. Qual seria o problema?

41. Qual seria a indicação visual relativa à saída Q_1 em aberto na parte do contador das dezenas dos minutos no circuito mostrado na Figura 8–51? Veja também a Figura 8–52.

42. Um dia (talvez uma segunda-feira) verifica-se que começa a aumentar as reclamações dos clientes que fazem uso de um estacionamento controlado por um sistema como o ilustrado nas Figuras 8–54 e 8–55. Os clientes dizem que entram no estacionamento porque a cancela está levantada e o sinal de LOTADO está apagado, porém, uma vez dentro do estacionamento, não encontram vagas. Como um técnico responsável pela manutenção, que problema você acha que existe e como faria a análise de defeito para reparar o sistema o quanto antes?

Aplicações em Sistemas Digitais

43. Implemente a lógica de entrada no circuito seqüencial do sistema de controle de semáforo usando apenas portas NAND.

44. Substitua os flip-flops D no contador de estados de código Gray dado na Figura 8–67 por flip-flops J-K.

45. Especifique o que pode ser feito para alterar o intervalo de tempo da luz verde de 25 s para 60 s.

Problemas Especiais de Projeto

46. Projete um contador de módulo 1000 usando CIs contadores de década 74F162.

47. Modifique o projeto do contador dado na Figura 8–44 para obter um módulo de 30.000.

48. Repita o Problema 47 para um módulo de 50.000.

49. Modifique o circuito do relógio digital mostrado nas Figuras 8–51, 8–52 e 8–53 de forma que ele possa ser presetado para qualquer hora desejada.

50. Projete um circuito de alarme para o relógio digital que pode detectar uma hora predeterminada (horas e minutos apenas) e produza um sinal para ativar um alarme audível.

51. Modifique o projeto do circuito dado na Figura 8–55 para um estacionamento com 1000 vagas e um outro para 3000 vagas.

52. Implemente a lógica de conversão de dados de paralelo para série dado na Figura 8–56 com dispositivos de função fixa específicos.

53. No Problema 15 verificou-se que o contador fica preso e alterna entre dois estados. Conclui-se que essa operação é o resultado de uma falha de projeto. Reprojete o contador de forma que quando ele for para o segundo dos estados nos quais fica preso, ele recicle para o estado de somente zeros no próximo pulso de clock.

54. Modifique o diagrama em bloco do sistema de controle de semáforo dado na Figura 8–63 de forma a acrescentar 15 s ao sinal de virar à esquerda na via principal imediatamente antes da luz verde.

Prática de Análise de Defeito Usando o Multisim

55. Abra o arquivo P08-55 e teste o contador assíncrono de 4 bits para determinar se existe defeito. Caso exista um defeito, Identifique-o se possível.

56. Abra o arquivo P08-56 e teste o contador síncrono de 3 bits para determinar se existe defeito. Caso exista um defeito, Identifique-o se possível.

57. Abra o arquivo P08-57 e teste o contador BCD para determinar se existe defeito. Caso exista um defeito, Identifique-o se possível.

58. Abra o arquivo P08-58 e teste o CI contador binário de 4 bits 74163 para determinar se existe defeito. Caso exista um defeito, Identifique-o se possível.

59. Abra o arquivo P08-59 e teste o CI contador de década Crescente/Decrescente 74190 para determinar se existe defeito. Caso exista um defeito, Identifique-o se possível.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 8–1 Operação de Contadores Assíncronos

1. Assíncrono significa que cada flip-flop após o primeiro é habilitado pela saída do flip-flop anterior.

2. Um contador de módulo 14 tem quatorze estados necessitando de quatro flip-flops.

SEÇÃO 8–2 Operação de Contadores Síncronos

1. Todos os flip-flops num contador síncrono recebem clock simultaneamente.

2. O contador pode ser presetado (inicializado) para qualquer estado dado.
3. O contador é habilitado quando *ENP* e *ENT* estiverem em nível ALTO; *RCO* vai para nível ALTO quando o estado final da seqüência é alcançado.

SEÇÃO 8–3 Contadores Síncronos Crescente/Descrescente

1. O contador vai para 1001.
2. CRESCENTE (UP):1111: DECRESCENTE (DOWN): 0000; o próximo estado é 1111.

SEÇÃO 8–4 Projeto de Contadores Síncronos

1. $J = 1$, $K =$ X ("dont' care")
2. $J =$ X ("don't care"), $K = 0$
3. **(a)** O próximo estado é 1011.
   (b) Q_3 (MSB): repouso ou SET; Q_2: repouso ou RESET; Q_1: repouso ou SET; Q_0 (LSB): SET ou toggle

SEÇÃO 8–5 Contadores em Cascata

1. Três contadores de década produzem ÷1000; 4 contadores de década produzem ÷10.000.
2. **(a)** ÷20: flip-flop e DIV 10
   (b) ÷32: flip-flop e DIV 16
   (c) ÷160: DIV 16 e DIV 10
   (d) ÷320: DIV 16 e DIV 10 e flip-flop

SEÇÃO 8–6 Decodificação de Contador

1. **(a)** Nenhum estado de transição porque existe uma mudança num único bit.
   (b) 0000, 0001, 0010, 0101, 0110, 0111
   (c) Nenhum estado de transição porque existe uma mudança num único bit.
   (d) 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110

SEÇÃO 8–7 Aplicações de Contadores

1. A porta G_1 reseta o flip-flop no primeiro pulso de clock após a contagem 12. A porta G_2 decodifica a contagem 12 para presetar o contador para 0001.
2. O contador de década das horas avança através de cada estado de zero a nove e, conforme ele recicla de nove para zero, o flip-flop é comutado para o estado SET. Isso produz um dez (10) no display. Quando o contador de década das horas estiver no estado 12, a porta NAND decodificadora faz com que o contador recicle para o estado 1 no próximo pulso de clock. O flip-flop *reseta*. Isso resulta um 01 no display.

SEÇÃO 8–8 Símbolos Lógicos com Notação de Dependência

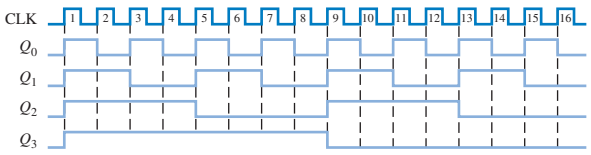
1. *C*: controle, geralmente clock. *M*: modo; *G*: AND
2. *D* indica armazenamento de dados.

SEÇÃO 8–9 Análise de Defeito

1. Nenhum pulso nas saídas *TC*: *CTEN* do primeiro contador em curto-circuito para GND ou para nível BAIXO; A saída *TC* do primeiro contador em curto-circuito para GND ou para nível BAIXO.
2. Com a saída do inversor aberta, o contador não recicla na contagem presetada mas funciona como um contador de módulo total.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

8–1. Veja a Figura 8–90.

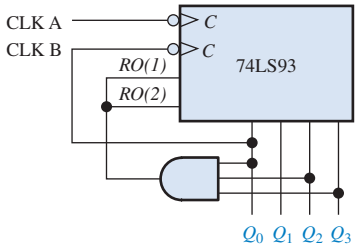


▶ FIGURA 8–90

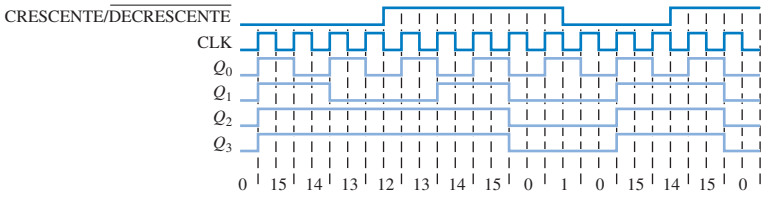
8–2. Conecte Q_0 na porta NAND como uma terceira entrada (Q_2 e Q_3 são as outras duas entradas). Conecte a linha $\overline{CLR}$ na entrada $\overline{CLR}$ de FF0 bem como FF2 e FF3.

8–3. Veja a Figura 8–91.

8–4. Veja a Figura 8–92.



▲ FIGURA 8–91



▲ FIGURA 8–92

▼ TABELA 8–15

ESTADO INVÁLIDO ATUAL			ENTRADAS J-K						PRÓXIMO ESTADO			
Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	Q_2	Q_1	Q_0	
0	0	0	0	0	1	1	1	1	0	1	1	
0	1	1	1	1	1	1	1	1	1	0	0	
1	0	0	0	0	1	1	1	0	1	1	1	estado válido
1	1	0	1	1	1	1	1	0	0	0	1	estado válido

8–5. Veja a Tabela 8–15.

8–6. A aplicação da lógica Booleana à lógica mostrada na Figura 8–37 mostra que a saída de cada porta OR está de acordo com a expressão no Passo 5.

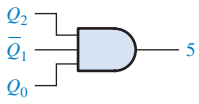
8–7. São necessários 5 contadores de década. $10^5 = 100.000$

8–8. $f_{Q0} = 1\ \text{MHz}/[(10)(2)] = 50\ \text{kHz}$

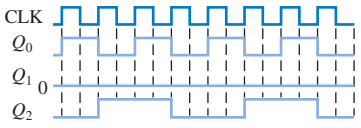
8–9. Veja a Figura 8–93.

8–10. $8AC0_{16}$ seria carregado. $16^4 - 8AC0_{16} = 65.536 - 32.520 = 30.016$
$f_{TC4} = 10\ \text{MHz}/30.016 = 333{,}2\ \text{Hz}$

8–11. Veja a Figura 8–94.



▲ **FIGURA 8–93**



▲ **FIGURA 8–94**

AUTOTESTE

1. (a) **2.** (b) **3.** (b) **4.** (c) **5.** (a) **6.** (c) **7.** (b) **8.** (c)
9. (d) **10.** (a) **11.** (c) **12.** (b) **13.** (b) **14.** (d)

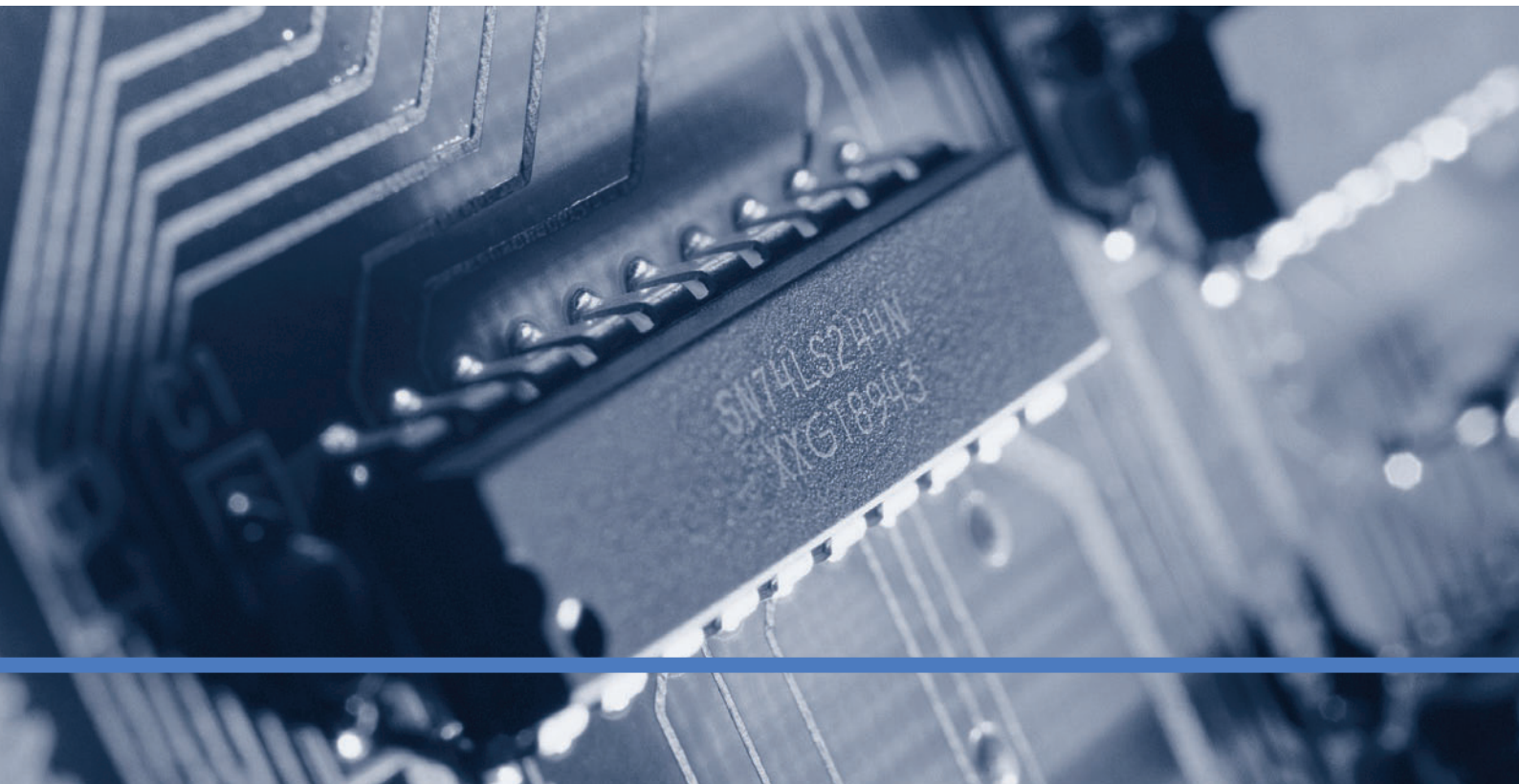
9

REGISTRADORES DE DESLOCAMENTO

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Identificar as formas básicas de movimentação de dados em registradores de deslocamento
- Explicar como operam os registradores de deslocamento com entrada serial/saída serial, entrada serial/saída paralela, entrada paralela/saída serial e entrada paralela/saída paralela
- Descrever como opera um registrador de deslocamento bidirecional



- Determinar a seqüência de um contador Johnson
- Configurar um contador em anel para gerar uma seqüência específica
- Construir um contador em anel a partir de um registrador de deslocamento
- Usar um registrador de deslocamento como um dispositivo de atraso de tempo
- Usar um registrador de deslocamento para implementar um conversor de dados de serial para paralelo
- Implementar um codificador de teclado controlado por um registrador de deslocamento básico
- Interpretar símbolos, com notação de dependência, de registradores de deslocamento padrão 91-1984 da ANSI/IEEE
- Usar registradores de deslocamento em aplicações de sistemas

TERMOS IMPORTANTES

- Registrador
- Estágio
- Deslocamento
- Carga
- Bidirecional

INTRODUÇÃO

Os registradores de deslocamento são um tipo de circuito lógico muito parecido com os contadores digitais. Os registradores são usados principalmente no armazenamento de dados digitais e não possuem uma característica interna de seqüência de estado como os contadores. Entretanto, existem exceções e essas são abordadas na Seção 9-7.

Neste capítulo, os tipos básicos de registradores de deslocamento são estudados e diversas aplicações são apresentadas. Além disso, é introduzido um método de análise de defeito.

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

74XX164 74XX165 74XX174
74XX194 74XX195

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

Esse tópico ilustra os conceitos deste capítulo. Um sistema de controle de segurança num prédio é introduzido. O sistema usa dois tipos de registradores de deslocamento, bem como outros tipos de dispositivos abordados em capítulos anteriores. O sistema também inclui uma memória que será foco de Aplicações em Sistemas Digitais no Capítulo 10.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

9-1 FUNÇÕES BÁSICAS DE REGISTRADORES DE DESLOCAMENTO

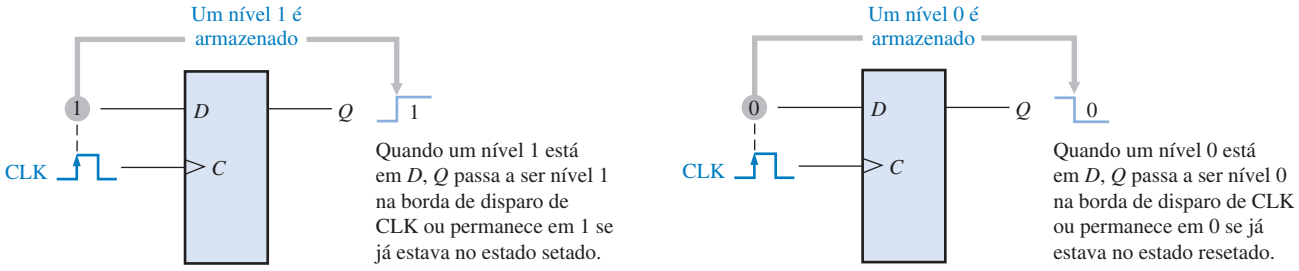
Os registradores de deslocamento consistem de arranjos de flip-flops e são importantes em aplicações que envolvem o armazenamento e a transferência de dados em sistemas digitais. Um registrador, diferentemente de um contador não tem uma seqüência de estados específica, exceto em certas aplicações muito especializadas. Um registrador, em geral, é usado somente para armazenamento e deslocamento de dados (1s e 0s) recebidos de uma fonte externa e normalmente não possui características internas de seqüência de estados.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar como um flip-flop armazena um bit de dado ■ Definir a capacidade de armazenamento de um registrador de deslocamento ■ Descrever a capacidade de deslocamento de um registrador

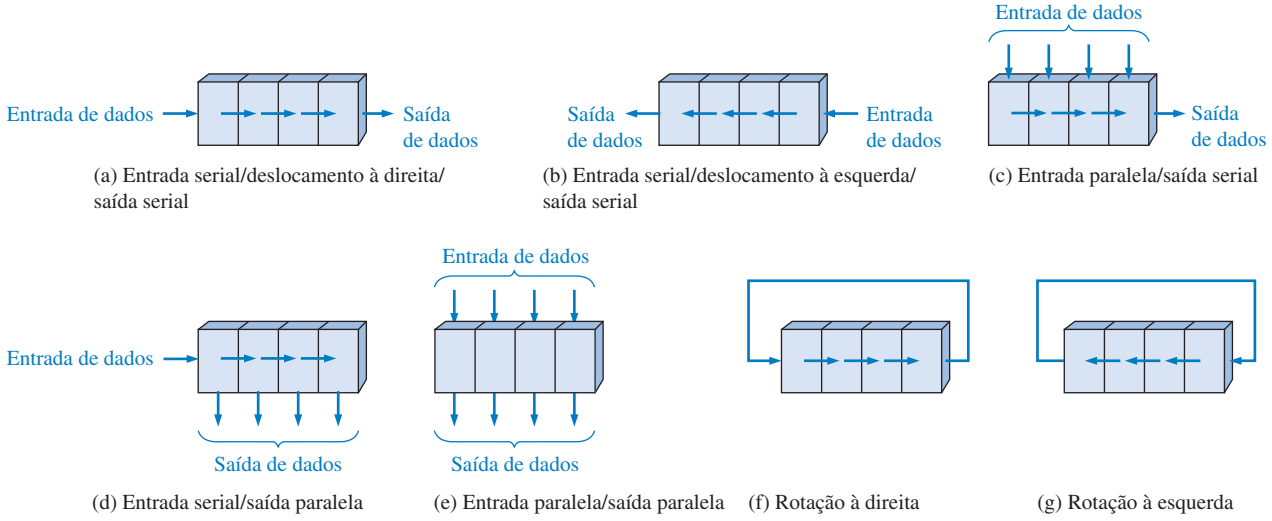
Um registrador pode consistir em um ou mais flip-flops usados para armazenar e deslocar dados.

Um **registrador** é um circuito digital com duas funções básicas: armazenamento de dados e movimentação de dados. A capacidade de armazenamento de um registrador o torna um importante tipo de dispositivo de memória. A Figura 9–1 ilustra o conceito de armazenamento de um nível 1 ou nível



▲ **FIGURA 9–1**

O flip-flop como um elemento de armazenamento.



▲ **FIGURA 9–2**

Movimentos básicos de dados em registradores de deslocamento. (São usados quatro bits como ilustração. Os bits se movem na direção das setas.)

0 num flip-flop D. Um nível 1 é aplicado na entrada de dados como mostrado, e um pulso de clock é aplicado armazenando o nível 1 *setando* o flip-flop. Quando o nível 1 na entrada é removido, o flip-flop permanece no estado setado, armazenando assim o 1. Um procedimento similar se aplica ao armazenamento de um nível 0 *resetando* o flip-flop, como também está ilustrado na Figura 9–1.

A *capacidade de armazenamento* de um registrador é o número total de bits (1s e 0s) dos dados digitais que ele pode reter. Cada **estágio** (flip-flop) é um registrador de deslocamento que representa um bit da capacidade de armazenamento; portanto, o número de estágios num registrador determina sua capacidade de armazenamento.

A capacidade de **deslocamento** de um registrador permite o movimento de dados de um estágio para outro dentro do registrador ou ainda para dentro ou para fora do registrador com a aplicação de pulsos de clock. A Figura 9–2 ilustra os tipos de movimentos de dados em registradores de deslocamento.

SEÇÃO 9–1 REVISÃO
As respostas estão no final do capítulo.

1. Em geral, qual é a diferença entre um contador e um registrador de deslocamento?
2. Quais as duas funções principais realizadas por um registrador de deslocamento?

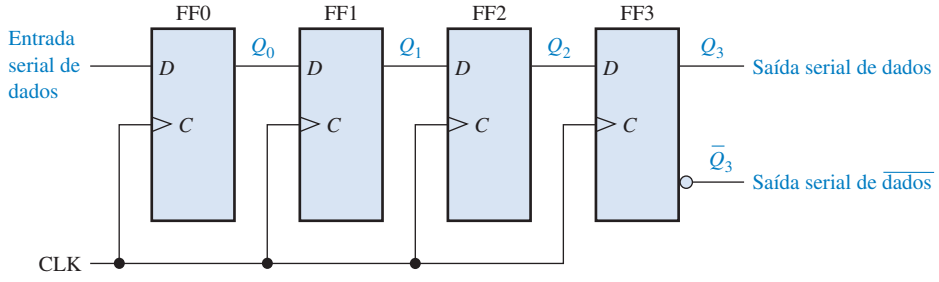
9-2 REGISTRADORES DE DESLOCAMENTO COM ENTRADA SERIAL/SAÍDA SERIAL

O registrador de deslocamento com entrada serial/saída serial aceita dados seriais – ou seja, um bit de cada vez numa única linha. Ele gera em sua saída a informação armazenada também de forma serial.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar como os bits de dados são inseridos de forma serial num registrador de deslocamento ■ Descrever como os bits de dados são deslocados através do registrador ■ Explicar como os bits de dados são obtidos de forma serial a partir de um registrador de deslocamento ■ Desenvolver e analisar diagramas de temporização para registradores com entrada serial/saída serial

Vamos analisar primeiro a entrada serial de dados num registrador de deslocamento típico. A Figura 9–3 mostra um dispositivo de 4 bits implementado com flip-flops D. Com quatro estágios, esse registrador de deslocamento pode armazenar até quatro bits de dados.



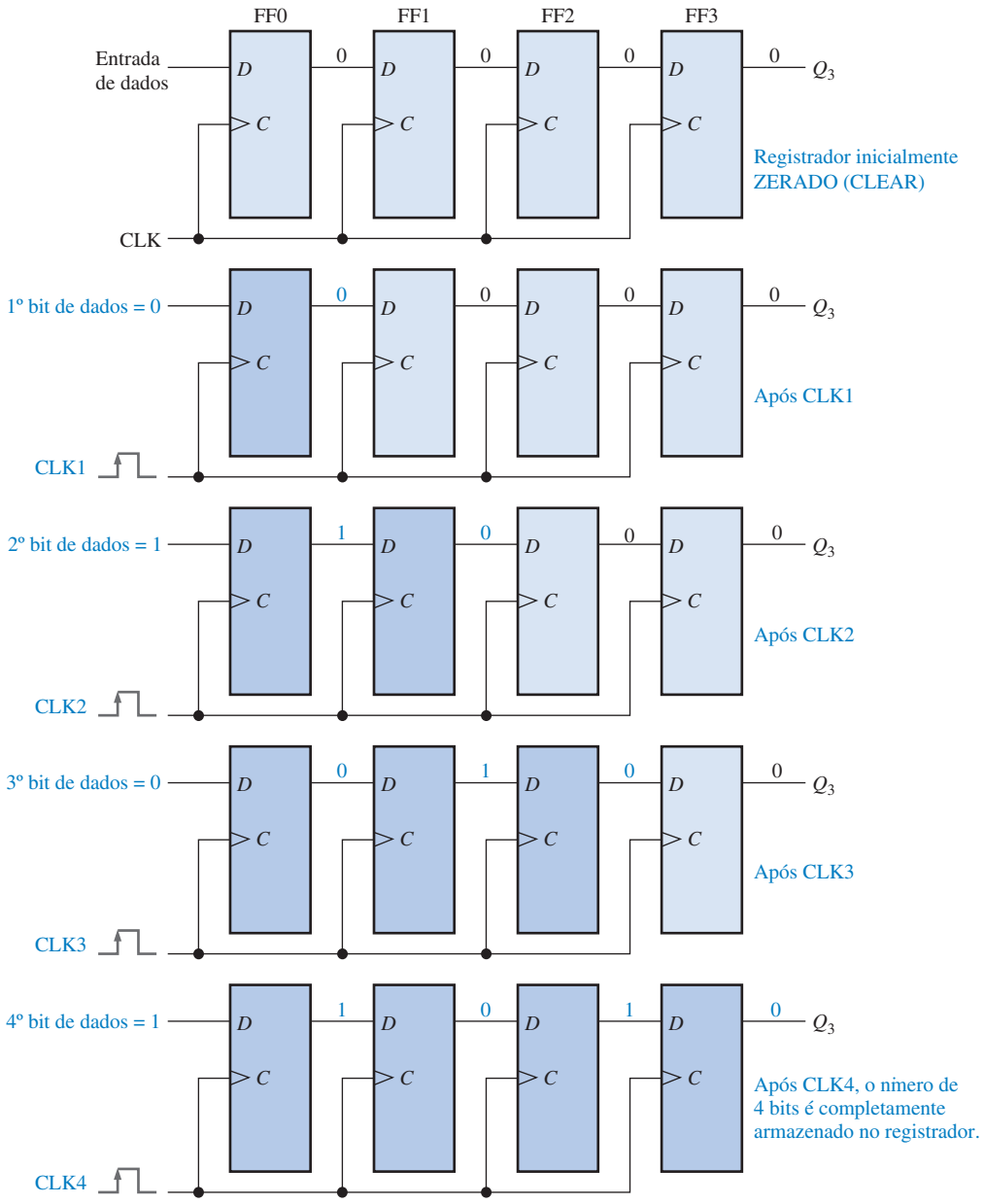
▲ **FIGURA 9–3**
Registrador com entrada serial/saída serial.

A Figura 9–4 ilustra uma entrada de quatro bits (1010) no registrador, começando com o bit mais à direita. O registrador está inicialmente zerado. O nível 0 é colocado na linha de entrada de dados, fazendo $D = 0$ para o FF0. Quando o primeiro pulso de clock é aplicado, FF0 é resetado, armazenando assim o nível 0.

Em seguida, o segundo bit, o qual é nível 1, é aplicado na entrada de dados, fazendo $D = 1$ para FF0 e $D = 0$ para FF1 porque a entrada D de FF1 está conectada na saída Q_0. Quando o segun-

NOTA: COMPUTAÇÃO

Freqüentemente, é necessário *limpar* (zerar) um registrador interno de um computador. Por exemplo, um registrador pode ser zerado antes de uma operação aritmética ou outra operação. Uma forma na qual registradores de computadores são zerados é usando software para subtrair o conteúdo de um registro dele próprio. O resultado sempre será zero. Por exemplo, uma instrução de computador que realiza essa operação é SUB AL,AL. Com essa instrução, o registrador denominado AL é zerado.



▲ **FIGURA 9–4**

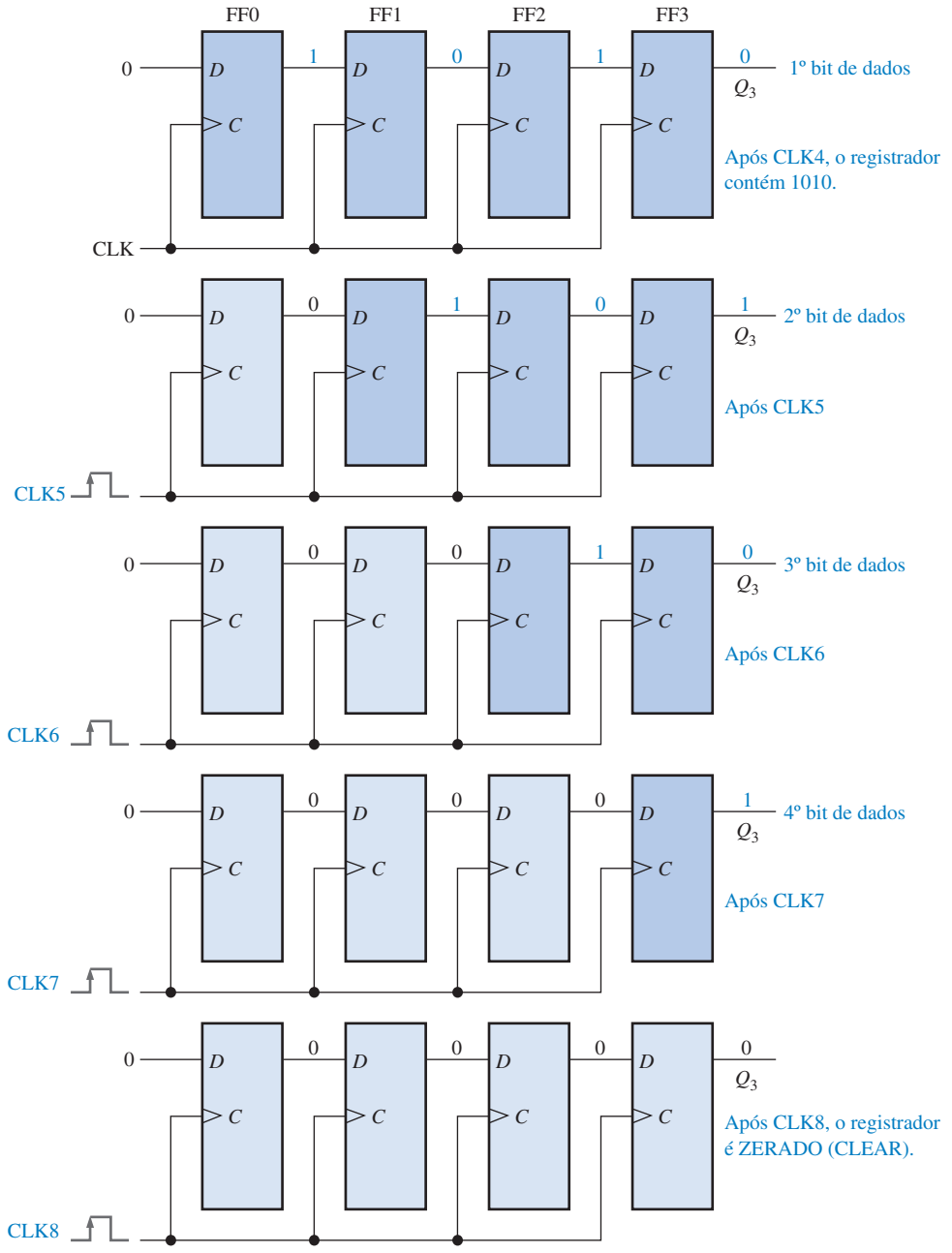
Quatro bits (1010) sendo inseridos de forma serial num registrador.

do pulso de clock ocorre, o nível 1 na entrada de dados é deslocado para FF0, fazendo com que FF0 seja setado; e o nível 0 que estava em FF0 é deslocado para FF1.

O terceiro bit, um nível 0, é agora colocado na linha de entrada de dados e um pulso de clock é aplicado. O nível 0 entra para o FF0, o nível 1 armazenado no FF0 é deslocado para o FF1 e o nível 0 armazenado em FF1 é deslocado para FF2.

Para dados seriais, um bit de cada vez é transferido.

O último bit, um nível 1, é colocado agora na entrada de dados e um pulso de clock é aplicado. Dessa vez o nível 1 entra no FF0, o nível 0 armazenado em FF0 é deslocado para FF1, o nível 1 armazenado em FF1 é deslocado para FF2 e o nível 0 armazenado em FF2 é deslocado para FF3. Isso completa a entrada serial dos quatro bits no registrador de deslocamento, onde eles podem ser armazenados por algum tempo enquanto os flip-flops tiverem alimentação cc.



▲ **FIGURA 9–5**

Quatro bits (1010) sendo deslocados de forma serial e substituídos somente por zeros.

Se desejarmos obter os dados do registrador, os bits têm que ser deslocados de forma serial para fora sendo obtido a partir da saída Q_3, conforme a Figura 9–5 ilustra. Após CLK4 na operação de entrada de dados descrita, o bit mais à direita, 0, aparece na saída Q_3. O pulso de clock CLK6 desloca o terceiro bit para a saída e CLK7 desloca o quarto bit para a saída. Enquanto os quatro bits originais são deslocados para fora, mais bits podem ser deslocados para dentro do registrador. Bits zeros são mostrados sendo deslocados para dentro do registrador.

EXEMPLO 9–1

Mostre os estados do registrador de 5 bits mostrado na Figura 9–6 para as formas de onda de entrada de dados especificada e do clock. Considere que o registrador esteja inicialmente zerado (todos os bits em nível 0).

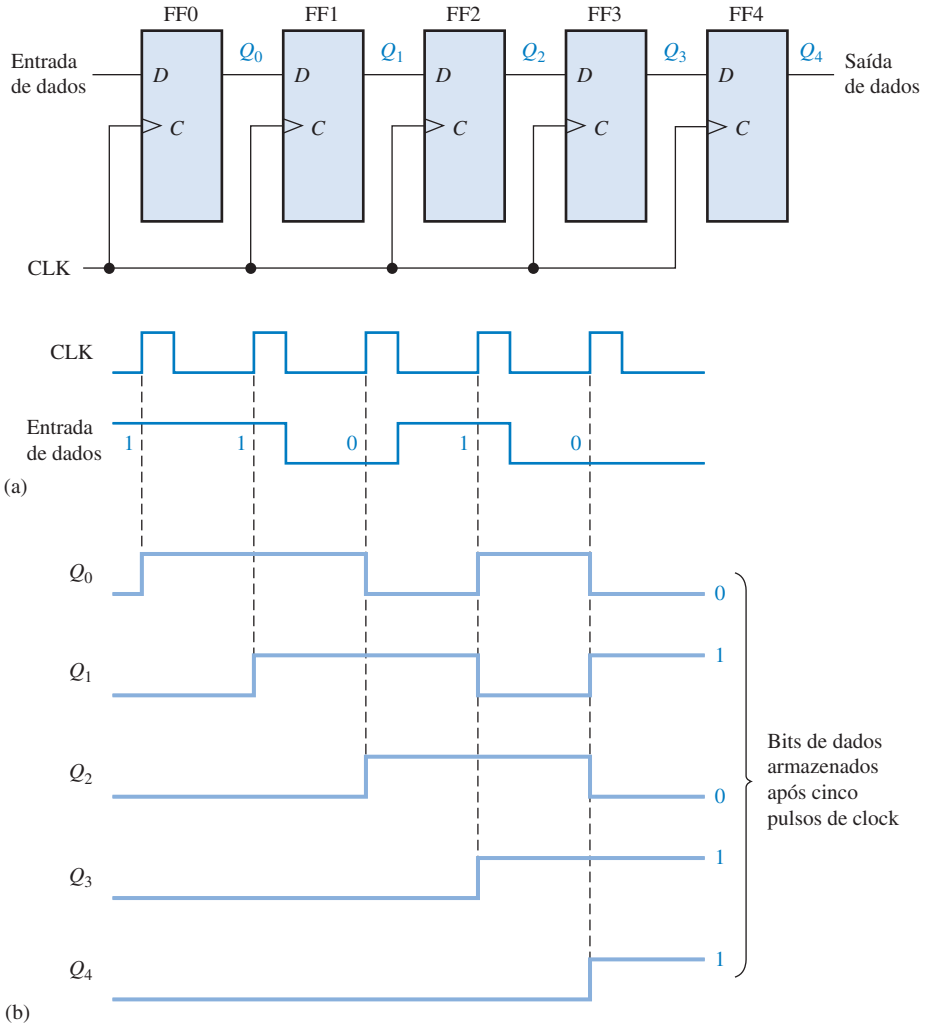


FIGURA 9–6

Abra o arquivo F09-06 para verificar a operação.

Solução O primeiro bit de dado (1) é inserido no registrador no primeiro pulso de clock e em seguida deslocado da esquerda para a direita enquanto os bits restantes são inseridos e deslocados. O registrador contém $Q_4Q_3Q_2Q_1Q_0 = 11010$ após cinco pulsos de clock. Veja a Figura 9–6(b).

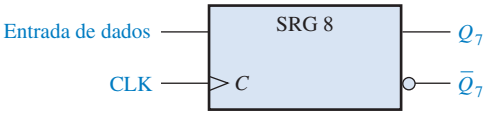
Problema relacionado* Mostre os estados do registrador se a entrada de dados for invertida (complementada). O registrador está inicialmente zerado.

* As respostas estão no final do capítulo.

A Figura 9–7 mostra um símbolo em bloco lógico tradicional para um registrador de deslocamento de 8 bits com entrada serial/saída serial. A indicação "SRG 8" indica um registrador de deslocamento (SRG) com uma capacidade de 8 bits.

FIGURA 9–7

Símbolo lógico para um registrador de deslocamento de 8 bits com entrada serial/saída serial.



SEÇÃO 9–2 REVISÃO

1. Desenvolva o diagrama lógico para o registrador de deslocamento dado na Figura 9–3, usando flip-flops J-K para substituir os flip-flops D.
2. Quantos pulsos de clock são necessários para inserir um byte de dados de forma serial num registrador de deslocamento de 8 bits?

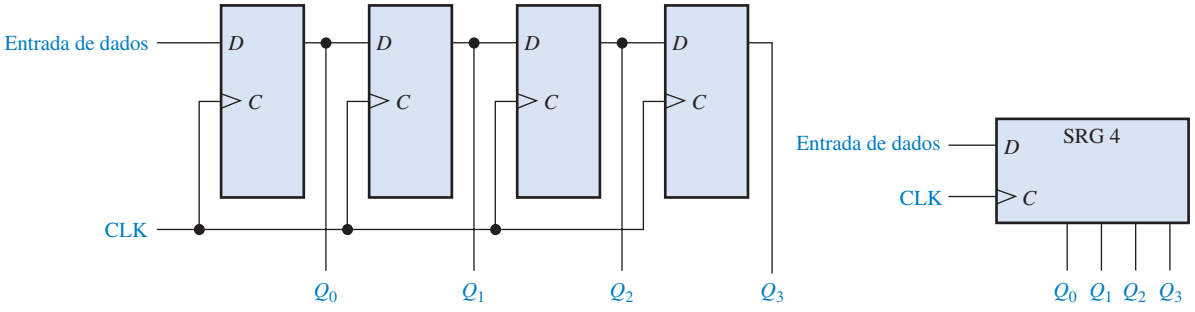
9-3 REGISTRADORES DE DESLOCAMENTO COM ENTRADA SERIAL/SAÍDA PARALELA

Os bits de dados são inseridos serialmente (primeiro o bit mais à direita) nesse tipo de registrador da mesma forma conforme discutido na Seção 9–2. A diferença está na forma na qual os bits de dados são obtidos na saída do registrador; num registrador com saída paralela, a saída de cada estágio está disponível. Uma vez armazenados os dados, cada bit aparece em sua linha de saída respectiva e todos os bits são disponibilizados simultaneamente, em vez de um bit de cada vez como no registrador com saída serial.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar como os bits de dados são obtidos na saída de um registrador de deslocamento em paralelo ■ Comparar a saída serial com a saída paralela ■ Discutir o CI registrador de deslocamento de 8 bits 74HC164 ■ Desenvolver e analisar os diagramas de temporização para registradores com entrada serial/saída paralela

A Figura 9–8 mostra um registrador de deslocamento de 4 bits com entrada serial/saída paralela e o seu símbolo lógico em bloco.



▲ **FIGURA 9–8**

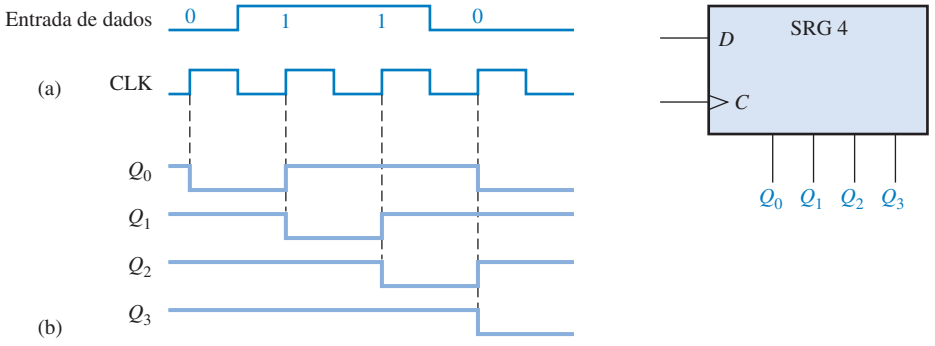
Um registrador de deslocamento com entrada serial/saída paralela.

EXEMPLO 9–2

Mostre os estados do registrador de 4 bits (SRG 4) para as formas de onda dadas na Figura 9–9(b).

Solução O registrador contém 0110 após quatro pulsos de clock. Veja a Figura 9–9(b).

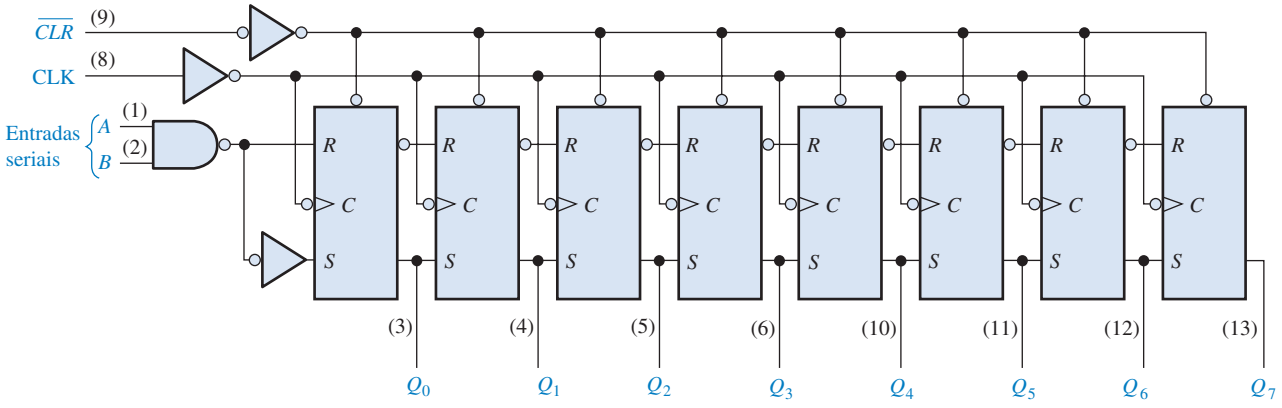
Problema relacionado Se a entrada de dados permanece em 0 após o quarto pulso de clock, qual o estado do registrador após três pulsos adicionais de clock?



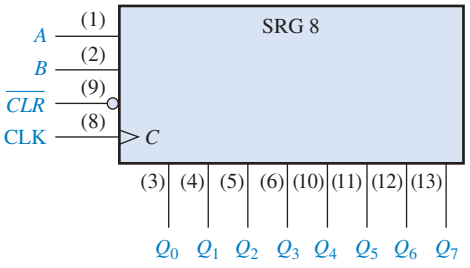
▶ **FIGURA 9–9**

REGISTRADOR DE DESLOCAMENTO DE 8 BITS COM ENTRADA SERIAL/SAÍDA PARALELA (74HC164)

O CI 74HC164 é um exemplo de um CI registrador de deslocamento que tem entrada serial/saída paralela. O diagrama lógico é mostrado na Figura 9–10(a) e o símbolo lógico em bloco é mostrado na parte (b). Observe que esse dispositivo tem duas entradas seriais com possibilidade de controle, *A* e *B*, e uma entrada de clear ($\overline{CLR}$) que é ativa em nível BAIXO. As saída paralelas são de Q_0 a Q_7.



(a) Diagrama lógico

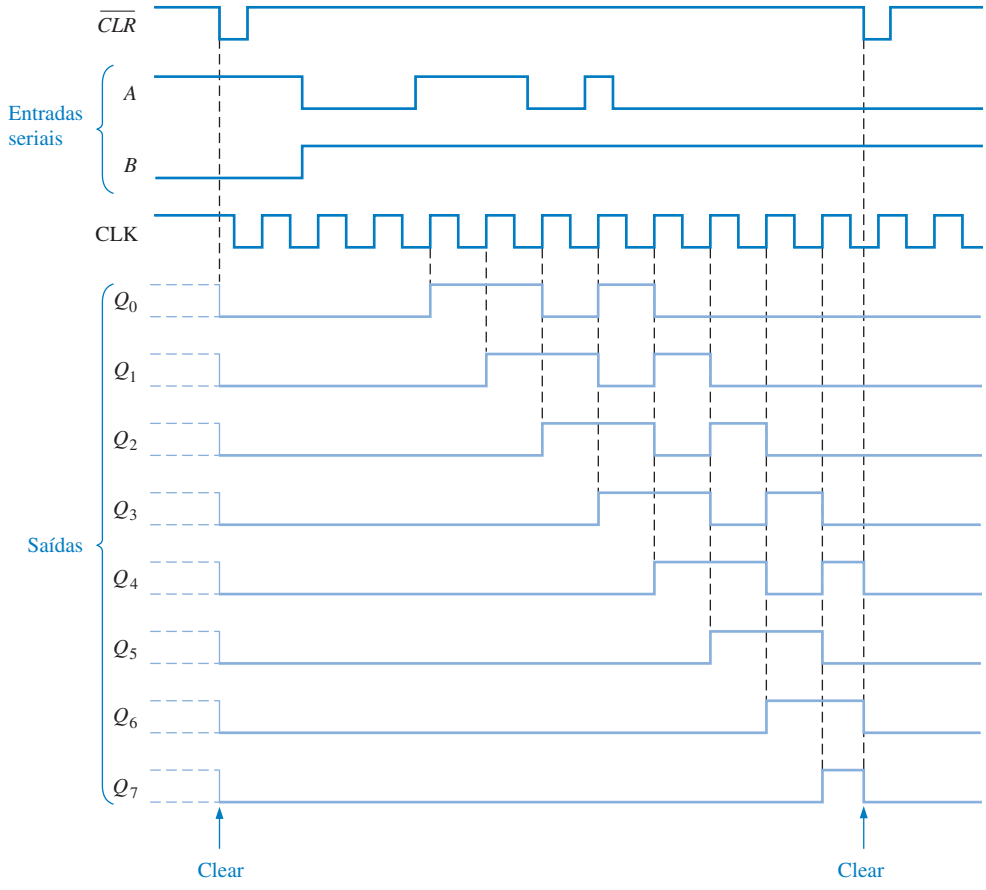


(b) Símbolo lógico

▲ **FIGURA 9–10**

O CI 74HC164 registrador de deslocamento de 8 bits com entrada serial/saída paralela.

Uma amostra de um diagrama de temporização do CI 74HC164 é mostrado na Figura 9–11. Observe que os dados seriais de entrada em *A* são deslocados através do registrador após a entrada *B* ir para nível ALTO.



▲ **FIGURA 9–11**

Amostra de um diagrama de temporização para o CI registrador de deslocamento 74HC164.

SEÇÃO 9–3 REVISÃO

1. A seqüência de bits 1101 é inserida serialmente (o bit mais à direita é o primeiro) num registrador de deslocamento de 4 bits com saída paralela que está inicialmente resetado. Quais são os estados das saídas *Q* após dois pulsos de clock?
2. Como um registrador com entrada serial/saída paralela pode ser usado como um registrador com entrada serial/saída serial?

9-4 REGISTRADORES DE DESLOCAMENTO COM ENTRADA PARALELA/SAÍDA SERIAL

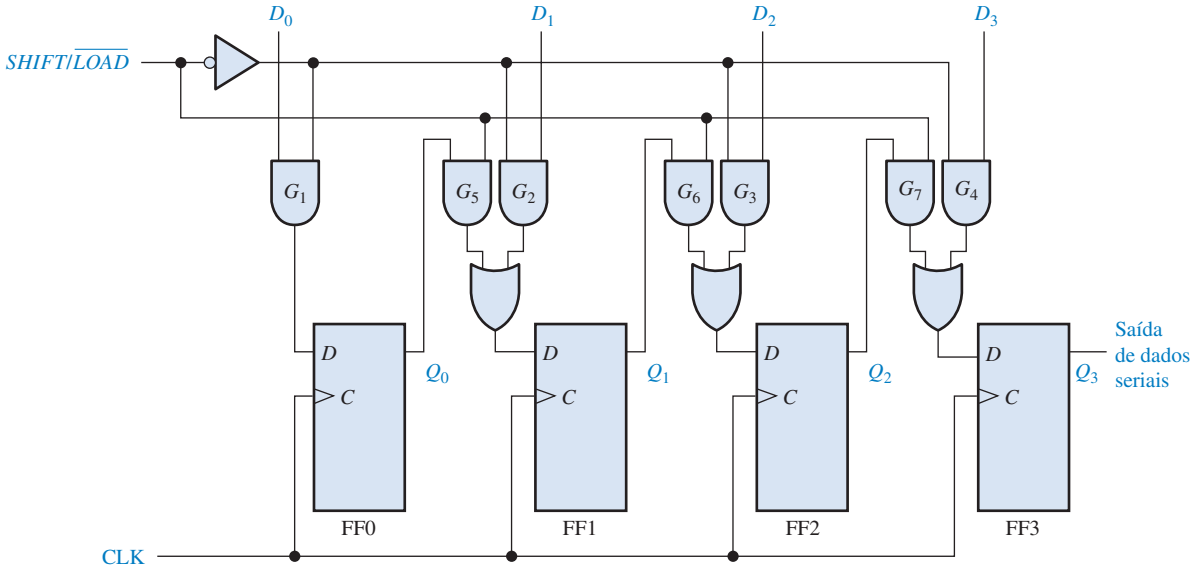
Para um registrador com entrada de dados em paralelo, os bits são inseridos simultaneamente nos seus respectivos estágios em linhas paralelas em vez de bit a bit numa única linha como acontece com a entrada serial de dados. A saída serial é a mesma descrita na Seção 9–2, uma vez que os dados estejam completamente armazenados no registrador.

Ao final do estudo desta seção você deverá ser capaz de:

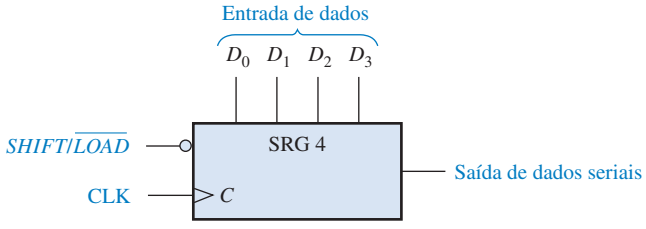
■ Explicar como os bits de dados são inseridos num registrador de deslocamento com entrada paralela ■ Comparar a entrada serial com a entrada paralela ■ Discutir o CI registrador de deslocamento de 8 bits com entrada paralela 74HC165 ■ Desenvolver e analisar diagramas de temporização para registradores com entrada paralela/saída serial

Para dados em paralelo, múltiplos bits são transferidos de cada vez.

A Figura 9–12 ilustra um registrador de deslocamento e um símbolo lógico típico. Observe que existem quatro linhas de entradas de dados, (D_0, D_1, D_2 e D_3) e uma entrada $SHIFT/\overline{LOAD}$, a qual permite a **carga** (load) dos quatro bits em paralelo no registrador. Quando $SHIFT/\overline{LOAD}$ for nível BAIXO, as portas G_1 a G_4 são habilitadas, permitindo que cada bit de dado seja aplicado na entrada D do seu respectivo flip-flop. Quando um pulso de clock for aplicado, os flip-flops com $D = 1$ serão setados e com $D = 0$ resetados, armazenando assim todos os quatro bits simultaneamente.



(a) Diagrama lógico



(b) Símbolo lógico

▲ FIGURA 9–12

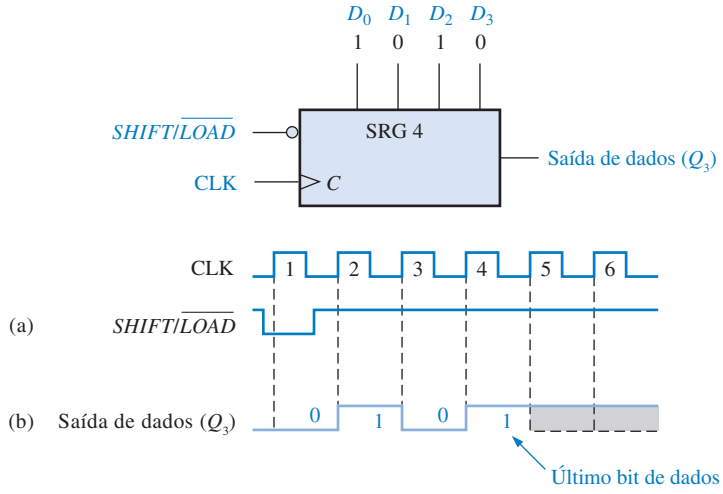
Um registrador de deslocamento de 4 bits com entrada paralela/saída serial. Abra o arquivo F09-12 para verificar a operação.

Quando $SHIFT/\overline{LOAD}$ for nível ALTO, as portas de G_1 a G_4 são desabilitadas e as portas de G_5 a G_7 são habilitadas, permitindo o deslocamento (shift) dos bits de dados à direita de um estágio pa-

ra o próximo. As portas OR permitem a operação de deslocamento normal ou a operação de entrada de dados em paralelo, dependendo de quais portas AND são habilitadas pelo nível lógico na entrada $SHIFT/\overline{LOAD}$. Observe que FF0 tem uma única porta AND para desabilitar a entrada paralela D_0. Não é necessário nesse caso um arranjo AND/OR por que não existe entrada serial de dados.

EXEMPLO 9–3

Mostre a forma de onda na saída de dados para um registrador de 4 bits com entrada paralela de dados a partir das formas de onda de $SHIFT/\overline{LOAD}$ e do clock dadas na Figura 9–13(a). Consulte o digrama lógico na Figura 9–12(a).



▲ FIGURA 9–13

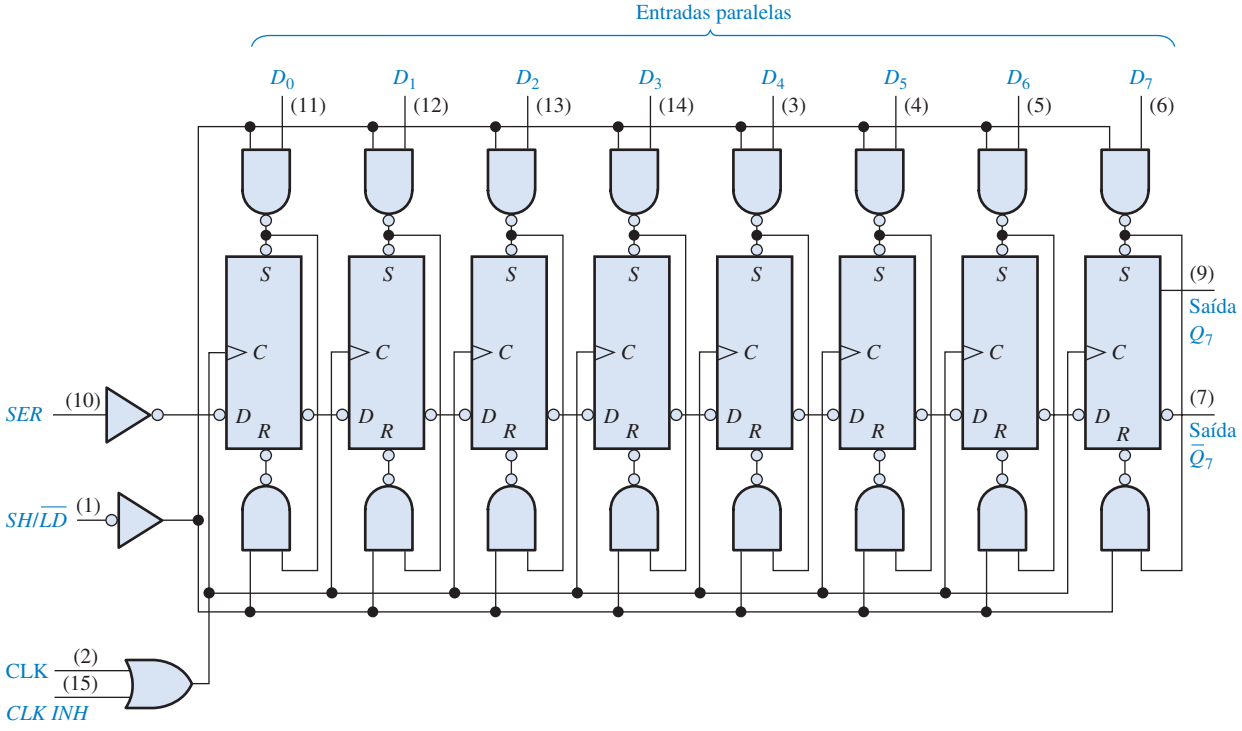
Solução No pulso de clock 1, os dados em paralelo ($D_0D_1D_2D_3 = 1010$) são carregados no registrador, fazendo Q_3 igual a 0. No pulso de clock 2 o nível 1 de Q_2 é deslocado para Q_3; no pulso de clock 3 o nível 0 é deslocado para Q_3; no pulso de clock 4 o último bit de dados (1) é deslocado para Q_3; e no pulso de clock 5 todos os bits foram deslocados para fora, e apenas os 1s restantes no registrador (considerando que a entrada D permanece em 1). Veja a Figura 9–13(b).

Problema relacionado Mostre a forma de onda na saída de dados para as entradas de clock e $SHIFT/\overline{LOAD}$ mostradas na Figura 9–13(a) se os dados paralelos são $D_0D_1D_2D_3 = 0101$.

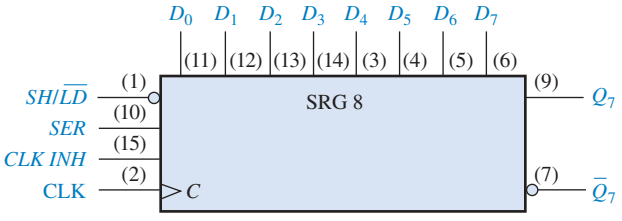
REGISTRADOR DE DESLOCAMENTO DE 8 BITS COM CARGA PARALELA (74HC165)

O CI 74HC165 é um exemplo de um CI registrador de deslocamento que tem entrada paralela/saída serial (ele também pode operar como entrada serial/saída serial). A Figura 9–14(a) mostra o diagrama lógico interno para esse dispositivo e a parte (b) mostra um símbolo lógico em bloco típico. Um nível BAIXO na entrada $SHIFT/\overline{LOAD}$ $(SH/\overline{LD})$ habilita todos as portas NAND para a carga paralela. Quando um bit de entrada é nível 1, o flip-flop é setado assincronamente por uma saída de nível BAIXO na porta superior.

Quando um bit de dado de entrada for nível 0, o flip-flop é resetado assincronamente por um nível BAIXO na saída da porta inferior. Alternativamente, os dados podem ser inseridos de forma serial na entrada *SER*. Além disso, o clock pode ser desabilitado a qualquer momento com um nível ALTO na entrada CLK INH. As saídas de dados seriais do registrador são Q_7 e o seu complemento $\overline{Q}_7$. Essa implementação é diferente do método síncrono de carga paralela discutido anteriormente, demonstrando que geralmente existem diversas formas de realizar a mesma função.



(a) Diagrama lógico

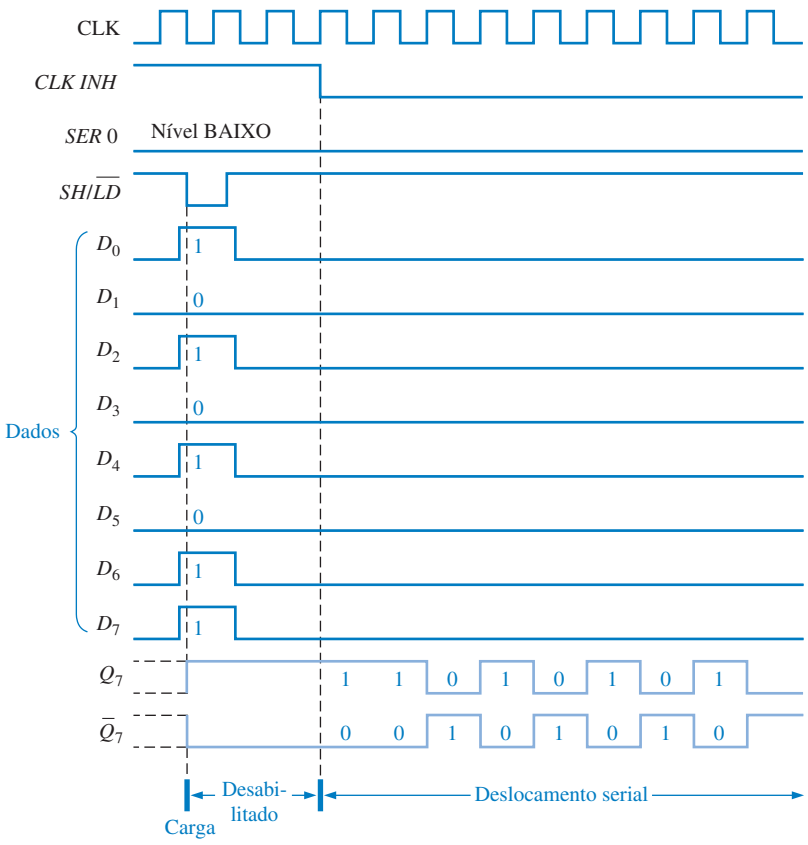


(b) Símbolo lógico

▲ FIGURA 9–14

O CI registrador de deslocamento de 8 bits com carga paralela 74HC165.

A Figura 9–15 é um diagrama de temporização mostrando um exemplo da operação de um CI registrador de deslocamento 74HC165.



▲ **FIGURA 9–15**

Amostra de um diagrama de temporização para o CI registrador de deslocamento 74HC165.

SEÇÃO 9–4 REVISÃO

1. Explique a função da entrada *SHIFT*/$\overline{LOAD}$.
2. A operação de carga paralela no CI registrador de deslocamento 74HC165 é síncrona ou assíncrona? O que isso significa?

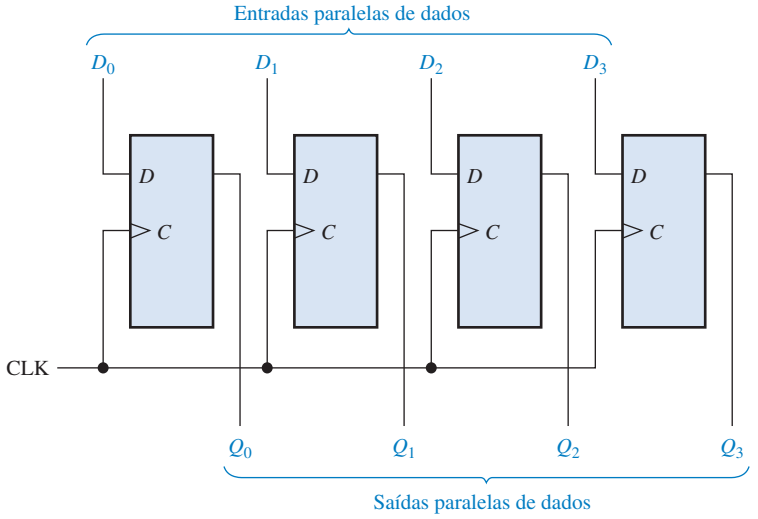
9-5 REGISTRADORES DE DESLOCAMENTO COM ENTRADA PARALELA/SAÍDA PARALELA

A entrada paralela de dados foi descrita na Seção 9–4 e a saída paralela de dados também foi discutida anteriormente. O registrador com entrada paralela/saída paralela emprega os dois métodos. Imediatamente em seguida à entrada de todos os bits de dados, esses aparecem nas saídas em paralelo.

Ao final do estudo desta seção você deverá ser capaz de:

- Discutir o registrador de deslocamento de 4 bit com entrada paralela/saída paralela 74HC195
- Desenvolver e analisar diagramas de temporização para registradores com entrada paralela/saída paralela

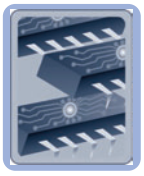
A Figura 9–16 mostra um registrador com entrada paralela/saída paralela.



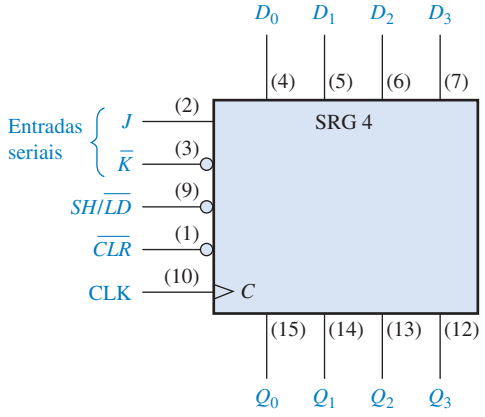
▲ **FIGURA 9–16**

Registrador com entrada paralela/saída paralela.

REGISTRADOR DE DESLOCAMENTO DE 4 BITS COM ACESSO PARALELO (74HC195)



O CI 74HC195 pode ser usado para operar com entrada paralela/saída paralela. Como ele também tem uma entrada serial, pode ser usado para operar com entrada serial/saída paralela. Ele ainda pode ser usado para operar com entrada paralela/saída serial usando Q_3 como saída. O símbolo lógico em bloco típico é mostrado na Figura 9–17.

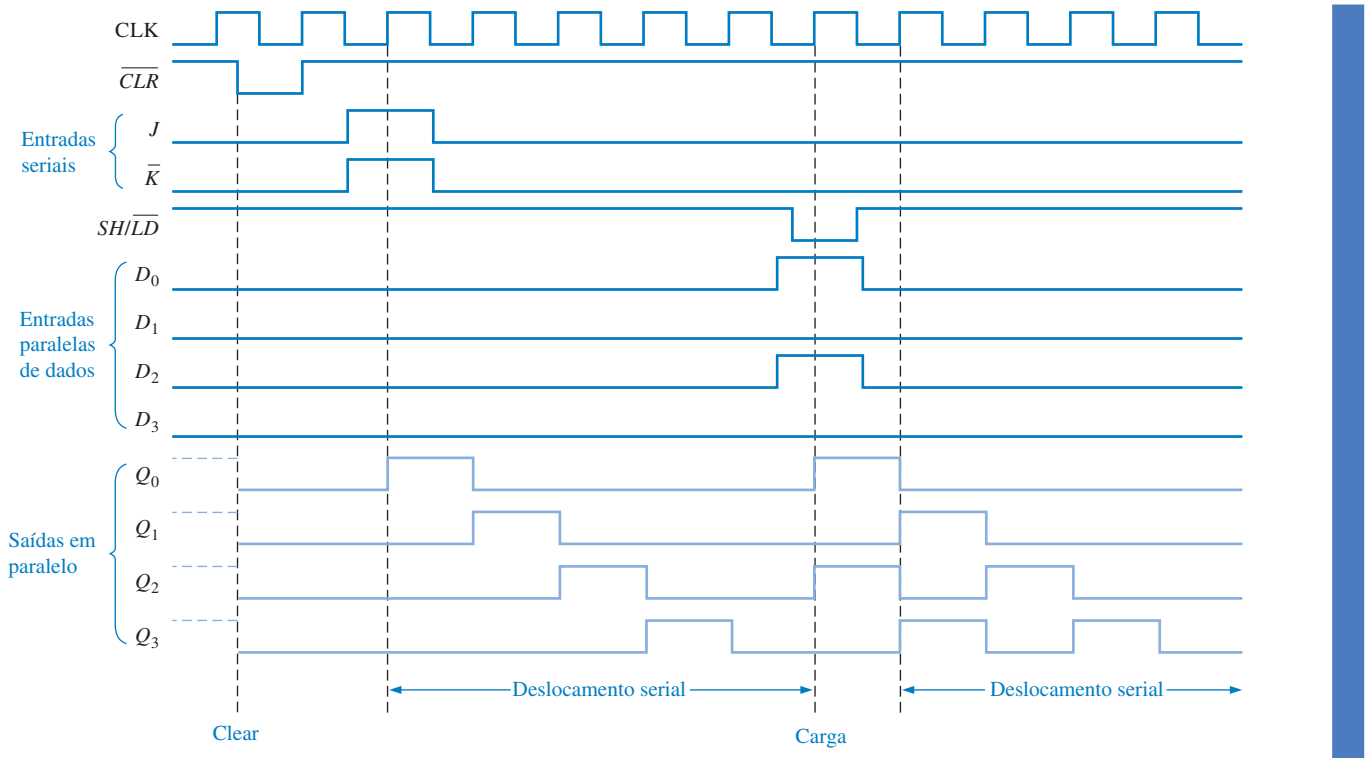


▶ **FIGURA 9–17**

O CI registrador de deslocamento de 4 bits com acesso paralelo 74HC195.

Quando a entrada $SHIFT/\overline{LOAD}$ ($SH/\overline{LD}$) é nível BAIXO, os dados nas entradas paralelas são inseridos de forma síncrona na transição positiva do clock. Quando $SH/\overline{LD}$ for nível ALTO, os dados armazenados são deslocados à direita (de Q_0 para Q_3) sincronizados pelo clock. As entradas J e $\overline{K}$ são as entradas seriais de dados para o primeiro estágio do registrador (Q_0); Q_3 pode ser usada para saída serial de dados. A entrada de clear, que é ativa em nível BAIXO, é assíncrona.

O diagrama de temporização da Figura 9-18 ilustra a operação desse registrador.



▲ **FIGURA 9–18**

Amostra de um diagrama de temporização para o CI registrador de deslocamento 74HC195.

SEÇÃO 9–5 REVISÃO

1. Na Figura 9–16, $D_0 = 1$, $D_1 = 0$, $D_2 = 0$ e $D_3 = 1$. Após três pulsos de clock, quais são os dados nas saídas?
2. Para o CI 74HC195, $SH/\overline{LD} = 1$, $J = 1$, e $\overline{K} = 1$. Qual é o estado da saída Q_0 após um pulso de clock?

9-6 REGISTRADORES DE DESLOCAMENTO BIDIRECIONAIS

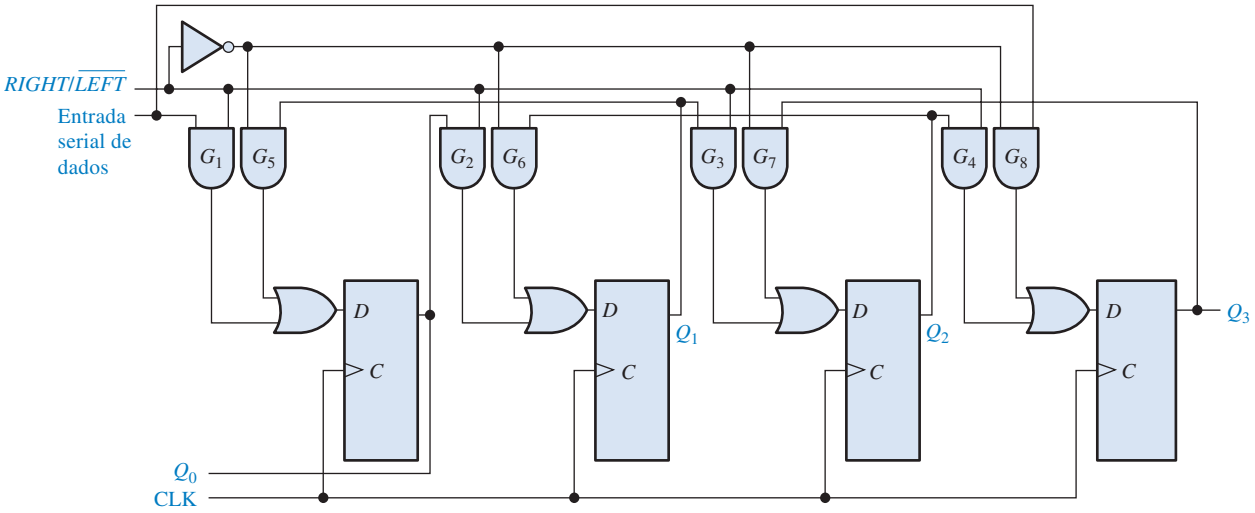
Um registrador de deslocamento bidirecional é aquele no qual os dados podem ser deslocados para a esquerda ou para a direita. Isso pode ser implementado usando lógica de controle que habilita a transferência do bit de dado de um estágio para o próximo estágio à direita ou à esquerda, dependendo do nível lógico na linha de controle.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a operação de um registrador de deslocamento bidirecional ■ Discutir o CI registrador de deslocamento bidirecional universal de 4 bits 74HC194 ■ Desenvolver e analisar diagramas de temporização para registradores de deslocamento bidirecionais

Um registrador de deslocamento **bidirecional** de 4 bits é mostrado na Figura 9–19. Um nível ALTO na entrada de controle $RIGHT/\overline{LEFT}$ permite que os bits de dados inseridos no registrador sejam deslocados para a direita, e um nível BAIXO permite que os bits de dados sejam deslocados para a esquerda. Um exame na lógica de controle torna a operação aparente. Quando a entrada de controle $RIGHT/\overline{LEFT}$ for nível ALTO, as portas de G_1 a G_4 são habilitadas e o estado da

saída Q de cada flip-flop passa para a entrada D do flip-flop *seguinte*. Quando um pulso de clock ocorre, os bits de dados são deslocados uma posição para a *direita*. Quando a entrada de controle $RIGHT/\overline{LEFT}$ for nível BAIXO, as portas de G_5 a G_8 são habilitadas sendo que a saída Q de cada flip-flop passa para a entrada D do flip-flop *precedente*. Quando um pulso de clock ocorre, os bits de dados são então deslocados uma posição à *esquerda*.

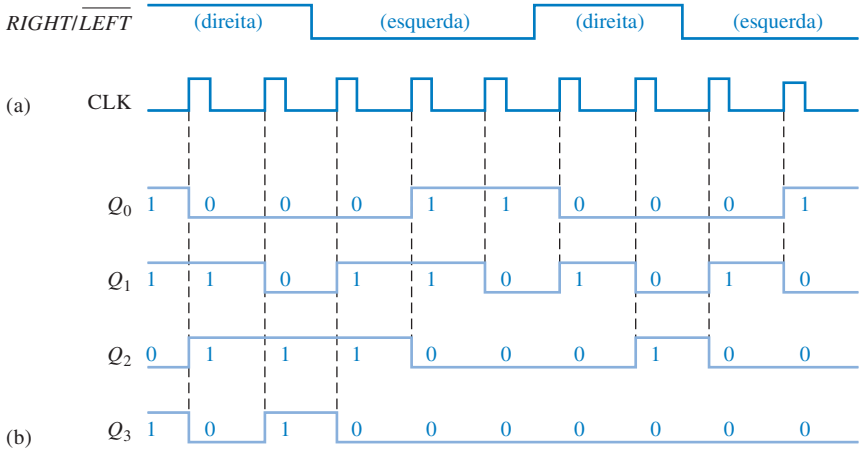


▲ **FIGURA 9–19**

Registrador de deslocamento bidirecional de 4 bits. Abra o arquivo F09-19 para verificar a operação.

EXEMPLO 9–4

Determine o estado do registrador de deslocamento mostrado na Figura 9–19 após cada pulso de clock considerando a forma de onda da entrada $RIGHT/\overline{LEFT}$ mostrada na Figura 9–20(a). Considere que $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 1$ e que a linha de entrada serial de dados esteja em nível BAIXO.

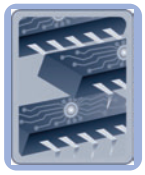


▶ **FIGURA 9–20**

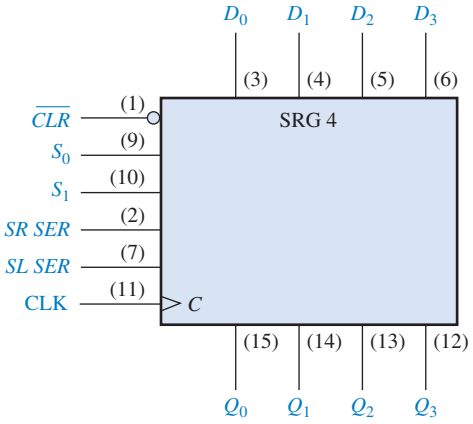
Solução Veja a Figura 9–20(b).

Problema relacionado Inverta a forma de onda $RIGHT/\overline{LEFT}$ e determine o estado do registrador de deslocamento dado na Figura 9–19 após cada pulso de clock.

REGISTRADOR DE DESLOCAMENTO BIDIRECIONAL UNIVERSAL DE 4 BITS (74HC194)

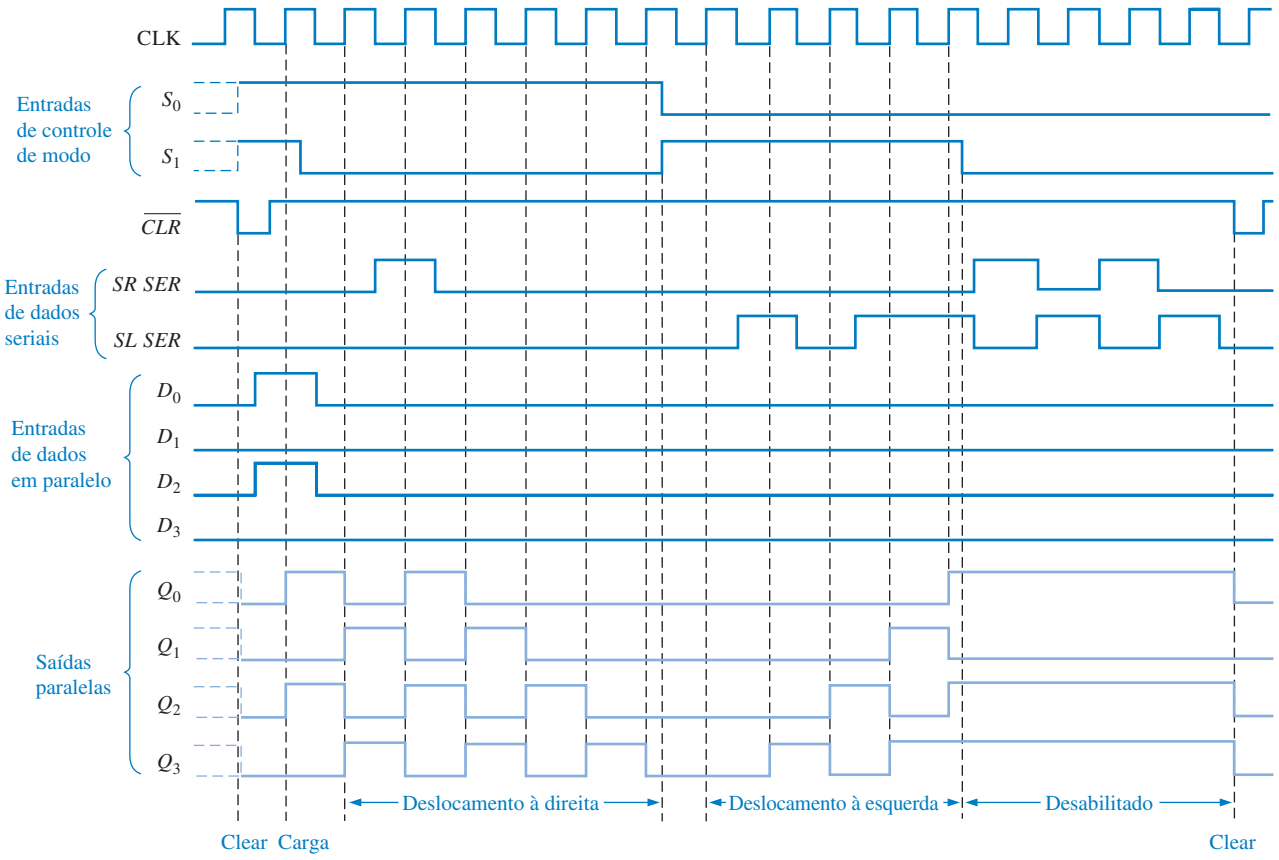


O CI 74HC194 é um exemplo de um registrador de deslocamento bidirecional universal. Um **registrador de deslocamento universal** tem capacidade de entrada e saída serial e paralela de dados. O símbolo lógico em bloco é mostrado na Figura 9–21 e um diagrama de temporização é mostrado na Figura 9–22.



◀ **FIGURA 9–21**

O CI registrador de deslocamento bidirecional universal de 4 bits 74HC194.



▲ **FIGURA 9–22**

Amostra do diagrama de temporização para o CI registrador de deslocamento 74HC194.

A carga paralela, que é sincronizada com a transição positiva do clock, é realizada aplicando-se os quatro bits de dados nas entradas paralelas e nível ALTO nas entradas S_0 e S_1. O deslocamento à direita é realizado de forma síncrona com a borda positiva do clock quando S_0 for nível ALTO e S_1 for nível BAIXO. Os dados seriais nesse modo são inseridos na entrada serial de deslocamento à direita (*SR SER*). Quando S_0 for nível BAIXO e S_1 for nível ALTO, os bits de dados são deslocados à esquerda de forma síncrona com o clock, sendo os novos dados inseridos na entrada serial à esquerda (*SL SER*). A entrada *SR SER* vai para o estágio Q_0 e *SL SER* vai para o estágio Q_3.

SEÇÃO 9–6 REVISÃO

1. Considere que o registrador de deslocamento bidirecional de 4 bits mostrado na Figura 9–19 tenha os seguintes conteúdos: $Q_0 = 1$, $Q_1 = 1$, $Q_2 = 0$ e $Q_3 = 0$. Existe um nível 1 na linha de entrada de dados seriais. Se a entrada *RIGHT*/$\overline{LEFT}$ for nível ALTO durante três pulsos de clock e nível BAIXO durante mais dois pulsos de clock, quais são os conteúdos após o quinto pulso de clock?

9-7 REGISTRADORES DE DESLOCAMENTO COMO CONTADORES

Um registrador de deslocamento usado como contador é basicamente um registrador de deslocamento com a saída serial conectada de volta à entrada serial para produzir seqüências especiais. Esses dispositivos são freqüentemente classificados como contadores porque exibem uma seqüência de dados específica. Dois dos tipos mais comuns de registradores de deslocamento usados como contadores, o contador Johnson e o contador em anel, são apresentados neste capítulo.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir como um registrador de deslocamento usado como contador difere de um registrador de deslocamento básico ■ Explicar a operação de um contador Johnson ■ Especificar uma seqüência Johnson para qualquer número de bits ■ Explicar a operação de um contador em anel e determinar a seqüência de qualquer contador específico

Contador Johnson

Em um **contador Johnson**, o complemento da saída do último flip-flop é conectado de volta na entrada *D* do primeiro flip-flop (isso pode ser feito também com outros tipos de flip-flops). Esse arranjo com realimentação produz uma seqüência característica de estados, conforme mostra a Tabela 9–1 para um dispositivo de 4 bits e na Tabela 9–2 para um dispositivo de 5 bits. Observe que a seqüência de 4 bits tem um total de 10 estados. Em geral, um contador Johnson produz um módulo de $2n$, onde n é o número de estágios no contador.

As implementações de contadores Johnson de 4 e 5 estágios são mostradas na Figura 9–23. A implementação de um contador Johnson é muito simples independentemente do número de estágios. A saída Q de cada estágio é conectada na entrada D do próximo estágio (considerando que flip-flops D sejam usados). A única exceção é que a saída $\overline{Q}$ do último estágio é conectada de volta na entrada D do primeiro estágio. Conforme mostra as seqüências nas Tabelas 9–1 e 9–2, o contador é "preenchido" com 1s da esquerda para a direita e, em seguida, é "preenchido" com 0s novamente.

TABELA 9–1

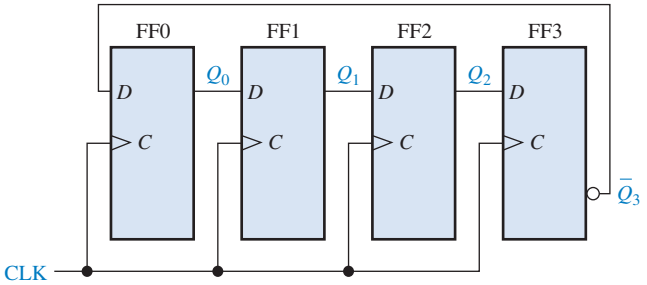
Seqüência Johnson de 4 bits

PULSO DE CLOCK	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

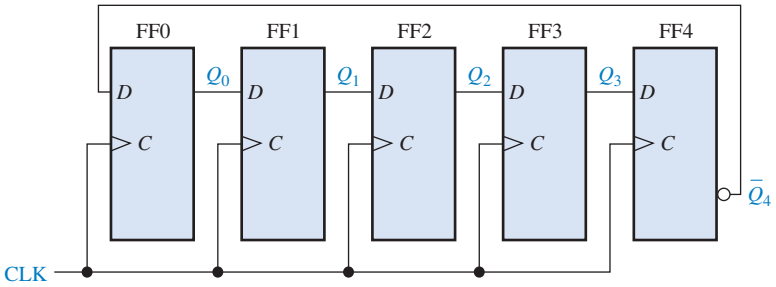
TABELA 9–2

Seqüência Johnson de 5 bits

PULSO DE CLOCK	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1



(a) Contador Johnson de 4 bits

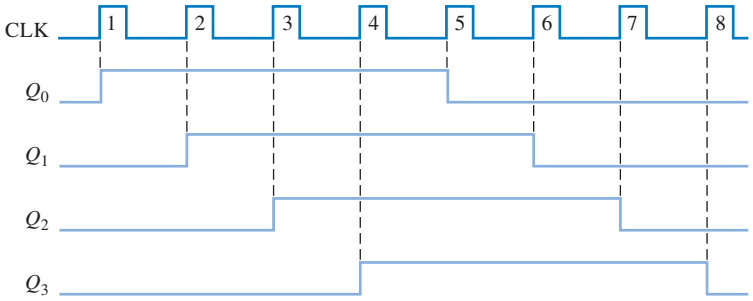


(b) Contador Johnson de 5 bits

FIGURA 9–23

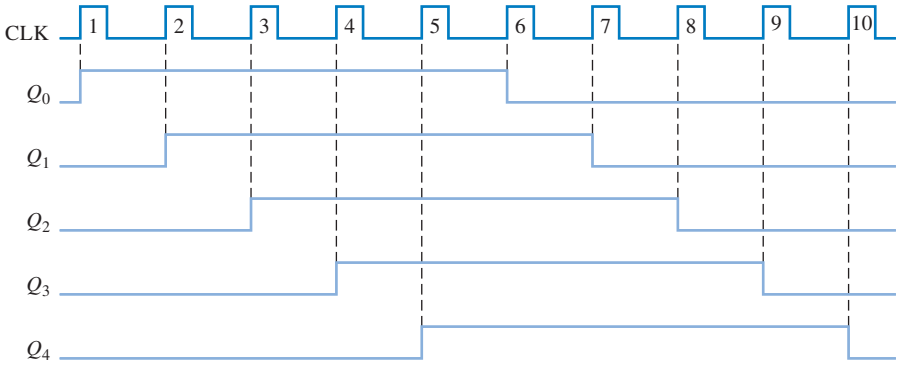
Contadores Johnson de 4 e 5 bits.

Diagramas das operações de temporização dos contadores de 4 e 5 bits são mostrados nas Figuras 9-24 e 9-25, respectivamente.



▶ **FIGURA 9–24**

Seqüência de temporização para um contador Johnson de 4 bits.



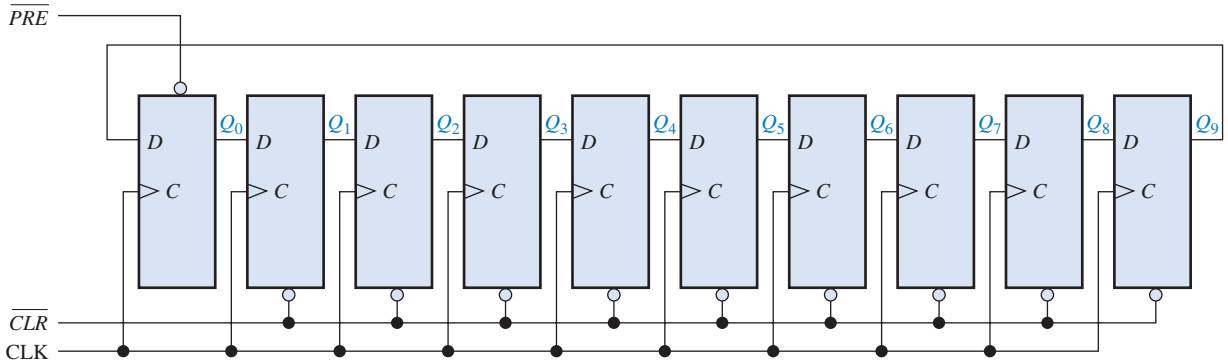
▶ **FIGURA 9–25**

Seqüência de temporização para um contador Johnson de 5 bits.

Contador em Anel

O **contador em anel** utiliza um flip-flop para cada estado em sua seqüência. Ele tem a vantagem de não necessitar de portas de decodificação. No caso de um contador em anel de 10 bits, existe uma única saída para cada dígito decimal.

A Figura 9–26 mostra um diagrama lógico para um contador em anel de 10 bits. A seqüência para esse contador em anel é dada na Tabela 9–3. Inicialmente, um nível 1 está presente no primeiro flip-flop e o restante dos flip-flops estão resetados. Observe que as conexões entre estágios são as mesmas que para um contador Johnson, exceto que a saída Q em vez de $\overline{Q}$ é realimentada a partir do último estágio. As dez saídas do contador indicam diretamente a contagem decimal dos pulsos de clock. Por exemplo, um nível 1 em Q_0 representa zero, um nível 1 em Q_1 representa um,



▲ **FIGURA 9–26**

Um contador em anel de 10 bits. Abra o arquivo F09-26 para verificar a operação.

TABELA 9–3

Seqüência de um contador em anel de 10 bits

PULSO DE CLOCK	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

um nível 1 em Q_2 representa dois, um nível 1 em Q_3 representa três, e assim por diante. Temos que verificar que um nível 1 é sempre mantido no contador e simplesmente desloca "em torno do anel", avançando um estágio para cada pulso de clock.

Seqüências modificadas podem ser conseguidas colocando mais que um único 1 no contador, conforme ilustrado no Exemplo 9–5.

EXEMPLO 9–5

Se um contador em anel de 10 bits similar ao da Figura 9–26 tem o estado inicial 1010000000, determine a forma de onda de cada uma das saídas Q.

Solução Veja a Figura 9–27.

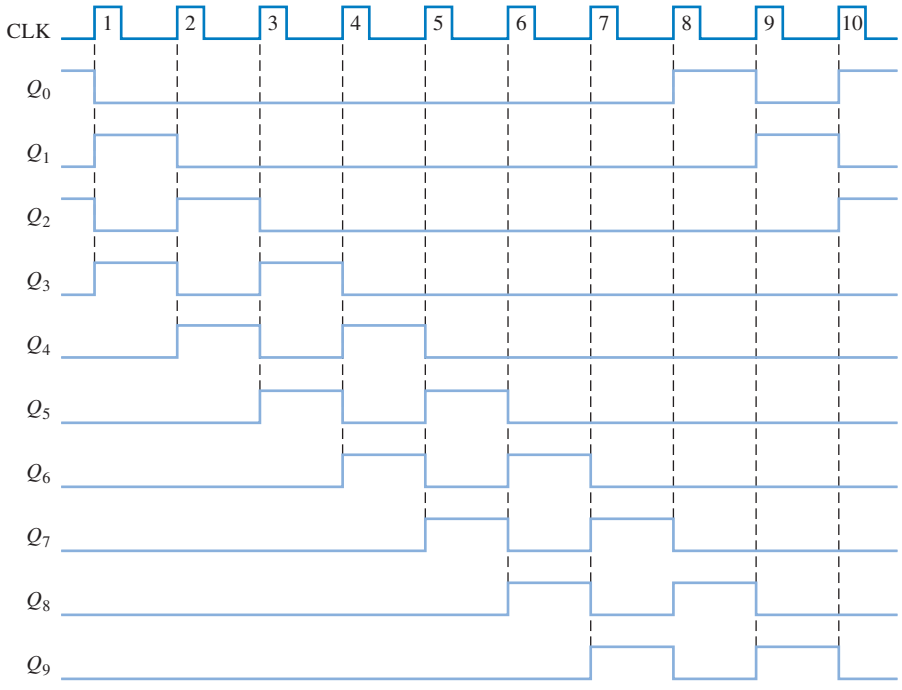


FIGURA 9–27

Problema relacionado Se um contador em anel de 10 bits tem um estado inicial 0101001111, determine a forma de onda para cada saída Q.

SEÇÃO 9–7 REVISÃO

1. Quantos estados existem na seqüência de um contador Johnson de 8 bits?
2. Escreva a seqüência de estados para um contador Johnson começando com 000.

9-8 APLICAÇÕES DE REGISTRADORES DE DESLOCAMENTO

Registradores de deslocamento são encontrados em muitos tipos de aplicações, das quais algumas são apresentadas nesta seção.

Ao final do estudo desta seção você deverá ser capaz de:

■ Usar um registrador de deslocamento para gerar uma atraso de tempo ■ Implementar uma seqüência para contador em anel especificada usando um CI registrador de deslocamento 74HC195 ■ Discutir como registradores de deslocamento são usados para converter dados do formato serial para paralelo ■ Definir UART ■ Explicar a operação de um codificador de teclado e como registradores são usados nessa aplicação

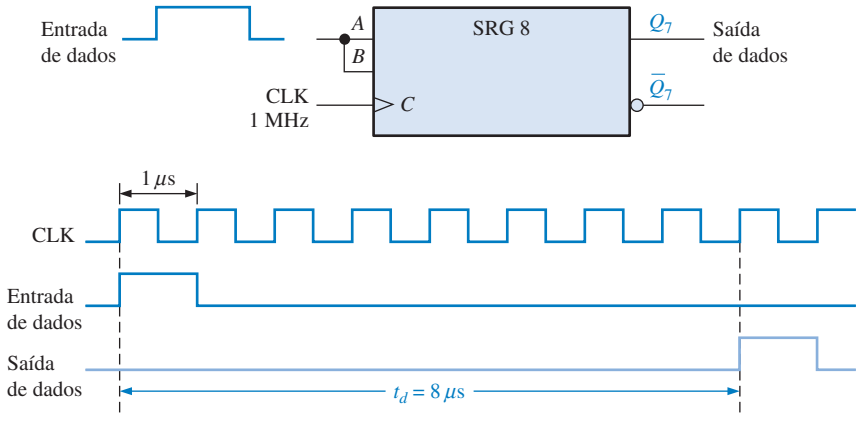
NOTA: COMPUTAÇÃO

Os registradores de propósito geral no processador Pentium são registradores de 32 bits que podem ser usados para armazenamento temporário de dados bem como usos específicos. Quatro desses registradores são apresentados a seguir. O *acumulador* (EAX) é usado principalmente para armazenamento temporário de dados e operandos de instruções. O *registrador base* (EBX) é usado para armazenar um valor temporariamente. O *registrador de contagem* (ECX) é usado principalmente para determinar o número de repetições num certo loop, fluxo, deslocamento ou rotação. O *registrador de dados* (EDX) normalmente é usado para o armazenamento temporário de dados.

Atraso de Tempo

O registrador de deslocamento com entrada serial/saída serial pode ser usado para proporcionar um atraso da entrada para a saída que é uma função do número de estágios (n) no registrador e a freqüência do clock.

Quando um pulso de dado é aplicado na entrada serial como mostra a Figura 9–28 (A e B inter-conectadas), ele entra no primeiro estágio na borda de disparo do pulso de clock. Ele é então deslocado de estágio para estágio a cada pulso de clock sucessivo até que apareça na saída serial com um atraso de n períodos de clock. Essa operação de atraso de tempo é ilustrada na Figura 9–28, na qual um registrador de deslocamento de 8 bits com entrada serial/saída serial é usado com um clock de 1 MHz para obter um atraso de tempo (t_d) de 8μs ($8 \times 1\mu s$). Esse tempo pode ser ajustado para cima ou para baixo alterando a freqüência de clock. O atraso de tempo também pode ser aumentado fazendo a conexão em cascata de registradores de deslocamento, e diminuído obtendo a saída a partir de estágios sucessivamente mais próximos do primeiro, caso as saídas desses estágios estejam acessíveis, conforme ilustra o Exemplo 9–6.

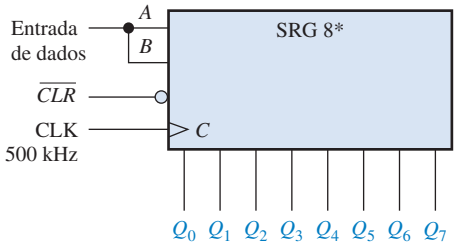


▲ **FIGURA 9–28**

Um registrador de deslocamento usado como um dispositivos de atraso de tempo.

EXEMPLO 9–6

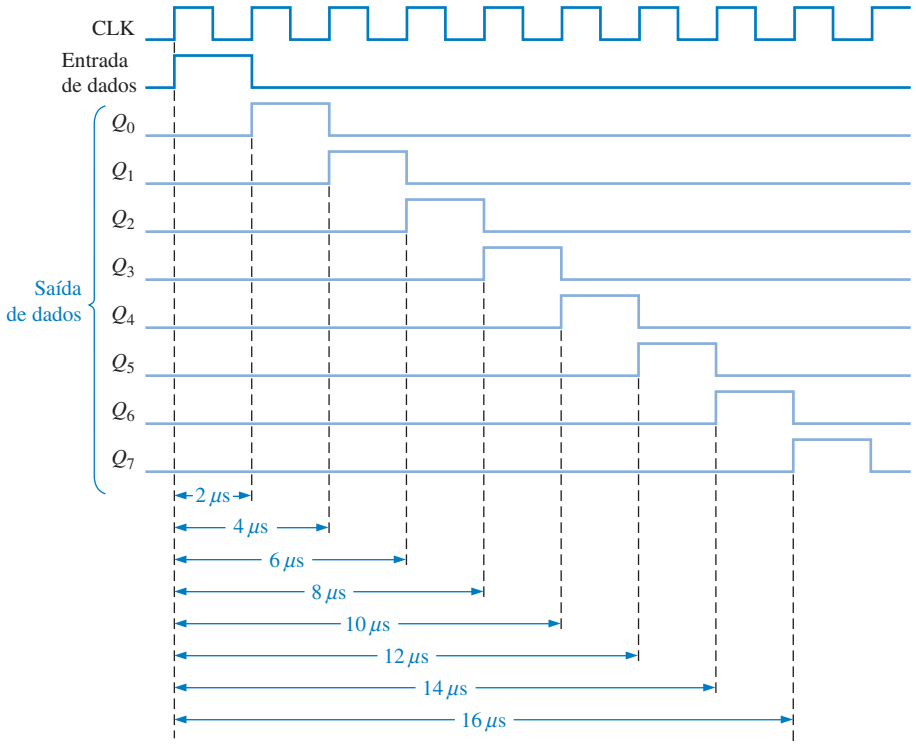
Determine o tempo de atraso entre a entrada serial e cada saída no circuito mostrado na Figura 9–29. Mostre um diagrama de temporização para ilustrar.



▶ **FIGURA 9–29**

* Deslocamentos de dados de Q_0 para Q_7.

Solução O período de clock é 2μs, Portanto, o atraso de tempo pode ser aumentado ou diminuído em incrementos de 2μs a partir de um mínimo de 2μs até um máximo de 16μs, conforme ilustrado na Figura 9–30.

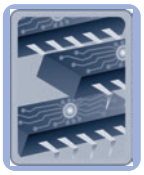


▲ **FIGURA 9–30**

Diagrama de temporização mostrando os atrasos de tempo do registrador dado na Figura 9–29.

Problema relacionado Determine a freqüência de clock necessária para obter um atraso de tempo de 24μs para a saída Q_7 na Figura 9–29.

UM CONTADOR EM ANEL USANDO O CI REGISTRADOR DE DESLOCAMENTO 74HC195



Se a saída for conectada de volta para a entrada serial, um registrador de deslocamento pode ser usado como um contador em anel. A Figura 9–31 ilustra essa aplicação com um CI registrador de deslocamento de 4 bits 74HC195.

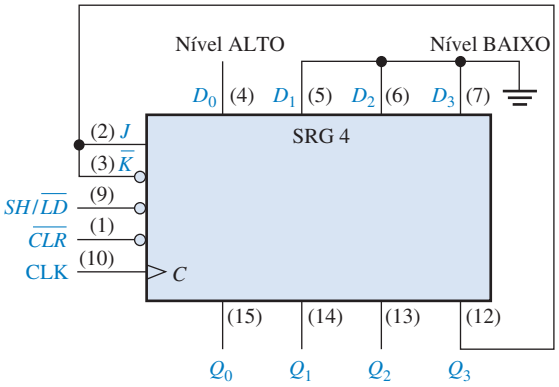


FIGURA 9–31

CI 74HC195 conectado como um contador em anel.

Inicialmente, a seqüência de bits 1000 (ou qualquer outra seqüência) pode ser carregada (*presete*) no contador de forma síncrona aplicando a seqüência de bits nas entradas paralelas, colocando a entrada $SH/\overline{LD}$ em nível BAIXO e aplicando um pulso de clock. Após essa inicialização, o nível 1 continua circulando pelo contador em anel, conforme o diagrama de temporização mostrado na Figura 9–23.

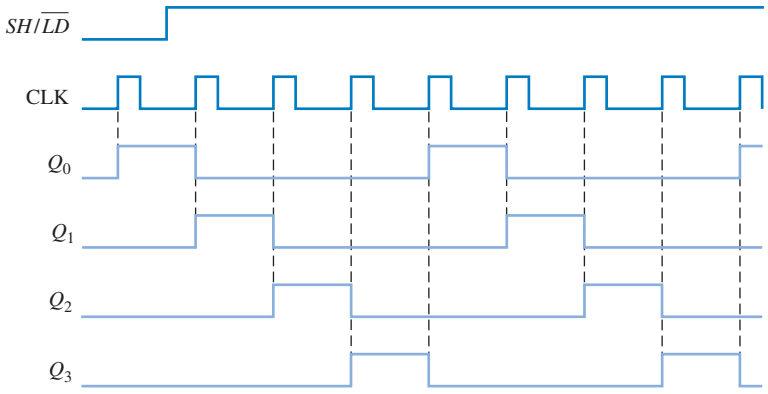


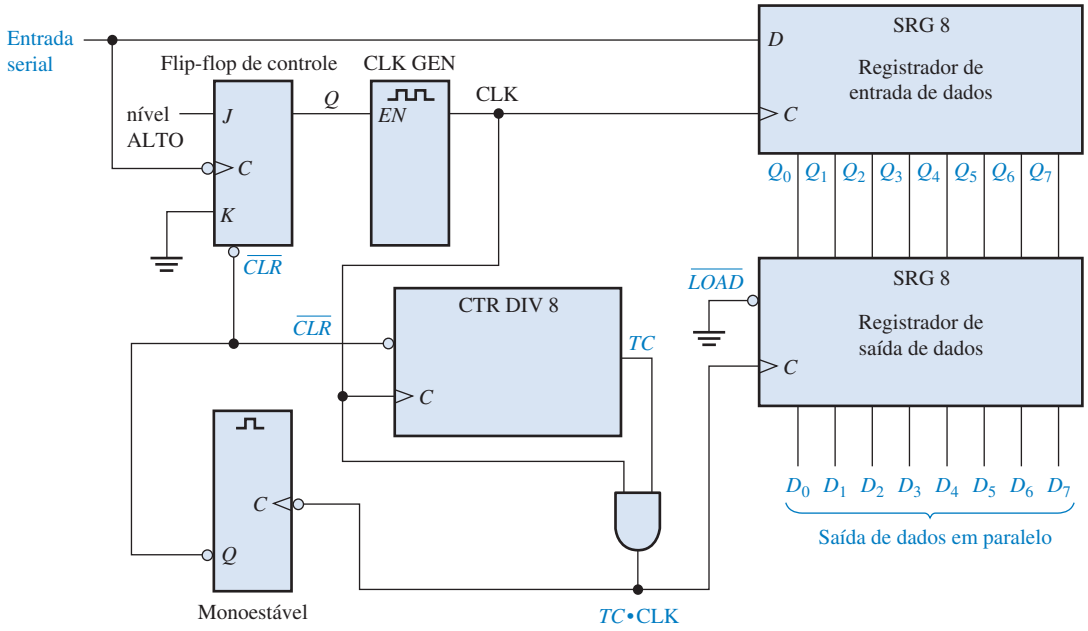
FIGURA 9–32

Diagrama de temporização mostrando dois ciclos completos do contador em anel dado na Figura 9–31 quando inicializado com 1000.

Conversor de Dados de Serial para Paralelo

A transmissão serial de dados de um sistema digital para outro é normalmente usada para reduzir o número de fios na linha de transmissão. Por exemplo, oito bits podem ser enviados de forma serial ao longo de um fio, mas são necessários oito fios para enviar o mesmo dado em paralelo.

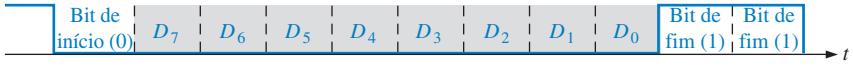
Um computador ou um sistema baseado em microprocessador normalmente necessita receber dados que estejam no formato paralelo, dessa forma é necessário converter de serial para paralelo. Um conversor de dados de serial para paralelo simplificado, no qual dois tipos de registradores de deslocamento são usados, é mostrado na Figura 9–33.



▲ **FIGURA 9–33**

Diagrama lógico simplificado de um conversor de serial para paralelo.

Para ilustrar a operação desse conversor de serial para paralelo, é usado o formato de dado serial mostrado na Figura 9–34. O primeiro bit (bit de início – *start bit*) é sempre 0 e sempre começa com uma transição de nível ALTO para nível BAIXO. Os próximos oito bits (D_7 a D_0) são os bits de dados (um dos bits pode ser a paridade) e os dois últimos bits (bits de fim – *stop bits*) são sempre nível 1. Quando nenhum dado estiver sendo enviado, existe o nível 1 continuamente na linha serial de dados.

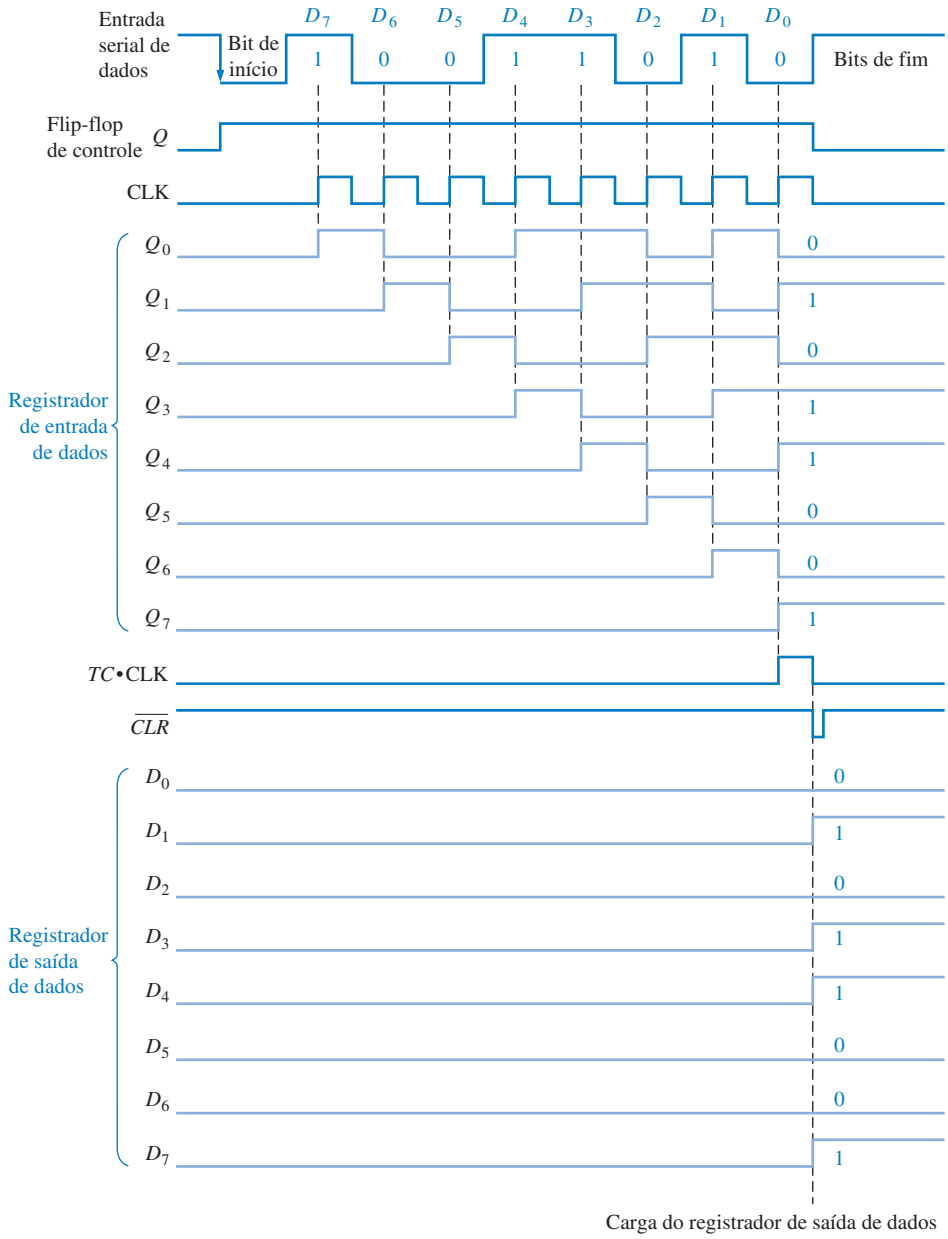


▲ **FIGURA 9–34**

Formato serial de dados.

A transição de nível ALTO para nível BAIXO do bit de início seta o flip-flop de controle, o qual habilita o gerador de clock. Após um tempo de atraso fixo, o gerador de clock começa a produzir uma forma de onda de pulsos, que é aplicada na entrada de dados do registrador e no contador divisor por 8. O clock tem uma freqüência precisamente igual a dos dados na entrada serial e o primeiro pulso de clock após o bit de início ocorre durante o primeiro bit de dado.

O diagrama de temporização na Figura 9–35 ilustra a seguinte operação básica: Os oito bits de dados (D_7 a D_0) são deslocados de forma serial para dentro do registrador de entrada de dados. Após os oito pulsos de clock, uma operação AND entre a transição do nível ALTO para o nível BAIXO na saída fim de contagem (*TC*) do contador e o clock (*TC*·CLK) carrega os oito bits que estão no registrador de entrada de dados para dentro do registrador de saída de dados. Essa mesma transição também dispara o monoestável, o qual produz um pulso de curta duração para resetar o contador e o flip-flop que por sua vez desabilita o gerador de clock. O sistema agora está pronto para o próximo grupo de onze bits e ele espera pela próxima transição do nível ALTO para o nível BAIXO do bit de início.



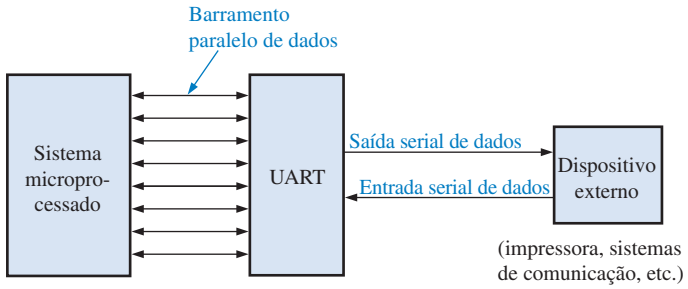
▲ **FIGURA 9–35**

Diagrama de temporização ilustrando a operação do conversor de dados de serial para paralelo dado na Figura 9–33.

A inversão do processo citado anteriormente pode ser realizada por um conversor de paralelo para serial. Entretanto, como o formato serial de dados tem que ser produzido, requisitos adicionais têm que ser considerados.

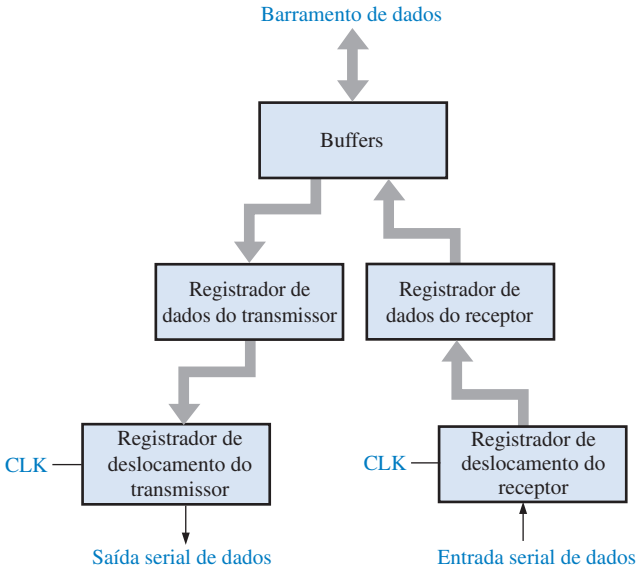
Transmissor/Receptor Assíncrono Universal (UART)

Conforme mencionado, os computadores e sistemas baseados em microprocessadores normalmente enviam e recebem dados no formato paralelo. Freqüentemente, esses sistemas têm que se comunicar com dispositivos externos que enviam e/ou recebem dados no formato serial. Um dispositivo de interfaceamento usado para realizar essas conversões é o UART (*Universal Asynchronous Receiver Transmitter* – Transmissor/Receptor Assíncrono Universal). A Figura 9–36 ilustra o UART numa aplicação de um sistema geral baseado em microprocessador.



◀ **FIGURA 9–36**
Interface UART.

Um dispositivo UART inclui um conversor de dados de serial para paralelo tal como discutimos e um conversor de paralelo para serial, como mostra a Figura 9–37. O barramento de dados é basicamente um conjunto de condutores em paralelo ao longo dos quais os dados se movimentam entre o UART e o sistema microprocessado. Buffers fazem a interface entre registradores de dados e o barramento de dados.



◀ **FIGURA 9–37**
Diagrama em bloco básico de um dispositivo UART.

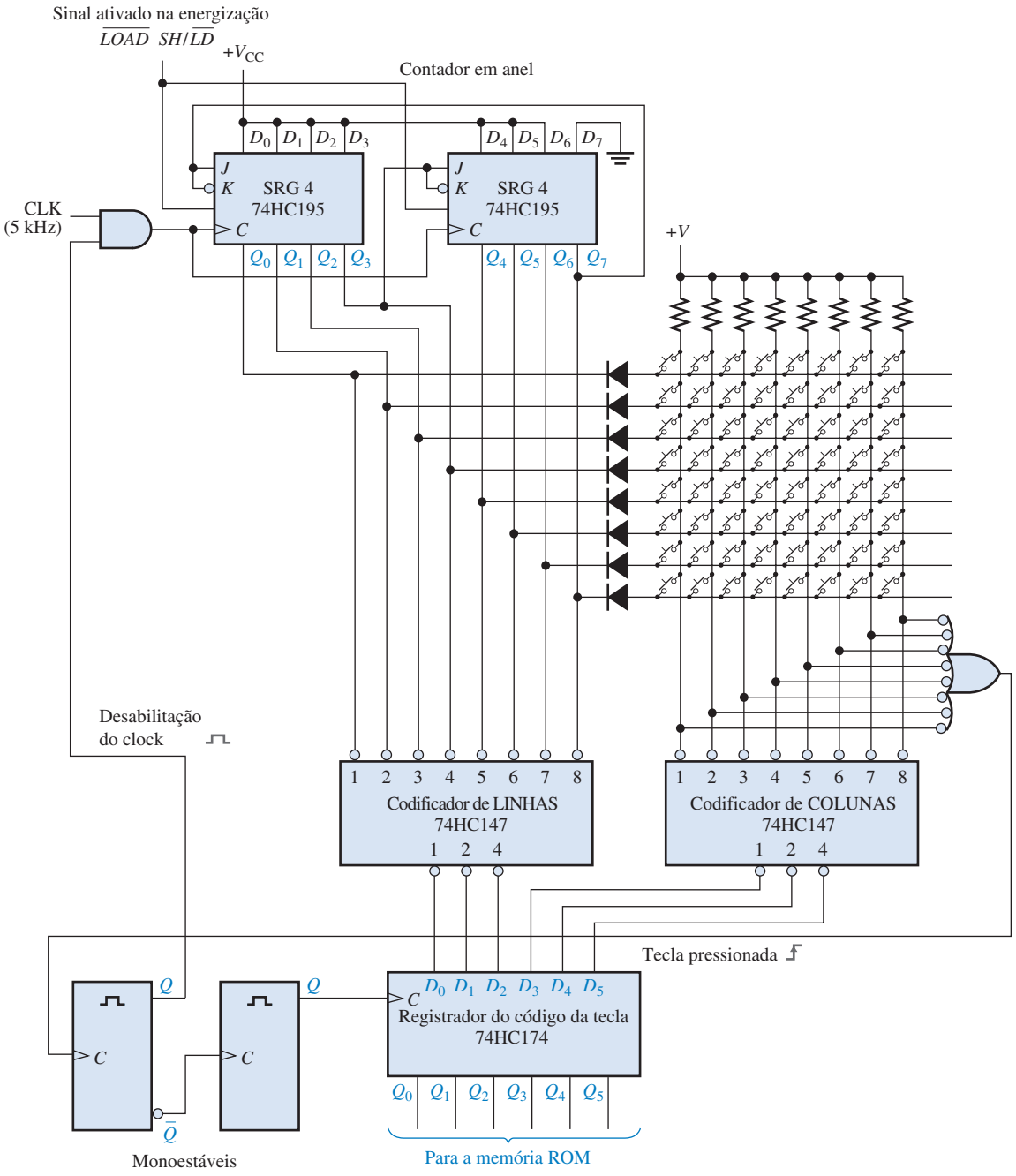
O UART recebe os dados no formato serial, converte os dados para o formato paralelo e os coloca no barramento de dados. O UART também aceita dados em paralelo a partir do barramento de dados, converte os dados para o formato serial e os transmite para um dispositivo externo.

Codificador de Teclado

O codificador de teclado é um bom exemplo da aplicação de um registrador de deslocamento usado como um contador em anel em conjunto com outros dispositivos. Lembre-se que um codificador de teclado de computador simplificado, sem armazenamento de dados, foi apresentado no Capítulo 6.

A Figura 9–38 mostra um codificador de teclado simplificado que codifica uma tecla pressionada numa matriz de 64 teclas organizada em oito linhas e oito colunas. Dois CIs registradores de deslocamento de 4 bits 74HC195 são conectados como um contador em anel com uma seqüência de bits fixa contendo sete 1s e um 0 presetado no momento em que o sistema é energizado. Dois CIs codificadores de prioridade 74HC147 (apresentado no Capítulo 6) são usados como um codificador de oito para três linhas (9 entradas em nível ALTO, 8 saídas não usadas) para codificar LINHAS e COLUNAS da matriz do teclado. O CI 74HC174 (seis flip-flops) é usado como um registrador com entrada paralela/saída paralela no qual o código LINHA/COLUNA dos codificadores de prioridade é armazenado.

A operação básica do codificador de teclado dado na Figura 9–38 é a seguinte: o contador em anel "escaneia" as linhas em busca de uma tecla acionada conforme o sinal de clock desloca o ní-



▲ **FIGURA 9–38**

Circuito simplificado de um codificador de teclado.

vel 0 ao longo do contador numa freqüência de 5 kHz. O nível 0 (BAIXO) é aplicado sequencialmente em cada LINHA, enquanto todas as outras LINHAS ficam em nível ALTO. Todas as LINHAS estão conectadas nas entradas do codificador de LINHAS, assim a saída de 3 bits do codificador de LINHAS em qualquer instante é a representação binária da LINHA que está em nível BAIXO. Quando uma tecla for pressionada, uma COLUNA é conectada a uma LINHA. Quando a LINHA for colocada em nível BAIXO pelo contador em anel, aquela coluna em particular também vai para nível BAIXO. O codificador de COLUNAS produz uma saída binária correspondente à COLUNA na qual tem uma tecla pressionada. O código de LINHA de 3 bits juntamente com o código de COLUNA de 3 bits identifica exclusivamente a tecla pressionada. Esse código de 6

bits é aplicado nas entradas do registrador de código da tecla. Quando uma tecla é pressionada, os dois monoestáveis produzem um pulso de clock atrasado para a operação de carga paralela do código de 6 bits no registrador de código da tecla. Esse atraso dá um tempo para que o repique da chave termine. Além disso, a saída do primeiro monoestável inibe o contador em anel evitando um escaneamento enquanto o dado está sendo carregado no registrador do código da tecla.

O código de 6 bits no registrador do código da tecla é agora aplicado na memória ROM (*read-only memory*) para ser convertido para um código alfanumérico apropriado que o caractere do teclado. As memórias ROM são estudadas no Capítulo 10.

SEÇÃO 9–8 REVISÃO

1. No codificador de teclado, quantas vezes por segundo o contador em anel escaneia o teclado?
2. Qual é o código de 6 bits LINHA/COLUNA (código da tecla) para a linha superior e a coluna mais à esquerda no codificador de teclado?
3. Qual é a finalidade dos diodos no codificador de teclado? Qual é a finalidade dos resistores?

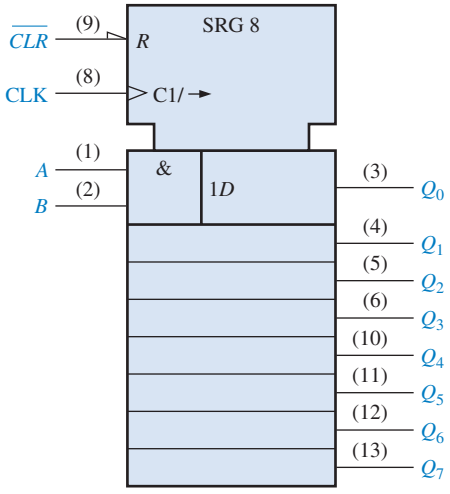
9-9 SÍMBOLOS LÓGICOS COM NOTAÇÃO DE DEPENDÊNCIA

São apresentados dois exemplos de símbolos do padrão 91-1984 da ANSI/IEEE com notação de dependência para registradores de deslocamento.

Ao final do estudo desta seção você deverá ser capaz de:

■ Entender e interpretar os símbolos lógicos com notação de dependência para os CIs registradores 74HC164 e 74HC194

O símbolo lógico para o CI registrador de deslocamento com saída paralela de 8 bits 74HC164 é mostrado na Figura 9–39. As entradas de controle comum são mostradas no bloco com um entalhe. A entrada clear ($\overline{CLR}$) está indicada por um *R* (de RESET) dentro do bloco. Como não existe prefixo de dependência para relacionar *R* com o clock (C_1), a função clear é assíncrona. O símbolo da seta para a direita após C_1 indica que o dado flui de Q_0 para Q_7. As entradas *A* e *B* passam por uma função AND conforme indicado pelo símbolo interno da AND (&), para prover uma entrada de dados síncrona, 1*D*, para o primeiro estágio (Q_0). Observe a dependência de *D* em *C*, conforme indicado pelo sufixo 1 em *C* e o prefixo 1 em *D*.



◀ **FIGURA 9–39**
Símbolo lógico para o CI 74HC164.

A Figura 9–40 é o símbolo lógico para o registrador de deslocamento bidirecional universal de 4 bits 74HC194. Começando na parte superior esquerda do bloco de controle, observe que a entrada $\overline{CLR}$ é ativa em nível BAIXO e assíncrona (sem prefixo de relação com *C*). As entradas S_0 e S_1

são entradas de modo que determinam os modos de operação para deslocamento à direita, deslocamento à esquerda e carga paralela, conforme mostra a indicação de dependência após o *M*. A indicação $\frac{0}{3}$ representa os estados binários de 0, 1, 2 e 3 nas entradas S_0 e S_1. Quando um desses dígitos é usado como prefixo para uma outra entrada, uma dependência é estabelecida. O símbolo 1→/2← na entrada de clock indica o seguinte: 1→ indica que um deslocamento à direita (Q_0 em direção a Q_3) ocorre quando as entradas de modo (S_0, S_1) estão no estado binário 1 ($S_0 = 1$, $S_1 = 0$), 2← indica que um deslocamento à esquerda (Q_3 em direção a Q_0) ocorre quando as entradas de modo estão no estado binário 2 ($S_0 = 0$, $S_1 = 1$). A entrada serial de deslocamento à direita (SR SER) é dependente do modo e dependente do clock, conforme indicado por 1, 4*D*. As entradas paralelas (D_0, D_1, D_2 e D_3) são todas dependentes do modo (o prefixo 3 indica o modo de carga paralela) e dependentes de clock, conforme indicado por 2, 4*D*. A entrada serial de deslocamento à esquerda (*SL SER*) é dependente do modo e dependente do clock, conforme indicado por 2, 4*D*.

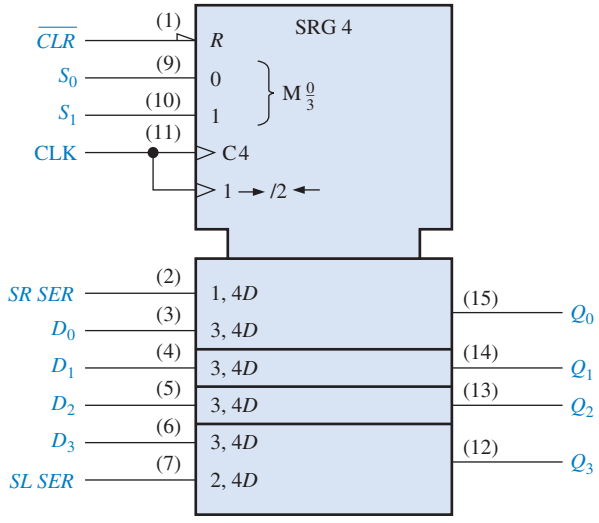
Os quatro modos para o CI 74HC194 são resumidos a seguir:

Não faz nada: $S_0 = 0, S_1 = 0$ (modo 0)

Deslocamento à direita: $S_0 = 1, S_1 = 0$ (modo 1, conforme em 1, 4*D*)

Deslocamento à esquerda: $S_0 = 0, S_1 = 1$ (modo 2, conforme em 2, 4*D*)

Carga paralela: $S_0 = 1, S_1 = 1$ (modo 3, conforme em 3, 4*D*)



▶ **FIGURA 9–40**
Símbolo lógico para o CI 74HC194.

SEÇÃO 9–9 REVISÃO

1. Na Figura 9–40 existem entradas que são dependentes das entradas de modo no estado 0?
2. A carga paralela é síncrona com o clock?

9-10 ANÁLISE DE DEFEITO

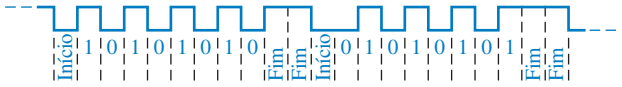
Um método tradicional de análise de defeito em lógica seqüencial e outros sistemas mais complexos usa um procedimento de teste prático do circuito com uma forma de onda conhecida (estímulo) e então observa a saída verificando se a seqüência de bits está correta.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar o procedimento de teste prático como uma técnica de análise de defeito ■ Discutir o teste prático de um conversor de serial para paralelo

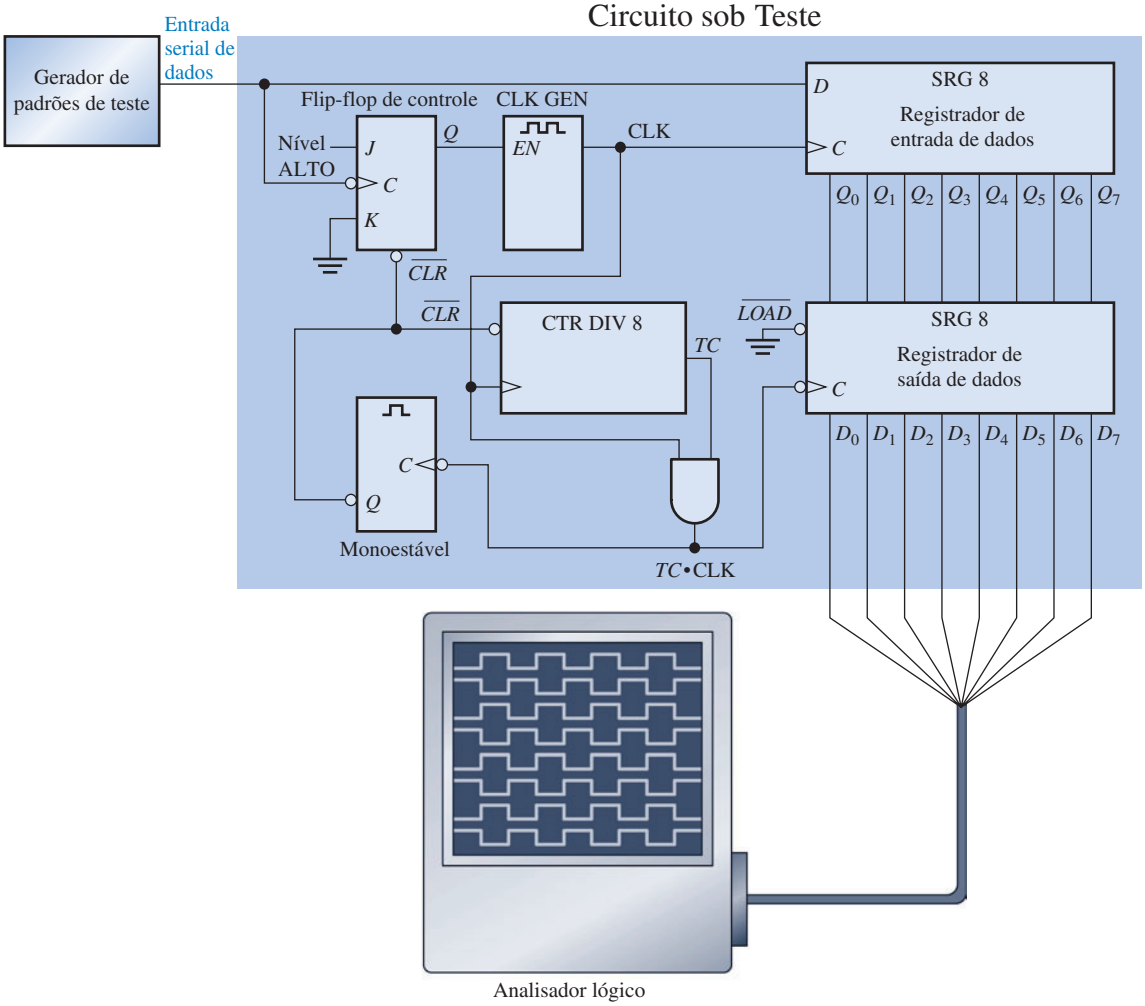
O conversor de dados de serial para paralelo dado na Figura 9–33 é usado para ilustrar o procedimento de teste prático. O principal objetivo do teste prático de um circuito é forçar todos os elementos (flip-flops e portas) em todos os seus estados certificando-se de que nenhum deles se encontre num determinado estado como resultado de um defeito. O padrão de teste de entrada, nesse caso, tem que ser projetado para forçar todos os flip-flops nos registradores a passar pelos dois estados, fazer com que os contadores recebam clocks de forma a passar pelos seus oito estados e fazer com que o flip-flop de controle, o monoestável e a porta AND executem as suas funções.

O padrão de teste de entrada que realiza esse objetivo para o conversor de dados de serial para paralelo é baseado no formato serial de dados que aparece na Figura 9–34. Ele consiste do padrão 10101010 em um grupo serial de bits de dados seguido por 01010101 no próximo grupo, como mostra a Figura 9–41. Esses padrões são gerados de forma repetitiva por um gerador de padrões de teste especial. A configuração básica de teste é mostrada na Figura 9–42.



◀ **FIGURA 9–41**

Amostra do padrão de teste.

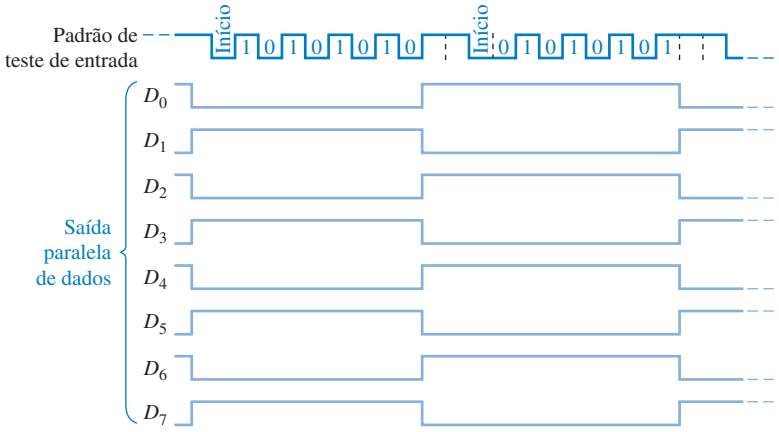


▲ **FIGURA 9–42**

Configuração básica de teste para o conversor de serial para paralelo dado na Figura 9–33.

Após os dois padrões passarem pelo circuito sob teste, todos os flip-flops no registrador de entrada de dados e no registrador de saída de dados terão passado pelos estados SET e RESET, o contador terá percorrido sua seqüência (uma para cada padrão de bit) e todos os outros dispositivos terão sido testados.

Para verificar a operação correta, cada uma das saídas de dados em paralelo é observada para um padrão alternado de 1s e 0s à medida que os padrões de teste de entrada são repetidamente deslocados para o registrador de entrada de dados e em seguida carregados no registrador de saída de dados. O diagrama de temporização para esse caso é mostrado na Figura 9–43. As saídas podem ser observadas aos pares num osciloscópio de duplo traço, ou todas as oito saídas podem ser observadas simultaneamente com um analisador lógico configurado para análise de temporização.



► **FIGURA 9–43**

Saídas corretas para o circuito sob teste dado na Figura 9–42. O padrão de teste de entrada é mostrado.

Se uma ou mais saídas do registrador de saída de dados não estiverem corretas, então temos que verificar as saídas do registrador de entrada de dados. Caso essas saídas estejam corretas, então o problema está associado com o registrador de saída de dados. Verifique as entradas do registrador de saída de dados diretamente nos pinos do CI para ver se há um circuito aberto em alguma linha. Verifique se a alimentação (V_{CC} e GND) está correta (observe se não há algum ruído na linha de GND). Verifique se a linha de carga está firme no nível BAIXO e se existem pulsos de clock na entrada de clock com amplitudes corretas. Certifique-se de que a conexão com o analisador lógico não esteja colocando saídas em curto-circuito. Se todas essas verificações passarem pela verificação, então é provável que o registrador de saída esteja com defeito. Se as saídas do registrador de entrada também estiverem erradas, o defeito pode estar associado com o próprio registrador de entrada ou com qualquer outra lógica, sendo que uma investigação adicional é necessária para isolar o problema.

DICA PRÁTICA

Quando se mede sinais digitais com um osciloscópio, devemos sempre usar o acoplamento cc (DC) em vez de acoplamento ca (AC). O motivo pelo qual o acoplamento ca não é melhor para visualizar sinais digitais é que o nível 0 V no sinal apareceria como nível *médio* do sinal num ponto diferente do valor real de 0 V. É muito mais fácil encontrar um GND "flutuante" ou um nível lógico errado com o acoplamento cc. Se suspeitar de um circuito aberto na linha GND num circuito digital, aumente a sensibilidade do osciloscópio para a máxima possível. Um bom GND nunca apresentará ruído nessa condição, porém um circuito aberto provavelmente apresentará algum ruído, que se mostra como uma flutuação aleatória no nível de 0 V.

SEÇÃO 9–10 REVISÃO

1. Qual é a finalidade de fornecer uma entrada de teste para um circuito lógico seqüencial?
2. Em geral, quando uma forma de onda de saída é verificada como correta, qual é o próximo passo a ser tomado?

Os problemas de análise de defeito que são abordados no CD-ROM estão disponíveis na Seção "Prática de Análise de Defeito Usando o Multisim" no final dos problemas do capítulo.

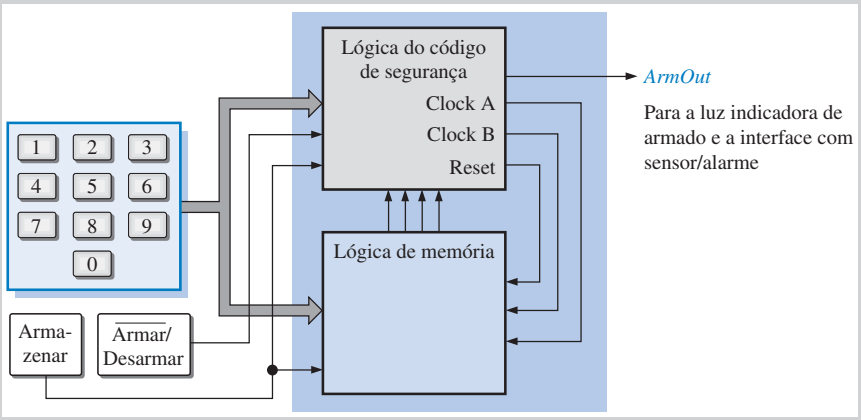
APLICAÇÕES EM SISTEMAS DIGITAIS

Nessa seção de Aplicações em Sistemas Digitais, um sistema relativamente simples é desenvolvido para controlar a segurança de uma sala ou prédio. O sistema pode ser programado com um código de segurança de 4 dígitos entrando com os quatros dígitos, um de cada vez, a partir de um teclado, no modo *desarmar*. Uma vez que o código de segurança foi inserido e armazenado, o sistema é comutado para o modo *armar*. Para desarmar o sistema, temos que inserir o código de 4 dígitos correto no teclado.

Operação Básica

Um diagrama em bloco básico é mostrado na Figura 9–44. O sistema lógico consiste do circuito lógico do código de segurança e do circuito lógico de memória. Nesse capítulo, o foco está relacionado ao circuito de entrada do código. O circuito lógico de memória será desenvolvido no Capítulo 10, sendo que as duas seções lógicas serão combinadas de forma a completar o sistema lógico.

A chave de controle coloca o sistema de segurança no modo *armar* ou no modo *desarmar*. A programação é realizada colocando primeiro o sistema no modo *desarmar* e em seguida pressionando a chave *armazenar* seguida da tecla do dígito para cada um dos quatro dígitos a serem inseridos. Após esse processo, a memória contém os códigos BCD para cada um dos quatro dígitos do código de segurança. Quando o sistema é comutado para o modo *armar*, a saída *ArmOut* habilita os sensores do sistema de alarme e as luzes e um LED para indicar que o sistema está armado. Para entrar na sala ou no prédio, o sistema tem que ser comutado para o modo *desarmar* e os quatro dígitos corretos do código de segurança têm que ser inseridos pelo teclado.



▲ **FIGURA 9–44**

Diagrama em bloco básico do sistema de segurança.

Lógica do Código de Segurança

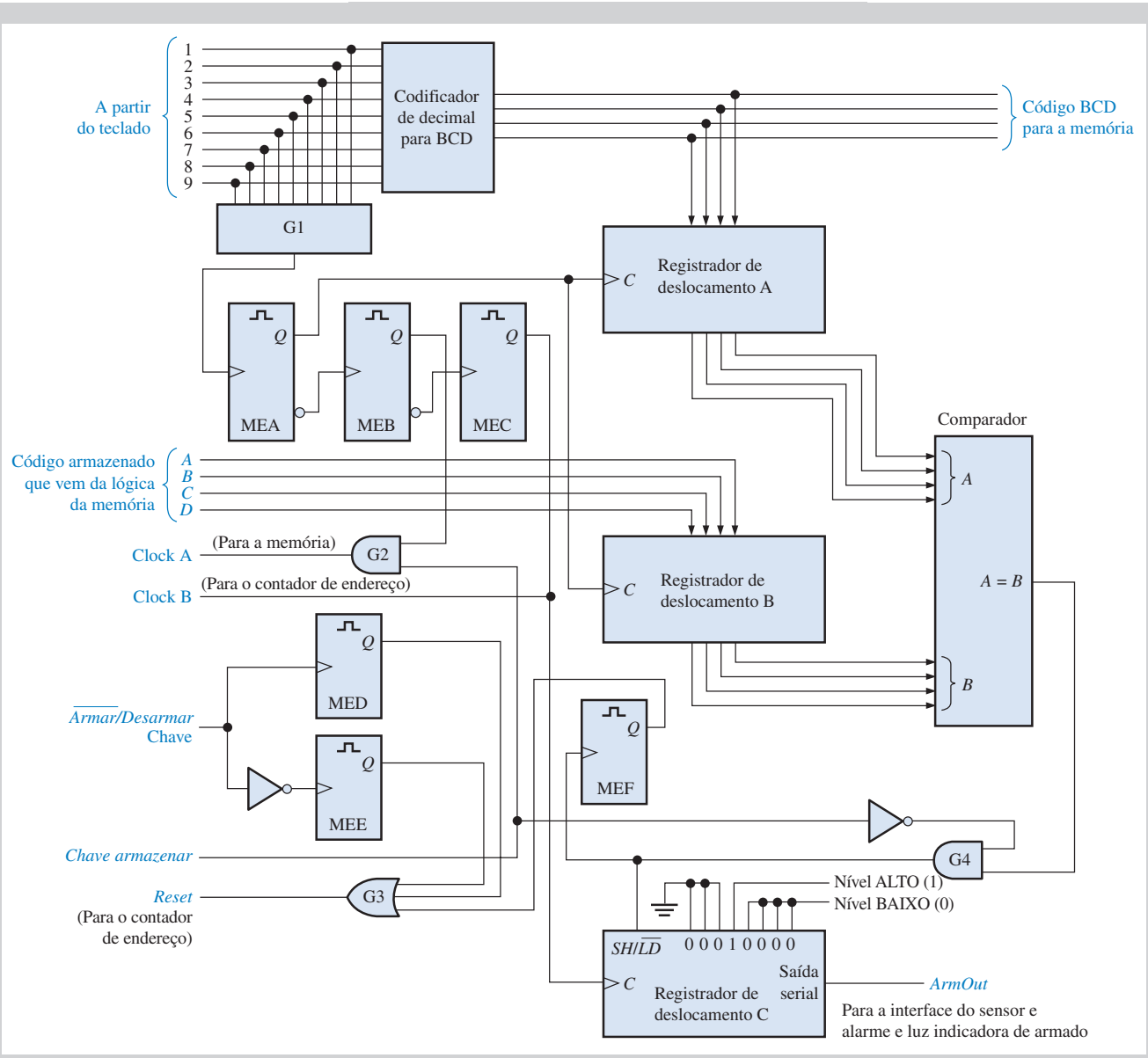
A lógica do código de segurança controla as operações de armar, desarmar, programar e inserir. O diagrama lógico básico é mostrado na Figura 9–45. Quando o sistema é armado primeiro colocando a chave na posição Armar, o registrador de deslocamento C contém 00010000 de forma que existe um nível BAIXO na saída *ArmOut* o qual ativa os sensores do sistema, o circuito de alarme e o indicador de ARMADO. Além disso, um pulso de RESET é gerado pelo monoestável *E* (MEE) para o contador de endereço de memória.

Inserir Para desativar o sistema de forma que alguém possa entrar no local em que o sistema de segurança está instalado, é necessário digitar o código correto de 4 dígitos que seja igual ao código armazenado na memória. O primeiro digito do código de segurança é inserido a partir do teclado. O codificador de decimal para BCD produz o código BCD que representa o dígito que foi pressionado no teclado. O monoestável *A* (MEA) é disparado através da porta G1 produzindo um pulso de clock que armazena o código BCD de 4 bits do codificador no registrador de deslocamento *A* e o código armazenado no primeiro endereço de memória passa para o registrador de deslocamento *B*. Uma vez que os códigos estejam nos registradores *A* e *B*, eles são aplicados nas entradas do comparador. Quando um dígito correto é inserido através do teclado, os 4 bits na entrada *A* do comparador e os 4 bits na entrada *B* serão iguais, resultando em um nível ALTO (1) na saída $A = B$ do comparador e colocando o registrador de deslocamento *C* no modo *deslocar* (SH – *shift*).

A borda de subida do pulso de saída de MEA dispara MEB, o qual, por sua vez, dispara MEC na borda de subida do pulso em sua saída. A saída de MEC gera o *clock B* para o contador de endereço de memória e pulsa a entrada de clock do registrador *C* para deslocar o dado 00010000 para a direita de forma que o registrador agora contenha o dado 00001000. Como ainda existe um 0 (nível BAIXO) na saída serial *ArmOut*, o sistema permanece armado (ativado).

Quando o segundo código do dígito correto é inserido através de teclado, o conteúdo do registrador de deslocamento *C* é deslocado passando a ser 00000100 e o sistema permanece ativado. Quando o código do terceiro dígito é inserido através do teclado, o conteúdo do registrador de deslocamento *C* passa a ser 00000010. Quando o código do quarto e último dígito é inserido, o conteúdo do registrador de deslocamento *C* passa para 00000001. Agora o nível 1 (ALTO) na saída serial *ArmOut* desarma o sistema permitindo a entrada da pessoa no local.

Caso um código incorreto de um dígito seja inserido em qualquer momento, a saída do comparador vai para nível BAIXO, produzindo um nível BAIXO em $SH/\overline{LD}$ e dispara MEF que envia um pulso de resete para o contador de endereço de memória. O registrador de deslocamento *C* agora está no modo de *carga paralela*. O MEC dispara o registrador *C* que é carregado com o código predefinido 00010000. Nesse ponto, temos que começar do início e reinserir os quatro códigos dos dígitos.



▲ **FIGURA 9–45**

Diagrama lógico básico da lógica do código de segurança.

Programação Para programar um código de 4 dígitos no sistema, a chave é colocada na posição desarmar. Isso dispara o monoestável MED que envia um pulso de resete através de G3 para o contador de endereço de memória resetando-o para 00, o primeiro endereço da memória. A chave armazenar é colocada na posição Armazenar o que desabilita a saída $A = B$ do comparador via porta G4 e habilita a saída de MEB via G2 para fornecer um clock para a memória durante a programação de um código na memória.

Em seguida, o primeiro dígito do código de segurança desejado é inserido através do teclado. MEA é disparado através da porta G1 como um resultado da tecla pressionada e, por sua vez, dispara MEB, o qual produz o clock A para armazenar o código na memória. MEB dispara MEC produzindo o clock B para o contador de endereço de memória fazendo-o avançar para o segundo endereço (01). O segundo dígito do código é inserido através do teclado, sendo repetida a seqüência descrita para o primeiro dígito. Após a inserção do quarto e último código do dígito, a memó-

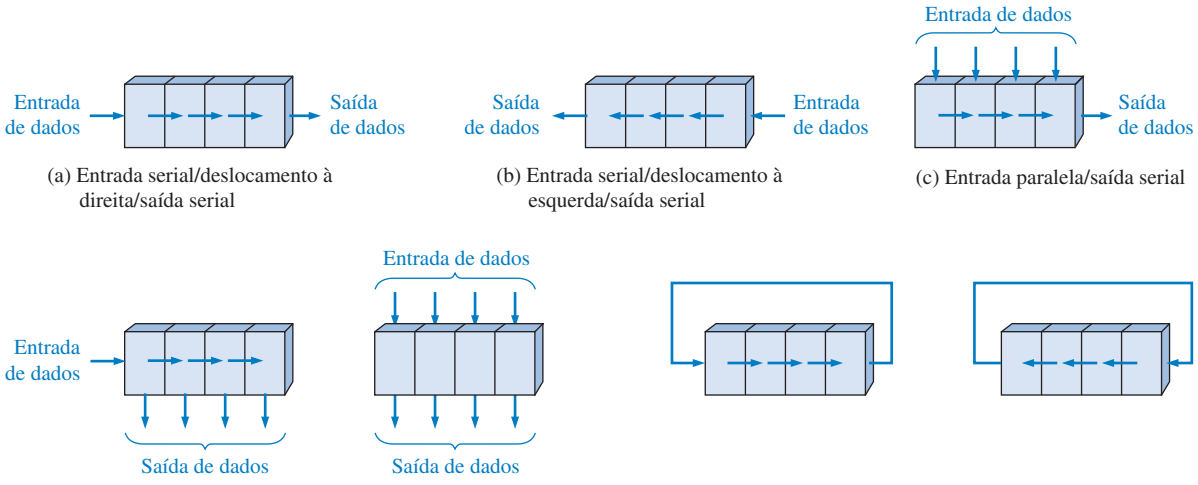
ria contém o código de segurança de 4 dígitos. Caso um dígito incorreto for inserido acidentalmente, temos que concluir a inserção dos 4 dígitos ou reativar a chave ARMAZENAR para garantir que o contador de memória contenha o primeiro endereço novamente. Uma vez feita a programação, o sistema é comutado para o modo *armar*.

Atribuições do Sistema

- *Atividade 1* Descreva a finalidade do registrador de deslocamento *A*.
- *Atividade 2* Descreva a finalidade do registrador de deslocamento *B*.
- *Atividade 3* Descreva a finalidade do registrador de deslocamento *C*.
- *Atividade 4* Descreva a finalidade do comparador.
- *Atividade Opcional* Usando CIs lógicos 74XX e outros componentes, implemente a lógica de código de segurança mostrada na Figura 9–45. Faça qualquer alteração que for necessária para acomodar os dispositivos usados. Faça a depuração e o teste do circuito lógico e descreva qualquer defeito de projeto (caso tenha algum) que você identificar.

RESUMO

- Os tipos básicos de movimentação de dados em registradores de deslocamento são ilustrados na Figura 9–46.



(a) Entrada serial/deslocamento à direita/saída serial

(b) Entrada serial/deslocamento à esquerda/saída serial

(c) Entrada paralela/saída serial

(d) Entrada serial/saída paralela

(e) Entrada paralela/saída paralela

(f) Rotação à direita

(g) Rotação à esquerda

▲ **FIGURA 9–46**

- Registradores de deslocamento usados como contadores são registradores de deslocamento com realimentação que exibem seqüências especiais. Como exemplos temos o contador Johnson e o contador em anel.
- O contador Johnson tem $2n$ estados em sua seqüência, onde n é o número de estágios.
- O contador em anel tem n estágios em sua seqüência.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Bidirecional Tem duas direções. Num registrador de deslocamento bidirecional, o dado armazenado pode ser deslocado para a direita ou para a esquerda.

Carga Para inserir dados num registrador de deslocamento.

Deslocamento Para movermos dados binários de estágio para estágio dentro de um registrador de deslocamento ou outro dispositivo de armazenamento ou mover dados em binário para dentro ou para fora de um dispositivo.

Estágio Um elemento de armazenamento num registrador.

Registrador Um ou mais flip-flops usado para armazenar um dado deslocado.

AUTOTESTE

As respostas estão no final do capítulo.

1. Um estágio em um registrador de deslocamento consiste em
   (a) um latch **(b)** um flip-flop
   (c) um byte de armazenamento **(d)** quatro bits de armazenamento
2. Para deslocar um byte de dados de forma serial num registrador, tem que ter
   (a) um pulso de clock **(b)** um pulso de carga
   (c) oito pulsos de clock **(d)** um pulso de clock para cada nível 1 no dado
3. Para carregar de forma paralela um byte de dados num registrador de deslocamento com uma carga síncrona, tem que ter
   (a) um pulso de clock **(b)** um pulso de clock para cada nível 1
   (c) oito pulsos de clock **(d)** um pulso de clock para cada nível 0 no dado
4. O grupo de bits 10110101 é deslocado de forma serial (primeiro o bit mais à direita) para dentro de um registrador de deslocamento de 8 bits com saída paralela que apresenta inicialmente os estados 11100100. Após dois pulsos de clock, o registrador contém
   (a) 01011110 **(b)** 10110101 **(c)** 01111001 **(d)** 00101101
5. Com uma freqüência de clock de 100 kHz, 8 bits podem ser inseridos de forma serial num registrador de deslocamento em
   (a) 80 μs **(b)** 8 μs **(c)** 80 ms **(d)** 10 μs
6. Com uma freqüência de clock de 1 MHz, oito bits podem ser inseridos de forma paralela num registrador de deslocamento
   (a) em 8μs **(b)** no tempo de atraso de propagação de oito flip-flops
   (c) em 1μs **(d)** no tempo de atraso de propagação de um flip-flop
7. Um contador Johnson de módulo 10 necessita de
   (a) dez flip-flops **(b)** quatro flip-flops **(c)** cinco flip-flops **(d)** doze flip-flops
8. Um contador em anel de módulo 10 necessita no mínimo de
   (a) dez flip-flops **(b)** cinco flip-flops **(c)** quatro flip-flops **(d)** doze flip-flops
9. Quando um registrador de deslocamento de 8 bits com entrada serial/saída serial é usado para gerar um atraso de tempo de 24μs, a freqüência de clock tem que ser de
   (a) 41,67 kHz **(b)** 333 kHz **(c)** 125 kHz **(d)** 8 MHz
10. A finalidade do uso de um contador em anel no circuito de codificação de teclado visto na Figura 9–38 é
    (a) aplicar seqüencialmente um nível ALTO em cada linha para detectar a tecla pressionada.
    (b) fornecer pulsos de disparo para o registrador do código da tecla.
    (c) aplicar seqüencialmente um nível BAIXO em cada linha para detecção da tecla pressionada.
    (d) para inverter seqüencialmente a polarização dos diodos em cada linha.

PROBLEMAS

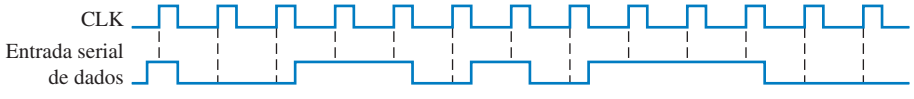
As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 9–1 Funções Básicas de Registradores de Deslocamento

1. Por que os registradores de deslocamento são considerados dispositivos básicos de memória?
2. Qual é a capacidade de armazenamento de um registro que pode guardar dois bytes de dados?

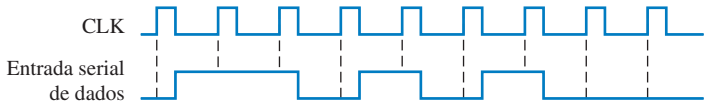
SEÇÃO 9–2 Registradores de Deslocamento com Entrada Serial/Saída Serial

3. Para a entrada de dados e o clock mostrados na Figura 9–47, determine os estados de cada flip-flop no registrador de deslocamento visto na Figura 9–3 e mostre as formas de onda de Q. Considere que o registrador contém inicialmente somente 1s.



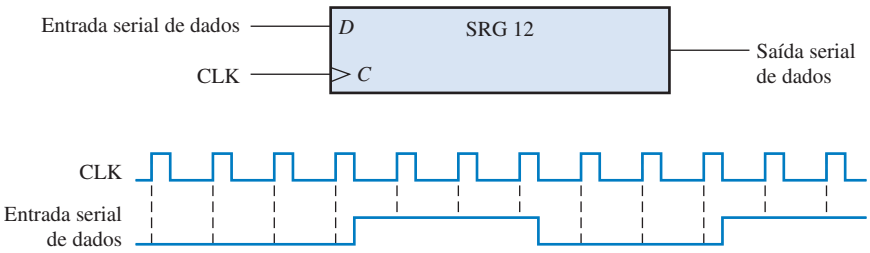
▶ FIGURA 9–47

4. Resolva o Problema 3 para as formas de onda dadas na Figura 9–48.



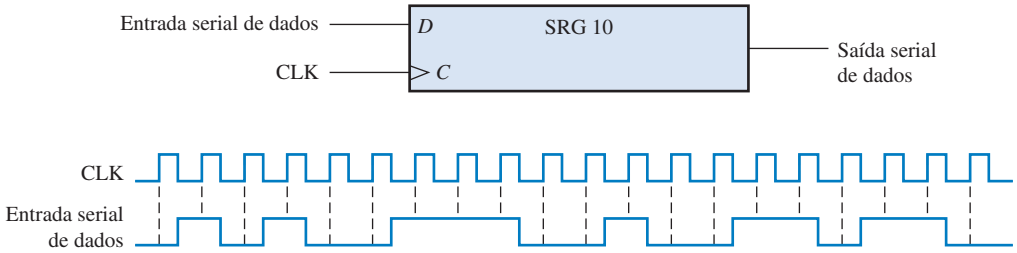
▶ FIGURA 9–48

5. Qual é o estado do registrador visto na Figura 9–49 após cada pulso de clock se ele começa no estado 101001111000?



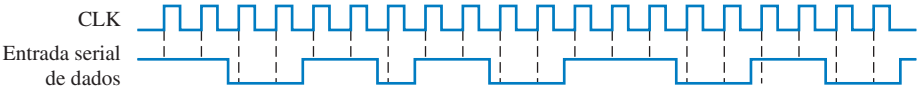
▲ FIGURA 9–49

6. Para o registrador de deslocamento com entrada serial/saída serial, determine a forma de onda na saída de dados para a entrada de dados e o clock mostrados na Figura 9–50. Considere que o registrador esteja inicialmente resetado.



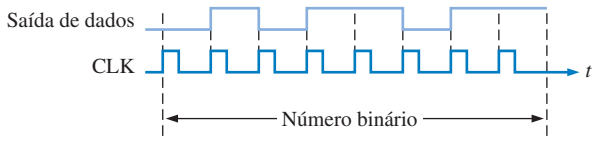
▲ FIGURA 9–50

7. Resolva o Problema 6 para as formas de onda dadas na Figura 9–51.



▲ FIGURA 9–51

8. Um registrador de deslocamento com entrada serial/saída serial e clock ativo na borda de subida tem uma forma de onda na saída de dados como mostra a Figura 9–52. Qual é o número binário armazenado num registrador de 8 bits se o primeiro bit de dados a sair (o bit mais à esquerda) for o LSB?



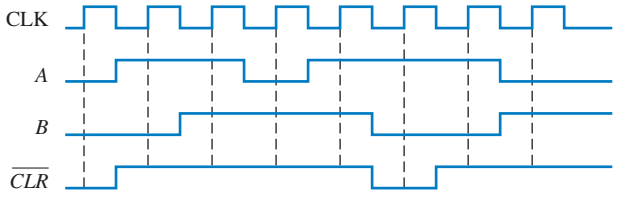
▶ FIGURA 9–52

SEÇÃO 9–3 Registradores de Deslocamento com Entrada Serial/Saída Paralela

9. Mostre um diagrama de temporização completo com as saídas em paralelo para o registrador de deslocamento dado na Figura 9–8. Use as formas de onda dadas na Figura 9–50 com o registrador inicialmente resetado.

10. Resolva o Problema 9 para as formas de onda de entrada vistas na Figura 9–51.

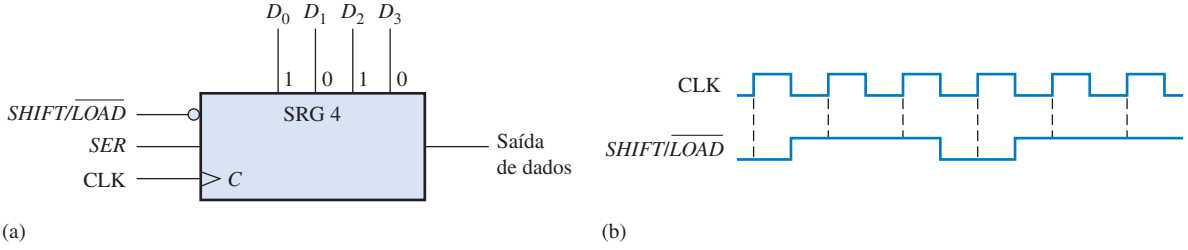
11. Desenvolva as formas de onda para as saídas de Q_0 a Q_7 para um CI registrador de deslocamento 74HC164 com as formas de onda de entrada mostradas na Figura 9–53.



▶ FIGURA 9–53

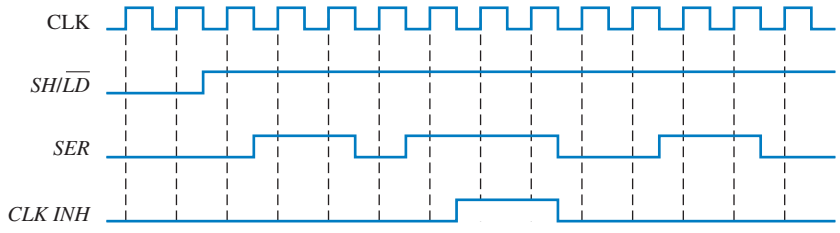
SEÇÃO 9–4 Registradores de Deslocamento com Entrada Paralela/Saída Serial

12. O registrador de deslocamento na Figura 9–54(a) tem as entradas $SHIFT/\overline{LOAD}$ e CLK como mostra a parte (b) da figura. A entrada serial de dados (*SER*) é nível 0. As entradas de dados em paralelo são $D_0 = 1$, $D_1 = 0$, $D_2 = 1$ e $D_3 = 0$, como mostrado. Desenvolva a forma de onda da saída de dados em relação às entradas.



▲ FIGURA 9–54

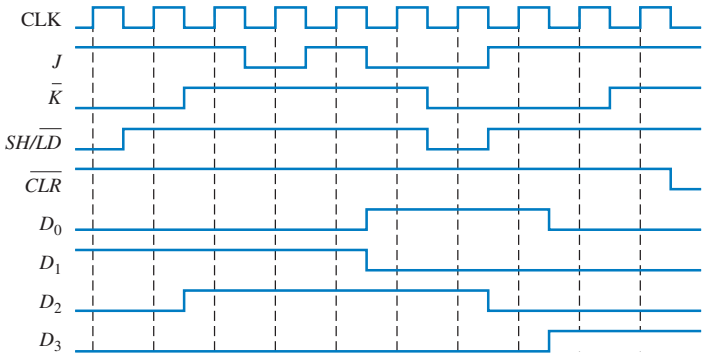
13. As formas de onda mostradas na Figura 9–55 são aplicadas ao registrador de deslocamento 74HC165. As entradas em paralelo são todas nível 0. Determine a forma de onda de Q_7.



▲ FIGURA 9–55

14. Resolva o Problema 13 se todas entradas em paralelo são nível 1.

15. Resolva o Problema 13 se a entrada *SER* for invertida.



▶ FIGURA 9–56

SEÇÃO 9–5 Registradores de Deslocamento com Entrada Paralela/Saída Paralela

16. Determine as formas de onda para todas as saídas *Q* do CI registrador de deslocamento de 4 bits 74HC195 quando as entradas são como mostra a Figura 9–56.

17. Resolva o Problema 16 se a entrada $SH/\overline{LD}$ for invertida e o registrador for inicialmente resetado.

18. Use dois CIs registradores de deslocamento 74HC195 para formar um registrador de deslocamento de 8 bits. Mostre as conexões necessárias.

SEÇÃO 9–6 Registradores de Deslocamento Bidirecionais

19. Para o registrador bidirecional de 8 bits visto na Figura 9–57, determine o estado do registrador após cada pulso de clock para a forma de onda da entrada de controle $RIGHT/\overline{LEFT}$ dada. Um nível ALTO nessa entrada habilita um deslocamento à direita, e um nível BAIXO habilita um deslocamento à esquerda. Considere que o registrador esteja inicialmente armazenando o número decimal setenta e seis em binário, com a posição mais à direita sendo o LSB. Existe um nível BAIXO na linha de entrada de dados.

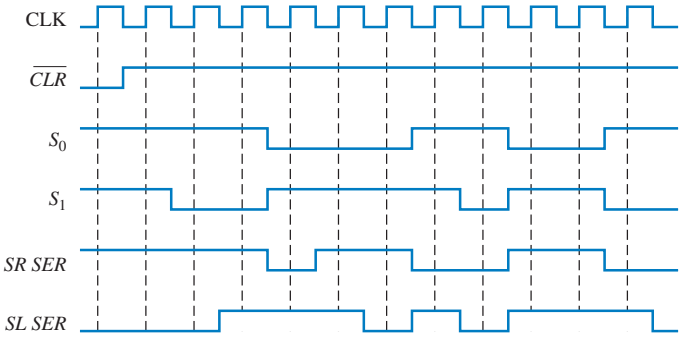
▲ FIGURA 9–57

20. Resolva o Problema 19 para as formas de onda dadas na Figura 9–58.

▶ FIGURA 9–58

21. Use dois CIs registradores de deslocamento de 4 bits bidirecionais 74HC194 para criar um registrador de deslocamento de 8 bits bidirecional. Mostre as conexões.

22. Determine as saídas *Q* de um CI 74HC194 com as entradas mostradas na Figura 9–59. As entradas D_0, D_1, D_2 e D_3 são nível ALTO.



▶ FIGURA 9–59

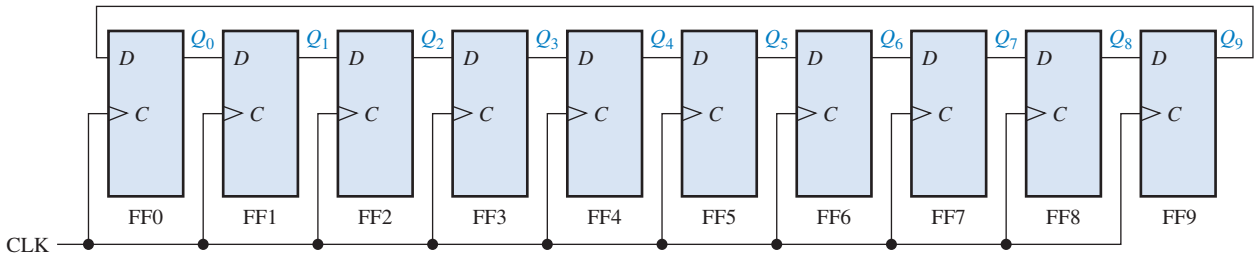
SEÇÃO 9–7 Registradores de Deslocamento como Contadores

23. Quantos flip-flops são necessários para implementar cada uma das seguintes configurações de contadores Johnson:

(a) módulo 6 **(b)** módulo 10 **(c)** módulo 14 **(d)** módulo 16

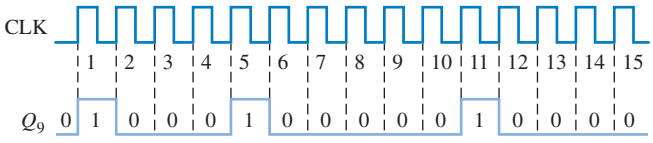
24. Desenhe o diagrama lógico para um contador Johnson de módulo 18. Mostre o diagrama de temporização e escreva a seqüência na forma tabular.

25. Para o contador em anel dado na Figura 9–60, mostre as formas de onda para cada saída de flip-flop em relação ao clock. Considere que o FF0 esteja inicialmente setado e que o restante esteja resetado. Mostre pelo menos dez pulsos de clock.



▲ FIGURA 9–60

26. Precisamos gerar a forma de onda mostrada na Figura 9–61. Projete um contador em anel e indique como ele pode ser presetado para produzir essa forma de onda na saída Q_9. No CLK16 o padrão mostrado começa a repetir.



▶ FIGURA 9–61

SEÇÃO 9–8 Aplicações de Registradores de Deslocamento

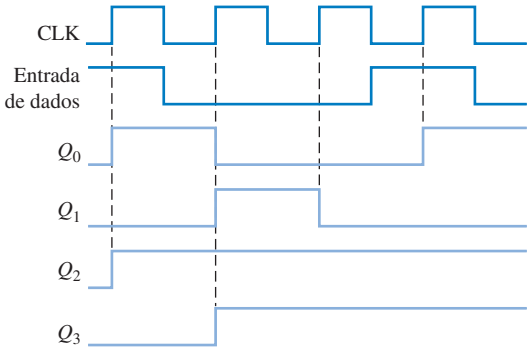
27. Use CIs registradores de deslocamento de 4 bits 74HC195 para implementar um contador em anel de 16 bits. Mostre as conexões.

28. Qual é a finalidade da entrada ao energizar na Figura 9–38?

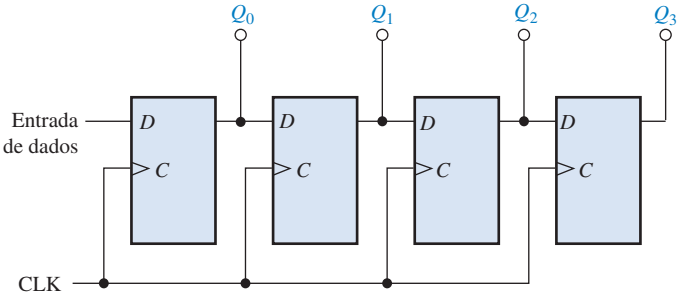
29. O que acontece quando duas teclas são pressionadas simultaneamente no circuito da Figura 9–38?

SEÇÃO 9–10 Análise de Defeito

30. Com base nas formas de onda dadas na Figura 9–62(a), determine o problema mais provável com o registrador visto na parte (b) da figura.



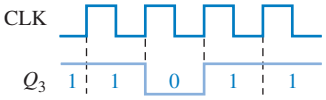
(a)



(b)

▲ FIGURA 9–62

31. Consulte o registrador de deslocamento com entrada paralela/saída serial dado na Figura 9–12. O registrador está no estado onde $Q_0Q_1Q_2Q_3 = 1001$ e $D_0D_1D_2D_3 = 1010$ é carregado nele. Quando a entrada $SHIFT/\overline{LOAD}$ for nível ALTO, os dados mostrado na Figura 9–63 são deslocados para fora. Essa operação está correta? Em caso negativo, qual deve ser o problema mais provável?



▶ **FIGURA 9–63**

32. Verificamos que o registrador visto na Figura 9–19 desloca os dados para a direita mas não para a esquerda. Qual deve ser o defeito mais provável?

33. Para o codificador de teclado mostrado na Figura 9–38, faça uma lista dos possíveis defeitos para cada um dos seguintes sintomas:

(a) O estado do registrador do código da tecla não muda para qualquer tecla pressionada.

(b) O estado do registrador do código da tecla não muda quando qualquer tecla na terceira linha é pressionada. Para todas as outras teclas é gerado o código correto.

(c) O estado do registrador do código da tecla não muda quando qualquer tecla na primeira coluna é pressionada. Para todas as outras teclas é gerado o código correto.

(d) Quando qualquer tecla da segunda coluna é pressionada, os três bits da esquerda do código da tecla ($Q_0Q_1Q_2$) são corretos, porém os três bits da direita são todos nível 1.

34. Desenvolva um procedimento para um teste prático do codificador de teclado visto na Figura 9–38. Especifique o procedimento passo a passo, indicando o código de saída a partir do registrador do código da tecla que deve ser observado em cada passo do teste.

35. Que sintomas são observados para os seguintes defeitos no conversor de serial para paralelo mostrado na Figura 9–33:

(a) a saída da porta AND está fixa no estado ALTO.

(b) a saída do gerador de clock está fixa no estado BAIXO.

(c) o terceiro estágio do registrador de entrada de dados está fixo no estado SET.

(d) a saída de contagem final do contador está fixa no estado ALTO.

Aplicações em Sistemas Digitais

36. Qual é a principal finalidade da lógica do código de segurança?

37. Considere que o código digitado seja 1939. Determine os estados dos registradores de deslocamento *A* e *C* após o segundo dígito correto ter sido digitado.

38. Considere que ao senha seja 7646 e que o número digitado seja 7645. Determine os estados dos registradores de deslocamento *A* e *C* após cada dígito correto ter sido digitado.

Problemas Especiais de Projeto

39. Especifique os dispositivos que podem ser usados para implementar o conversor de dados de serial para paralelo mostrado na Figura 9–33. Desenvolva o diagrama lógico completo mostrando quaisquer modificações necessárias para acomodar os dispositivos específicos usados.

40. Modifique o conversor de serial para paralelo dado na Figura 9–33 para prover uma conversão de 16 bits.

41. Projete um conversor de dados de paralelo para serial de 8 bits que produza o formato de dados mostrado na Figura 9–34. Mostre um diagrama lógico e especifique os dispositivos.

42. Projete um circuito para ativar o sinal $\overline{LOAD}$ na energização para o codificador de teclado apresentado na Figura 9–38. Esse circuito tem que gerar um pulso em nível BAIXO de curta duração quando a chave de alimentação for ligada.

43. Implemente o gerador de padrão de teste usado na Figura 9–42 para análise de defeito do conversor de serial para paralelo.

44. Reveja o sistema de controle e contagem de comprimidos que foi apresentado no Capítulo 1. (a) Utilizando o conhecimento adquirido neste capítulo, implemente os registradores A e B no sistema mencionado usando CIs de função fixa específicos. (b) Implemente o sistema usando seu software de desenvolvimento.

Prática de Análise de Defeito Usando o Multisim

45. Abra o arquivo P09-45 e teste o registrador de deslocamento de 4 bits para determinar se existe um defeito. Identifique o defeito se possível.

46. Abra o arquivo P09-46 e teste o registrador de deslocamento de 8 bits com entrada serial/saída paralela 74164 para determinar se existe um defeito. Identifique o defeito se possível.

47. Abra o arquivo P09-47 e teste o registrador de deslocamento de 8 bits com carga paralela 74165 para determinar se existe um defeito. Identifique o defeito se possível.

48. Abra o arquivo P09-48 e teste o registrador de deslocamento de 4 bits com acesso paralelo para determinar se existe um defeito. Se existir um defeito, Identifique-o se possível.

49. Abra o arquivo P09-49 e teste o contador em anel de 10 bits para determinar se existe um defeito. Se existir um defeito, Identifique-o se possível.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 9–1 Funções Básicas de Registradores de Deslocamento

1. Um contador tem uma seqüência especificada de estados, porém um registrador de deslocamento não.

2. Armazenamento e movimentação de dados são duas funções de um registrador de deslocamento.

SEÇÃO 9–2 Registradores de Deslocamento com Entrada Serial/Saida Serial

1. FF0: entrada de dados em J_0, entrada de dados em K_0; FF1:Q_0 em J_1, $\overline{Q}_0$ em K_1; FF2: Q_1 em J_2, $\overline{Q}_1$ em K_2; FF3: Q_2 em J_3, $\overline{Q}_2$ em K_3

2. Oito pulsos de clock

SEÇÃO 9–3 Registradores de Deslocamento com Entrada Serial/Saída Paralela

1. 0100 após 2 pulsos de clock

2. Tome a saída serial no flip-flop mais à direita para operação de saída serial

SEÇÃO 9–4 Registradores de Deslocamento com Entrada Paralela/Saída Serial

1. Quando $SHIFT/\overline{LOAD}$ for nível ALTO, os dados são deslocados à direita um bit a cada pulso de clock. Quando $SHIFT/\overline{LOAD}$ for nível BAIXO, os dados nas entradas paralelas são carregados no registrador.

2. A operação de carga paralela é assíncrona, assim, ela não depende do clock.

SEÇÃO 9–5 Registradores de Deslocamento com Entrada Paralela/Saída Paralela

1. As saídas de dados são 1001.

2. $Q_0 = 1$ após um pulso de clock.

SEÇÃO 9–6 Registradores de Deslocamento Bidirecionais

1. 1111 após o quinto pulso de clock.

SEÇÃO 9–7 Registradores de Deslocamento como Contadores

1. Dezesseis estados tem um contador Johnson de 8 bits.

2. Para um contador Johnson de 3 bits: 000, 100, 110, 111, 011, 001, 000

SEÇÃO 9–8 Aplicações de Registradores de Deslocamento

1. 625 escaneamentos/segundo

2. $Q_5Q_4Q_3Q_2Q_1Q_0 = 011011$

3. Os diodos proporcionam caminhos unidirecionais para puxar as LINHAs de nível BAIXO e evitar níveis ALTOs nas LINHAs onde está conectada a matriz de chaves. Os resistores elevam as COLUNAs para o nível ALTO.

SEÇÃO 9–9 Símbolos Lógicos com Notação de Dependência

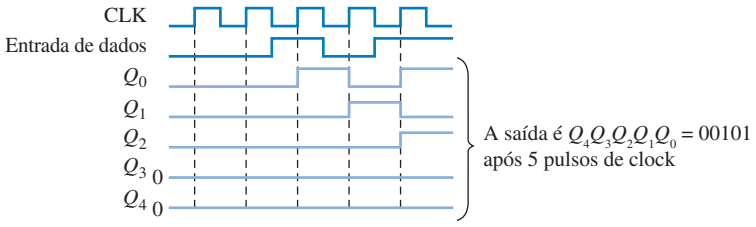
1. Nenhuma das entradas são dependentes das entradas de modo estando no estado 0.
2. Sim, a carga paralela é síncrona com o clock conforme a indicação 4*D*.

SEÇÃO 9–10 Análise de Defeito

1. Uma entrada de teste é usada para fazer com que o circuito passe por todos os estados.
2. Verifique a entrada para essa parte do circuito. Se o sinal nessa entrada estiver correto, o defeito está isolado no circuito entre a entrada boa e a saída ruim.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

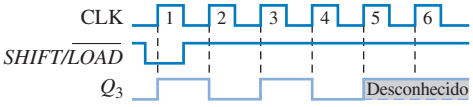
9–1. Veja a Figura 9–64.



▶ FIGURA 9–64

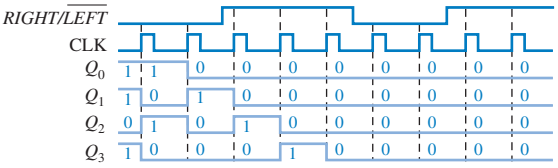
9–2. O estado do registrador após três pulsos de clocks adicionais é 0000.

9–3. Veja a Figura 9–65.



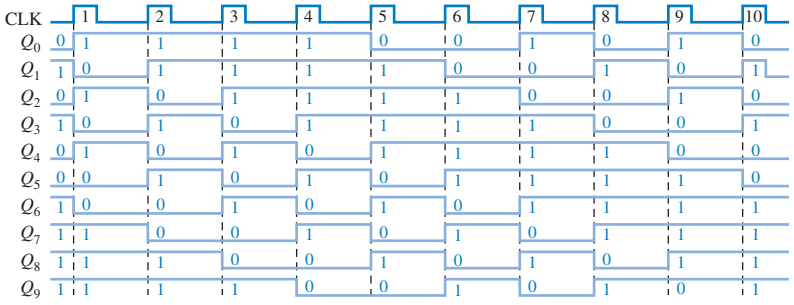
▶ FIGURA 9–65

9–4. Veja a Figura 9–66.



▶ FIGURA 9–66

9–5. Veja a Figura 9–67.



▶ FIGURA 9–67

9–6. $f = 1/3\ \mu s = 333$ kHz

AUTOTESTE

1. (b) **2.** (c) **3.** (a) **4.** (c) **5.** (a) **6.** (d) **7.** (c) **8.** (a)
9. (b) **10.** (c)

10

MEMÓRIA E ARMAZENAMENTO

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Definir as características básicas das memórias
- Explicar o que é uma RAM e como ela funciona
- Explicar a diferença entre RAMs estáticas (SRAMs) e RAMs dinâmicas (DRAMs)
- Explicar o que é uma ROM e como ela funciona
- Descrever os diversos tipos de PROMs
- Discutir as características de uma memória flash
- Descrever a expansão de ROMs e RAMs para aumentar o tamanho da palavra e a capacidade de palavras
- Discutir tipos especiais de memórias tais como FIFO e LIFO

- Descrever a organização básica dos discos magnéticos e fitas magnéticas
- Descrever a operação básica dos discos magneto-óptico e óptico
- Descrever os métodos básicos de teste de memória
- Desenvolver fluxogramas para teste de memória
- Usar um dispositivo de memória numa aplicação em sistemas digitais

TERMOS IMPORTANTES

- Byte
- Word
- Célula
- Endereço
- Capacidade
- Escrita
- Leitura
- RAM
- ROM
- SRAM
- Barramento
- DRAM
- PROM
- EPROM
- Memória flash
- FIFO
- LIFO
- Disco rígido

INTRODUÇÃO

No Capítulo 9, abordamos os registradores de deslocamento, que são um tipo de dispositivo de armazenamento; na realidade, um registrador de deslocamento é essencialmente uma memória em pequena escala. Os dispositivos de memória abordados nesse capítulo são geralmente usados para armazenar por um período mais longo uma maior quantidade de dados que um registrador é capaz.

Os computadores e outros tipos de sistemas necessitam de armazenamento permanente ou semi-permanente de uma grande quantidade de dados binários. Os sistemas microprocessados fazem uso de dispositivos de armazenamento e memórias para suas operações por causa da necessidade do armazenamento de programas e para retenção de dados durante o processamento.

Na terminologia da computação, geralmente quando se fala em *memória* nos referimos a RAM e ROM e quando se fala em armazenamento nos referimos a disco rígido, disco flexível e CD-ROM. Nesse capítulo, abordamos as memórias e os meios de armazenamento magnético e óptico.

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico Aplicações em Sistemas Digitais no final do capítulo conclui o sistema de segurança apresentado no Capítulo 9. O foco neste capítulo é o circuito lógico da memória do sistema, que memoriza o código de entrada. Uma vez desenvolvido o circuito lógico da memória, ele é interfaceado com o circuito lógico do código de segurança desenvolvido no Capítulo 9 para completar o sistema.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

10-1 FUNDAMENTOS DE MEMÓRIA SEMICONDUTORA

A memória é a parte de um sistema de armazenamento de uma grande quantidade de dados em binário. As memórias semicondutoras consistem em arranjos de elementos que geralmente são latches ou capacitores.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar como uma memória armazena dados binários ■ Discutir a organização básica de uma memória ■ Descrever a operação de escrita ■ Descrever a operação de leitura ■ Descrever a operação de endereçamento ■ Explicar o que são RAMs e ROMs

Unidades de Dados e Binário: Bits, Bytes, Nibbles e Words

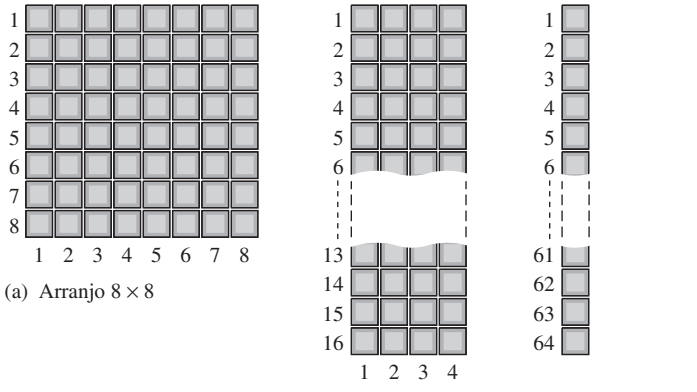
Como regra, as memórias armazenam dados em unidades que tem de um a oito bits. A menor unidade de dados binários, como sabemos, é o **bit**. Em muitas aplicações, os dados são manipulados em unidades de 8 bits denominadas de **byte** ou em múltiplos de unidades de 8 bits. Um byte pode ser dividido em duas unidades de 4 bits que são denominadas de **nibbles**. Uma unidade completa de informação é denominada de **word** e geralmente consiste de um ou mais bytes. Algumas memórias armazenam dados em grupos de 9 bits; um grupo de 9 bits consiste de um byte mais um bit de paridade.

NOTA: COMPUTAÇÃO

A definição geral de *word* (palavra) é uma unidade completa de informação que consiste de uma unidade de dados em binário. Quando aplicada às instruções de um computador, uma word é mais especificamente definida com dois bytes (16 bits). Uma parte muito importante da linguagem assembly usada em computadores, a diretiva DW (define word – definição de palavra) significa definir dados em unidades de 16 bits. Essa definição é independente do microprocessador em particular ou do tamanho do seu barramento de dados. A linguagem assembly também permite definições de bytes (8 bits) com a diretiva DB, double word (32 bits) com a diretiva DD e quad-words (64 bits) com a diretiva QD.

Arranjo Básico da Memória Semicondutora

Cada elemento de armazenamento numa memória pode reter um nível 1 ou um nível 0 é denominado de **célula**. As memórias são construídas de arranjos de células, conforme ilustrado na Figura 10–1 usando como exemplo 64 células. Cada bloco no **arranjo da memória** representa uma célula de armazenamento e a sua posição pode ser identificada especificando a linha e a coluna correspondente.



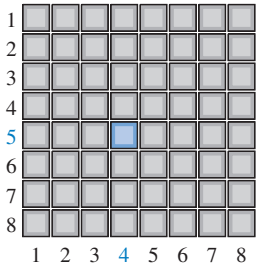
▲ **FIGURA 10–1**

Um arranjo de memória de 64 células organizadas de três formas diferentes.

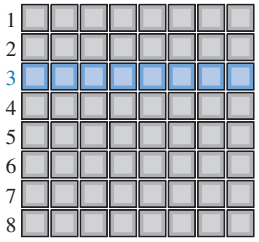
O arranjo de 64 células pode ser organizado de várias formas baseado nas unidades de dados. A Figura 10–1(a) mostra um arranjo 8 × 8, que pode ser visto como uma memória de 64 bits ou uma de 8 bytes. A parte (b) da figura mostra um arranjo de 16 × 4, que é uma memória de 16 nibbles e a parte (c) mostra um arranjo de 64 × 1, que é uma memória de 64 bits. Uma memória é identificada pelo número de palavras que ela pode armazenar vezes o tamanho da palavra. Por exemplo, uma memória de 16k × 8 pode armazenar 16.384 palavras de oito bits. A inconsistência que aparentemente existe aqui é comum na terminologia de memória. O número real de palavras é sempre uma potência inteira de 2, que, nesse caso, é 2^{14} = 16.384. Entretanto, é prática comum declarar o número que se aproxima de um múltiplo de mil, nesse caso, 16k.

Endereço de Memória e Capacidade

A localização de uma unidade de dado num arranjo de memória é denominada **endereço**. Por exemplo, na Figura 10–2(a), o endereço de um bit no arranjo de 2 dimensões é especificado por uma linha e uma coluna como mostrado. Na Figura 10–2(b), o endereço de um byte é especificado apenas pela linha. Assim, como podemos ver, o endereço depende de como a memória está organizada em unidades de dados. Os computadores pessoais têm memórias de acesso aleatório organizadas em bytes. Isso significa que o menor grupo que pode ser endereçado é de oito bits.



(a) O endereço do bit laranja é a linha 5 e a coluna 4.

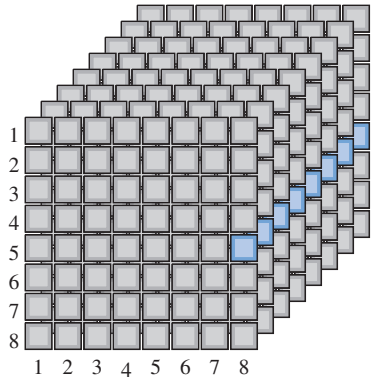


(b) O endereço do byte laranja é a linha 3.

◀ FIGURA 10–2

Exemplos de endereço de memória num arranjo de 2 dimensões.

Na Figura 10–3, o endereço de um byte num arranjo de três dimensões é especificado pela linha e pela coluna como mostrado. Nesse caso, o menor grupo que pode ser acessado é de oito bits.



O endereço do byte laranja é a linha 5 e a coluna 8.

◀ FIGURA 10–3

Exemplo de endereço de memória num arranjo de 3 dimensões.

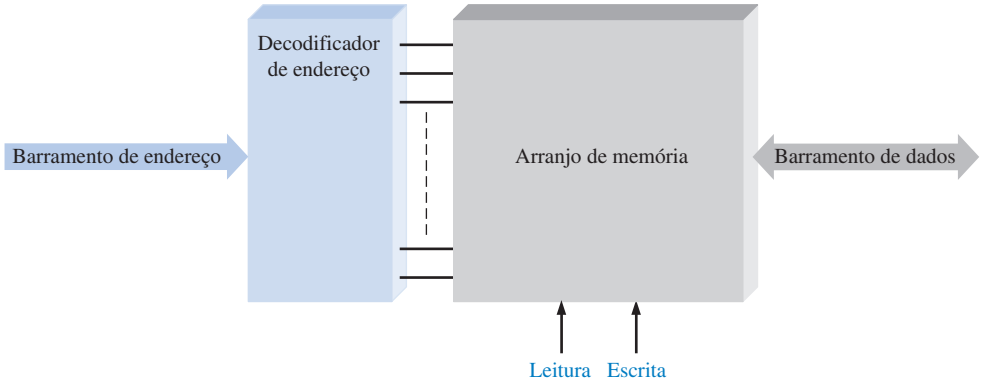
A **capacidade** de uma memória é o número total de unidades de dados que podem ser armazenadas. Por exemplo, no arranjo de memória organizado em bits na Figura 10–2(a), a capacidade é 64 bits. Na memória organizada em bytes na Figura 10–2(b), a capacidade é 8 bytes, que também é 64 bits. Na Figura 10–3, a capacidade é 64 bytes. As memórias dos computadores têm tipicamente 256 MB (MB é megabyte) ou mais de memória interna.

Operações Básicas com Memórias

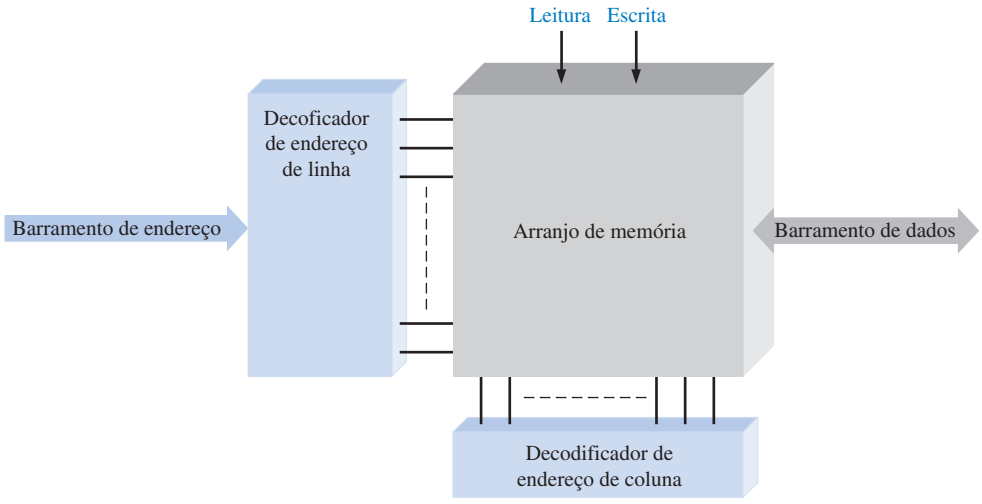
Como uma memória armazena dados binários, os dados têm que ser inseridos numa memória e copiados delas quando necessário. A operação de **escrita** insere dados num endereço específico da memória e a operação de **leitura** copia dados de um endereço específico na memória. A operação de endereçamento, que é parte das operações de leitura e escrita, seleciona o endereço de memória especificado.

As unidades de dados entram na memória durante uma operação de escrita e saem da memória durante uma operação de leitura através de um conjunto de linhas denominado *barramento de dados*. Conforme indicado na Figura 10–4, o barramento de dados é bidirecional, o que significa que os dados podem trafegar em qualquer direção (para dentro ou para fora da memória). Nesse

caso de memórias organizadas em bytes, o barramento de dados tem pelo menos oito linhas de forma que todos os oito bits no endereço selecionado são transferidos em paralelo. Para uma operação de escrita ou leitura, um endereço é selecionado colocando um código binário, que representa o endereço desejado, num conjunto de linhas denominado *barramento de endereço*. O código do endereço é decodificado internamente e o endereço apropriado é selecionado. No caso do arranjo de memória de 3 dimensões dado na Figura 10–4(b) existem dois decodificadores, um para as linhas e outro para as colunas. O número de linhas no barramento de endereço depende da capacidade da memória. Por exemplo, um código de endereço de 15 bits pode selecionar 32.768 posições (2^{15}) na memória, um código de endereço de 16 bits pode selecionar 65.536 posições (2^{16}) na memória, e assim por diante. Em computadores pessoais, um barramento de endereço de 32 bits pode selecionar 4.294.967.296 posições (2^{32}), expressa como 4G.



(a) Arranjo de memória de 2 dimensões

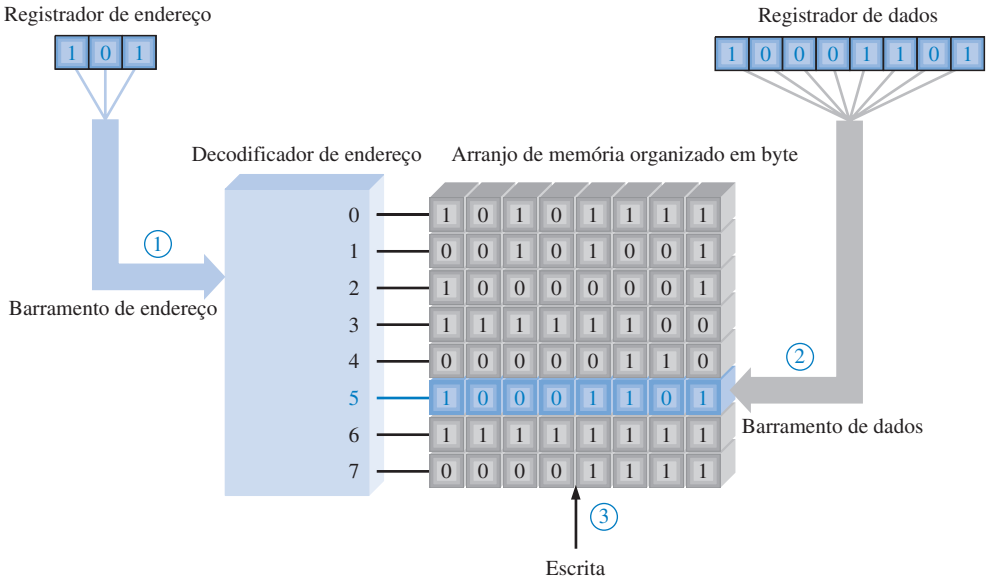


(b) Arranjo de memória de 3 dimensões

▶ FIGURA 10–4

Diagrama em bloco de uma memória de 2 dimensões e uma memória de 3 dimensões mostrando o barramento de endereço, decodificador(es) de endereço, barramento de dados bidirecional e entradas de leitura/escrita.

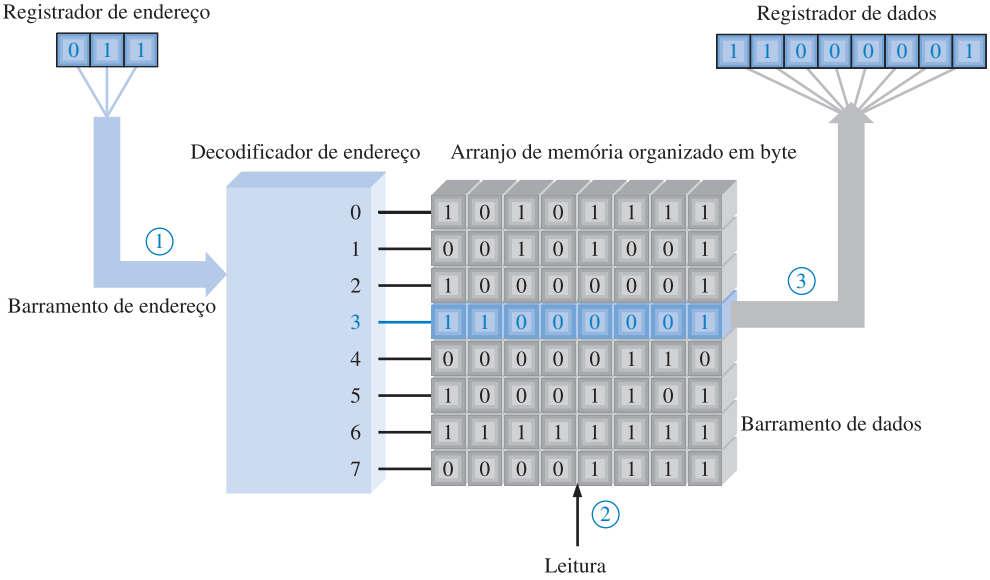
A Operação de Escrita Uma operação de escrita simplificada é ilustrada na Figura 10–5. Para armazenar um byte de dados numa memória, um código existente no registrador de endereço é colocado no barramento de endereço. Uma vez que o código do endereço esteja no barramento, o decodificador de endereço decodifica o endereço e seleciona na memória a posição especificada. A memória então recebe um comando de escrita e o byte de dados armazenado no registrador de dados é colocado no barramento de dados e armazenado no endereço de memória selecionado, completando assim a operação de escrita. Quando um novo byte de dados é escrito num endereço de memória, o byte de dados atual armazenado nesse endereço é sobrescrito (substituído pelo novo byte de dados).



① O código de endereço 101 é colocado no barramento de endereço e o endereço 5 é selecionado.
② O byte de dados é colocado no barramento de dados.
③ O comando de escrita faz com que o byte de dados seja armazenado no endereço 5, substituindo o dado anterior.

◀ **FIGURA 10–5**
Ilustração da operação de escrita.

A Operação de Leitura Uma operação de leitura simplificada está ilustrada na Figura 10–6. Novamente, um código existente no registrador de endereço é colocado no barramento de endereço. Uma vez que o código do endereço esteja no barramento, o decodificador de endereço decodifica o endereço e seleciona a posição especificada na memória. A memória então recebe um comando de leitura e uma "cópia" do byte de dados que está armazenado no endereço de memória selecionado é colocado no barramento de dados e carregado no registrador de dados, completando assim a operação de leitura. Quando um byte de dados é lido a partir de um endereço de memória, ele também permanece armazenado no endereço. Essa operação é denominada *leitura não destrutiva*.



① O código de endereço 011 é colocado no barramento de endereço e o endereço 3 é selecionado.
② O comando de leitura é aplicado.
③ O conteúdo do endereço 3 é colocado no barramento de dados e deslocado no registrador de dados. O conteúdo do endereço 3 não é apagado pela operação de leitura.

◀ **FIGURA 10–6**
Ilustração da operação de leitura.

RAMs e ROMs

As duas principais categorias das memórias semicondutoras são a RAM e a ROM. A **RAM** (*random-access memory* – memória de acesso aleatório), é um tipo de memória na qual todos os endereços são acessados em tempos iguais e podem ser selecionados em qualquer ordem para uma operação de leitura ou escrita. Todas as RAMs apresentam a capacidade de *leitura* e *escrita*. Devido as RAMs perderem os dados armazenados quando a alimentação é desligada, elas são memórias **voláteis**.

A **ROM** (*read-only memory* – memória apenas de leitura) é um tipo de memória na qual os dados são armazenados permanentemente ou semi-permanentemente. Dados podem ser lidos da ROM, porém não existe operação de escrita como na RAM. A ROM, assim como a RAM, é uma memória de acesso aleatório, mas o termo *RAM* tradicionalmente significa uma memória de *leitura/escrita* de acesso aleatório. Vários tipos de RAMs e ROMs são abordados nesse capítulo. Devido as ROMs manterem os dados armazenados mesmo se a alimentação for desligada, elas são memórias **não-voláteis**.

SEÇÃO 10–1 REVISÃO
As respostas estão no final do capítulo.

1. Qual é a menor unidade de dado que pode ser armazenada em uma memória?
2. Qual é a capacidade de bit de um memória que pode armazenar 256 bytes de dados?
3. O que é uma operação de escrita?
4. O que é uma operação de leitura?
5. Como uma determinada unidade de dado é armazenada em uma memória?
6. Descreva a diferença entre RAM e ROM.

10-2 MEMÓRIAS DE ACESSO ALEATÓRIO (RAMs)

As RAMs são memórias nas quais os dados podem ser escritos ou lidos a partir de qualquer endereço selecionado em qualquer seqüência. Quando uma unidade de dados é escrita num determinado endereço numa RAM, a unidade de dado armazenada anteriormente nesse endereço é substituída pela nova unidade de dado. Quando uma unidade de dado é lida a partir de um determinado endereço na RAM, a unidade de dado permanece armazenada e não é apagada pela operação de leitura. Essa operação de leitura não destrutiva pode ser vista como uma cópia do conteúdo de um endereço enquanto deixa o conteúdo intacto. Uma RAM é usada tipicamente para armazenamento de dados de curta duração porque ela não pode manter os dados quando a alimentação é desligada.

Ao final do estudo desta seção você deverá ser capaz de:

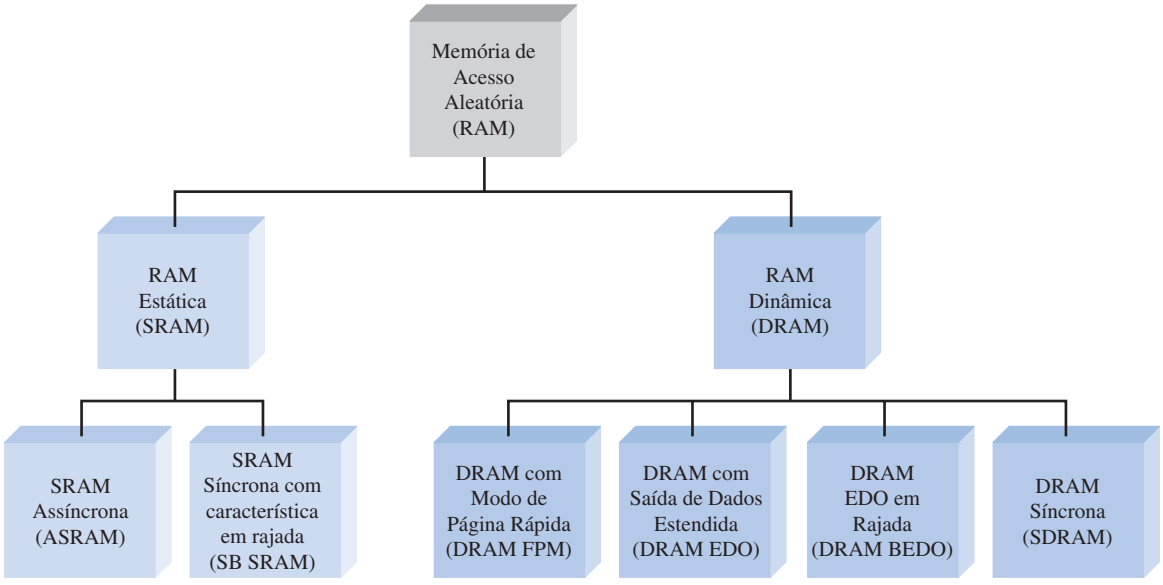
■ Citar as duas categorias de RAM ■ Explicar o que é uma SRAM ■ Descrever a célula de armazenamento de uma SRAM ■ Explicar a diferença entra uma SRAM assíncrona e uma SRAM em rajada ■ Explicar o que é uma DRAM ■ Descrever a célula de armazenamento de uma DRAM ■ Discutir os tipos de DRAM ■ Comparar a SRAM com a DRAM

Família de RAMs

As duas categorias de RAM são a *RAM estática* (SRAM) e a *RAM dinâmica* (DRAM). As RAMs estáticas geralmente usam latches como elemento de armazenamento e, portanto, podem armazenar dados indefinidamente *enquanto a tensão de alimentação estiver presente*. As RAMs dinâmicas usam capacitores como elementos de armazenamento e não podem reter os dados por muito tempo sem que o capacitor seja recarregado por um processo denominado renovação (*refresh*). Tanto as RAMs estáticas quanto as RAMs dinâmicas perdem os dados quando a alimentação é removida e, portanto, são classificadas como memórias voláteis.

Os dados podem ser lidos tão rapidamente de uma SRAM quanto de uma DRAM. Entretanto, as DRAMs podem armazenar mais dados que as SRAMs para um dado tamanho físico e custo porque as células de DRAM são muito mais simples, e mais células podem ser preenchidas numa determinada área do chip do que para uma SRAM.

Os tipos básicos de SRAM são as *SRAMs assíncronas* e as *SRAMs síncronas* com uma característica de rajada. Os tipos básicos de DRAM são as *DRAMs com Modo de Página Rápida* (FPM DRAM – *Fast Page Mode DRAM*), a *DRAM com Saída de Dados Estendida* (DRAM EDO – *Extended Data Out*), a *DRAM EDO em Rajada* (DRAM BEDO) e as *DRAMs síncronas* (SDRAM). Esses tipos são mostrados na Figura 10–7.



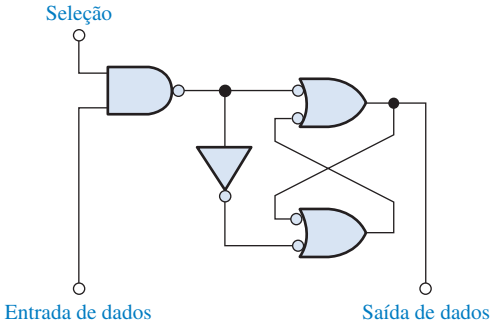
▲ **FIGURA 10–7**

A família RAM.

RAMs Estáticas (SRAMs)

Célula de Memória Todas as RAMs estáticas são caracterizadas por células de memória latch. Enquanto a tensão de alimentação estiver aplicada numa célula de **memória estática**, ela retém indefinidamente o estado 0 ou 1. Se a alimentação for removida, o bit de dado armazenado é perdido.

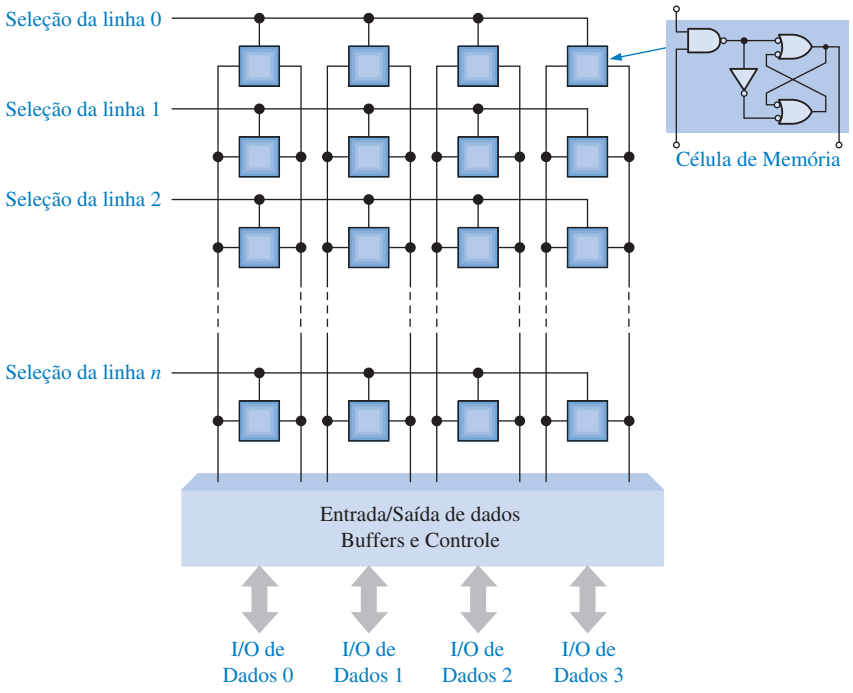
A Figura 10–8 mostra uma célula de memória latch **SRAM** básica. A célula é selecionada por um nível ativo na linha de seleção e um bit de dado (1 ou 0) é escrito na célula colocando-o na linha de entrada de dados. Um bit de dado é lido a partir da linha de saída de dados.



◀ **FIGURA 10–8**

Uma típica célula de memória latch SRAM.

Arranjo Básico de Células de Memória Estática As células de memória numa SRAM são organizadas em linhas e colunas, conforme ilustrado na Figura 10–9 para o caso de um arranjo de $n \times 4$. Todas as células numa linha compartilham a mesma seleção de linha. Cada conjunto de linhas de entrada de dados e saída de dados se conectam a cada célula numa dada coluna e estão conectadas a uma única linha de dados que serve como entrada (*in*) e saída (*out*) de dados (I/O de Dados) através dos buffers de entrada e saída de dados.

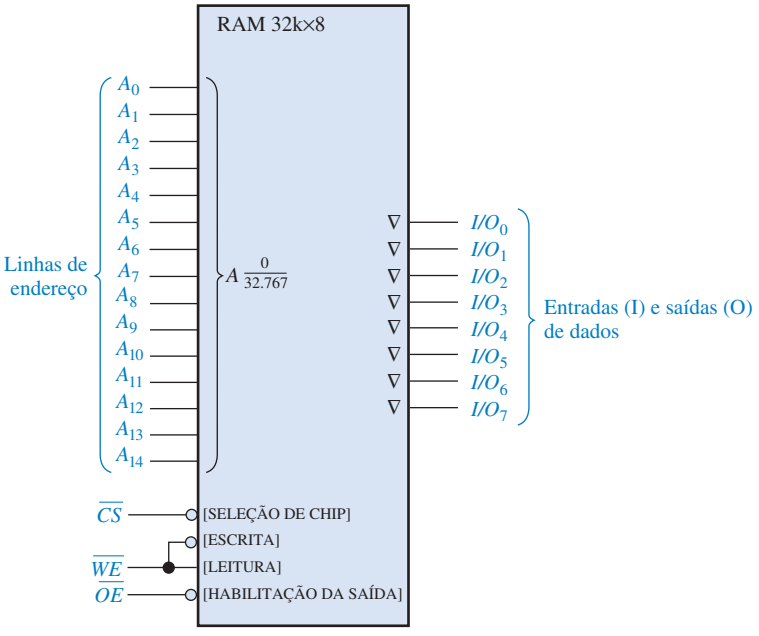


▶ **FIGURA 10–9**
Arranjo básico de uma SRAM.

Para escrever uma unidade de dados, nesse caso um nibble, numa determinada linha de células no arranjo de memória, a seleção de linha é colocada no seu estado ativo e quatro bits de dados são colocados nas linhas de I/O de Dados. O sinal de escrita é então colocado no seu estado ativo, o que faz com que cada bit de dado seja armazenado na célula selecionada na coluna associada. Para ler uma unidade de dado, a linha de leitura é colocada no seu estado ativo, o que faz com que os quatro bits armazenados na linha selecionada apareçam nas linhas de I/O de Dados.

Organização Básica de uma SRAM Assíncrona

Uma SRAM assíncrona é aquela na qual a operação não está sincronizada com um sistema de clock. Para ilustrar a organização geral de uma SRAM, é usada uma memória de 32k × 8 bits. Um símbolo lógico para essa memória é mostrado na Figura 10–10.



▶ **FIGURA 10–10**
Diagrama lógico para uma SRAM assíncrona de 32k × 8.

No modo LEITURA, os oito bits de dados que são armazenados no endereço selecionado aparecem nas linhas de saída de dados. No modo ESCRITA, os oito bits de dados que são aplicados nas linhas de entradas de dados são armazenados no endereço selecionado. As entradas e saídas de dados (I/O_0 a I/O_7) compartilham as mesmas linhas. Durante a LEITURA, elas funcionam como saídas (O_0 a O_7) e durante a ESCRITA elas funcionam como entradas (I_0 a I_7).

Saídas e Barramentos Tristate Os buffers *tristate* nas memórias permitem que as linhas de dados funcionem como entrada ou saída e conectem a memória ao barramento de dados de um computador. Esses buffers têm três estados de saída: ALTO (1), BAIXO (0) e alta impedância (aberto). As saídas *tristate* são indicadas pelos símbolos lógicos por um pequeno triângulo invertido (∇), como mostra a Figura 10–10 e são usadas para compatibilização com estruturas de barramentos tais como os que são encontrados em sistemas microprocessados.

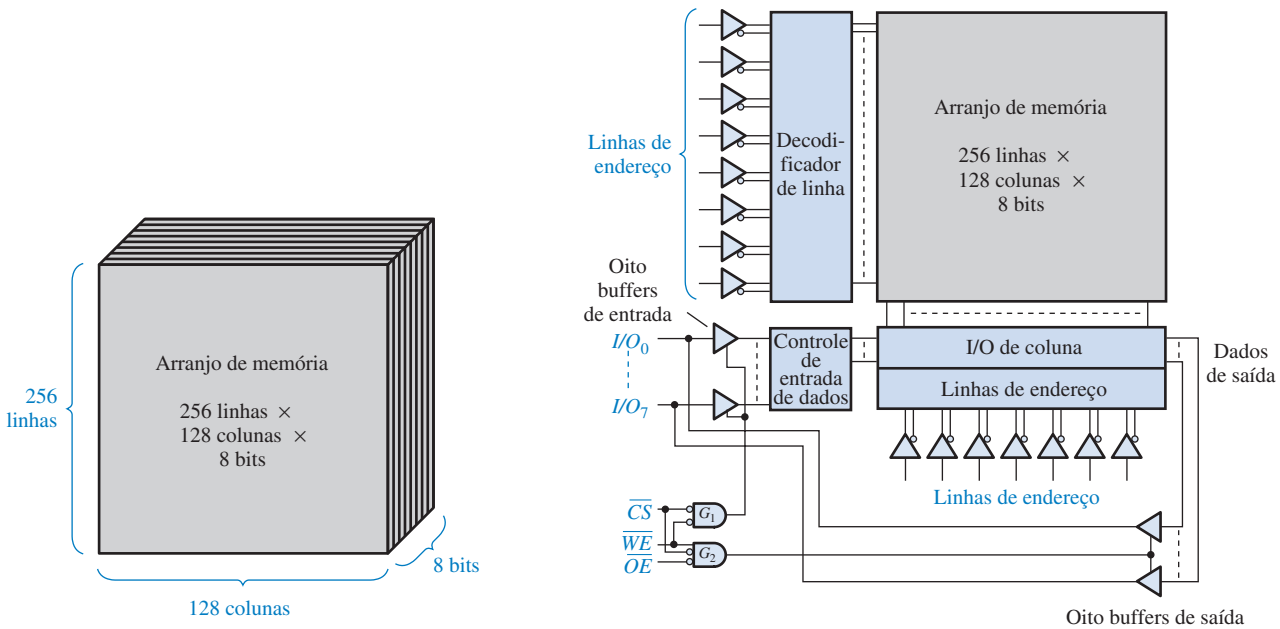
Fisicamente, um **barramento** é um conjunto de percursos condutores que servem para interconectar dois ou mais componentes funcionais de um sistema ou vários sistemas diversos. Eletricamente, um barramento é uma coleção de níveis de tensão e/ou corrente específicos e sinais que permitem que os vários dispositivos conectados ao barramento de dados se comuniquem e funcionarem corretamente juntos.

Por exemplo, um microprocessador é conectado às memórias e dispositivos de entrada/saída através de certas estruturas de barramentos. Um barramento de endereço permite que o microprocessador enderece as memórias e o barramento de dados provê a transferência de dados entre o microprocessador, as memórias e os dispositivos de entrada/saída tais como monitores, impressoras, teclados e modems. O barramento de controle permite ao microprocessador controlar a transferência de dados e a temporização para os diversos componentes.

Arranjo de Memória Chips de SRAM podem ser organizados em bits isolados, nibbles (4 bits), bytes (8 bits) ou múltiplos bytes (16, 24, 32 bits, etc.).

A Figura 10–11 mostra a organização de uma SRAM típica de 32k × 8. O arranjo de células de memória está organizado em 256 linhas e 128 colunas, cada uma com 8 bits, como mostrado em (a). Na verdade, existem 2^{15}=32.728 endereços, e cada um contém 8 bits. A capacidade da memória desse exemplo é 32.768 bytes (tipicamente expressa como 32 kB).

A SRAM na Figura 10–11(b) funciona como descrito a seguir. Primeiro, a entrada de seleção do chip $\overline{CS}$, tem que ser nível BAIXO para que a memória opere. Oito das 15 linhas de endereço



(a) Configuração de um arranjo de memória

(b) Diagrama em bloco da memória

▲ **FIGURA 10–11**

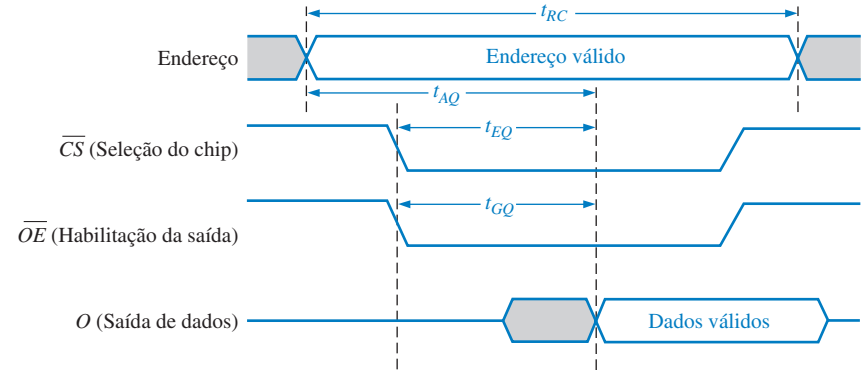
Organização de uma SRAM assíncrona de 32k × 8.

são decodificadas pelo decodificador de linha para selecionar uma das 256 linhas. Sete das quinze linhas de endereço são decodificadas pelo decodificador de coluna para selecionar uma das 128 colunas de 8 bits.

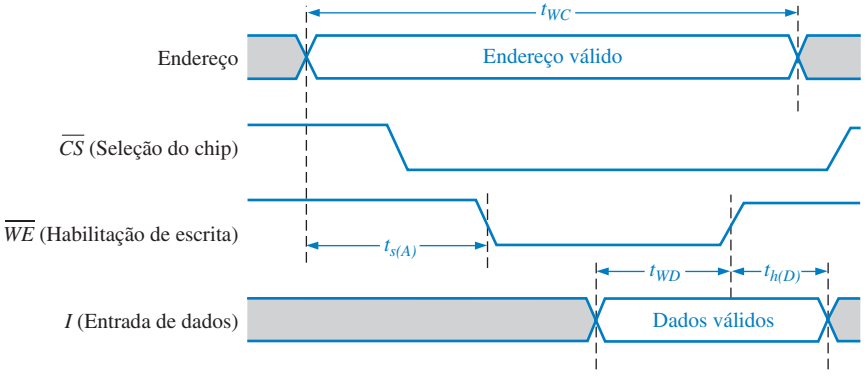
Leitura No modo LEITURA, a entrada de habilitação de escrita $\overline{WE}$ é nível ALTO e a habilitação de saída $\overline{OE}$ é nível BAIXO. Os buffers tristate de entrada são desabilitados pela porta G_1 e os buffers tristate de saída da coluna são habilitados pela porta G_2. Portanto, os oito bits de dados do endereço selecionado são encaminhados através das colunas I/O para as linhas de dados (I/O_0 a I/O_7), que atuam como linhas de saída de dados.

Escrita No modo de escrita, $\overline{WE}$ é nível BAIXO e $\overline{OE}$ é nível ALTO. Os buffers de entrada são habilitados pela porta G_1 e os buffers de saída são desabilitados pela porta G_2. Portanto, os oito bits de dados de entrada nas linhas de dados são encaminhados através dos blocos de controle de dados de entrada e I/O de coluna para o endereço selecionado e armazenados.

Ciclos de Leitura e Escrita A Figura 10–12 mostra os diagramas de temporização para um ciclo de leitura e um ciclo de escrita na memória. Para o ciclo de leitura mostrado na parte (a), um código de endereço válido é aplicado nas linhas de endereço para um intervalo de tempo especificado denominado *tempo do ciclo de leitura* (t_{RC} – *read cycle time*). Em seguida os sinais de seleção do chip ($\overline{CS}$) e habilitação da saída ($\overline{OE}$) vão para o nível BAIXO. Um intervalo de tempo após a entrada $\overline{OE}$ ir para nível BAIXO, um byte de dados válido a partir do endereço selecionado aparece nas linhas de dados. Esse intervalo de tempo é denominado *tempo de acesso de habilitação da saída* (t_{GQ}). Dois outros tempos de acesso para o ciclo de leitura são o *tempo de acesso de endereço* (t_{AQ}), medido a partir do início de um endereço válido até que um dado válido apareça nas linhas de dados, e o *tempo de acesso de habilitação do chip* (t_{EQ}), medido a partir da transição de nível ALTO para nível BAIXO de $\overline{CS}$ até o aparecimento de um dado válido nas linhas de dados.



(a) Ciclo de leitura ($\overline{WE}$ em nível ALTO)



(b) Ciclo de escrita ($\overline{WE}$ em nível BAIXO)

► **FIGURA 10–12**

Temporização básica dos ciclos de leitura e escrita para a SRAM dada na Figura 10–11.

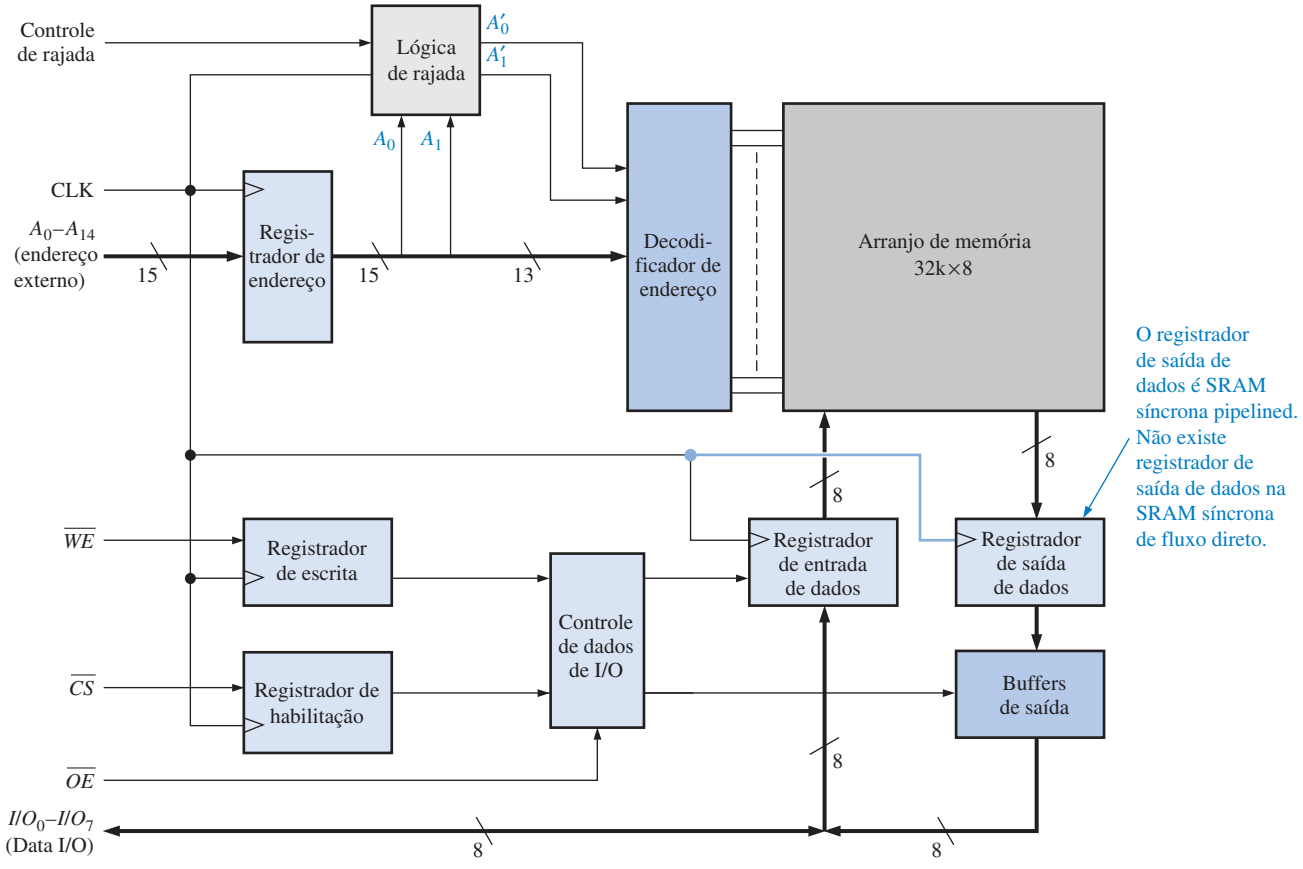
Para o ciclo de escrita mostrado na Figura 10–12(b), um código de endereço válido é aplicado nas linhas de endereço por um intervalo de tempo especificado denominado *tempo do ciclo de escrita* (t_{WC}). Em seguida, as entradas de seleção do chip ($\overline{CS}$) e habilitação de escrita ($\overline{WE}$) vão para nível BAIXO. O intervalo de tempo necessário a partir do início de um endereço válido até o instante em que a entrada $\overline{WE}$ pode ir para nível BAIXO é denominado *tempo de setup de endereço* ($t_{s(A)}$). O tempo no qual a entrada $\overline{WE}$ tem que ser nível BAIXO é a largura do pulso de escrita. O tempo no qual a entrada $\overline{WE}$ tem que ser mantida em nível BAIXO após um dado válido ser aplicado nas entradas de dados é designado com t_{WD}; o tempo no qual um dado válido tem que ser mantido nas linhas de dados após a entrada $\overline{WE}$ ir para nível ALTO é o *tempo de manutenção do dado* ($t_{h(D)}$).

Durante cada ciclo de escrita, uma unidade de dado é escrita na memória.

RAM Síncrona Básica com Característica de Rajada

Diferentemente da SRAM assíncrona, a SRAM síncrona é sincronizada com o sistema de clock. Por exemplo, num sistema de computador, a SRAM síncrona opera com o mesmo sinal de clock que o microprocessador de forma que o microprocessador e a memória estejam sincronizados para uma operação mais rápida.

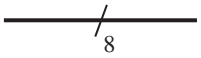
O conceito fundamental da característica síncrona de uma SRAM pode ser mostrado com a Figura 10–13, a qual apresenta um diagrama em bloco simplificado de uma memória de 32k × 8 para fins de ilustração. A SRAM síncrona é similar à SRAM assíncrona em termos do arranjo de memória, do decodificador de endereço e das entradas de leitura/escrita e habilitação. A diferença básica é que a SRAM síncrona usa registradores com clock para sincronizar todas as entradas com o clock do sistema. O endereço, a entrada de leitura/escrita, a habilitação do chip e os dados de entrada são todos armazenados nos respectivos registradores com a borda ativa do pulso de clock. Uma vez que essas informações são armazenadas, a operação da memória está em sincronismo com o clock.



▲ **FIGURA 10–13**

Um diagrama em bloco básico de uma SRAM síncrona com característica de rajada.

Com a finalidade de simplificação, uma notação para múltiplas linhas paralelas ou barramento de linhas é introduzida na Figura 10–13, como uma alternativa para o desenho de cada uma das linhas separadamente. Um conjunto de linhas paralelas pode ser indicado por uma única linha mais grossa com um corte junto ao número de linhas do conjunto. Por exemplo, a notação a seguir representa um conjunto de 8 linhas paralelas:

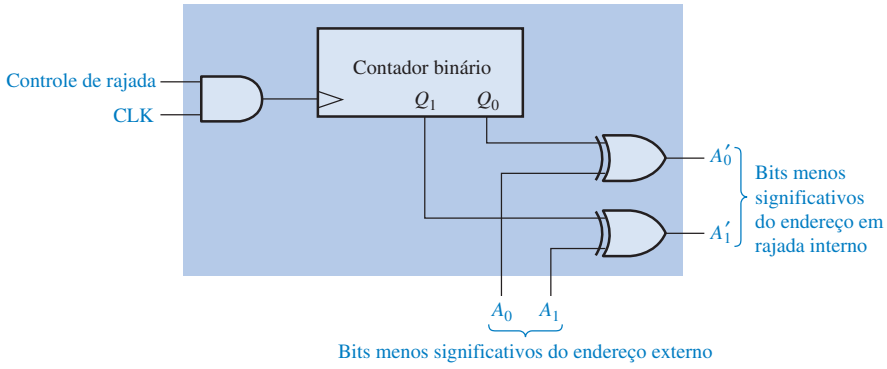


Os bits de endereço de A_0 a A_{14} são armazenados no registrador de endereço na borda positiva do pulso de clock. No mesmo pulso de clock, o estado da linha de habilitação da escrita ($\overline{WE}$) e de seleção do chip ($\overline{OE}$) são armazenados no registrador de escrita e no registrador de habilitação respectivamente. Esses são registradores de um bit ou simplesmente flip-flops. Além disso, no mesmo pulso de clock os dados de entrada são armazenados no registrador de entrada de dados numa operação de escrita e os dados no endereço de memória selecionado são armazenados no registrador de saída de dados para uma operação de leitura, conforme determinado pelo controle de I/O de dados baseado nas entradas do registrador de escrita, do registrador de habilitação e da habilitação de saída ($\overline{OE}$).

Dois tipos básicos de SRAM síncrona são a de *fluxo direto* e a *pipelined*. A SRAM síncrona de fluxo direto não tem um registrador de saída de dados, assim os dados de saída passam de forma assíncrona para as linhas de I/O através dos buffers de saída. A SRAM síncrona **pipelined** tem um registrador de saída de dados, conforme mostra a Figura 10–13, assim as saídas de dados são colocadas de forma síncrona nas linhas de I/O.

A característica de Rajada Conforme mostra a Figura 10–13, as SRAMs síncronas normalmente têm uma característica de endereço em rajada, o que permite ler ou escrever na memória até quatro posições usando um único endereço. Quando um endereço externo é armazenado no registrador de endereço, os dois bits menos significativos do endereço (A_0 e A_1) são aplicados no circuito lógico de rajada. Isso produz uma seqüência de quatro endereços internos somando 00, 01, 10 e 11 aos bits menos significativos do endereço nos pulsos de clock sucessivos. A seqüência sempre começa com um endereço base, que é o endereço externo mantido no registrador de endereço.

O circuito lógico de rajada de endereço numa SRAM síncrona típica consiste de um contador binário e portas EX-OR, como mostra a Figura 10–14. Para um circuito lógico de rajada de 2 bits, a seqüência do endereço de rajada interno é formada pelos bits do endereço base ($A_2 - A_{14}$) mais os dois bits do endereço em rajada (A_1' e A_2').



▶ **FIGURA 10–14**
Lógica de endereçamento de rajada.

Para começar a seqüência de rajada, o contador está no seu estado 00 e os dois bits menos significativos do endereço são aplicados nas entradas das portas EX-OR. Considerando que A_0 e A_1 sejam ambos nível 0, a seqüência de endereço interno em termos dos seus dois bits menos significativos é 00, 01, 10 e 11.

Memória Cache

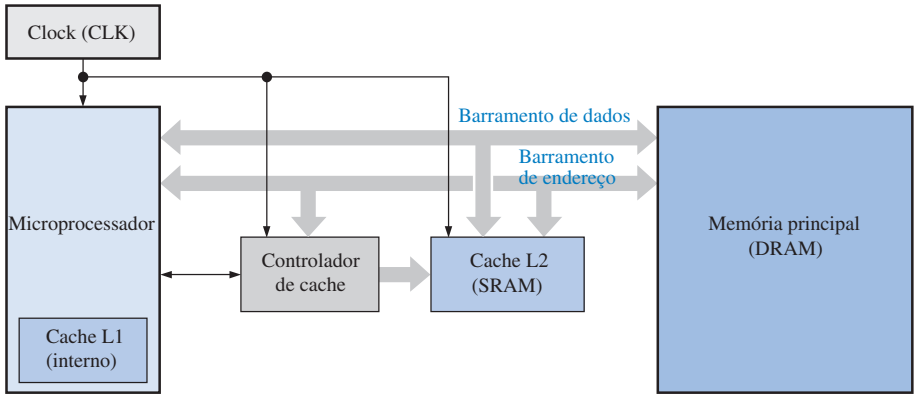
Uma das principais aplicações de SRAMs é na memória cache dos computadores. A **memória cache** é relativamente pequena, de alta velocidade e que armazena as instruções ou dados mais recentes usados a partir da memória principal que é maior, porém mais lenta. A memória cache tam-

bém pode usar RAM dinâmica (DRAM) que será abordada logo a seguir. Tipicamente, uma SRAM é várias vezes mais rápida que uma DRAM. Em geral, a memória cache consegue informações armazenadas para o microprocessador de uma forma mais rápida que se fosse usada apenas a DRAM de alta capacidade. A memória cache é basicamente um método de custo efetivo de melhorar o desempenho de um sistema sem recorrer a opção cara de ter toda a memória rápida.

O conceito de memória cache é baseado na idéia de que os programas de computadores tendem a obter instruções ou dados a partir de uma área da memória principal antes de se mover para uma outra área. Basicamente, o controlador de cache "supõe" que área da memória dinâmica mais lenta a CPU (unidade central de processamento) necessitará em seguida e a move para a memória cache de forma que esses dados estarão pontos quando necessário. Se o controlador de cache supor corretamente, os dados estarão imediatamente disponíveis para o microprocessador. Caso o controlador de cache faça uma dedução errada, a CPU terá que acessar a memória principal e esperar muito mais tempo pelas instruções ou dados corretos. Felizmente, o controlador de cache acerta na maioria das vezes.

Analogia para a Cache Existem diversas analogias que podem ser usadas para descrever a memória cache, mas a comparação com uma geladeira talvez seja a mais efetiva. Uma geladeira pode ser vista como uma "cache" para determinados alimentos enquanto que o supermercado se compara à memória principal onde todos os alimentos são guardados. Cada vez que desejamos comer ou beber alguma coisa, vamos à geladeira (cache) primeiro para ver se o item desejado está lá. Caso esteja, economizamos tempo. Caso contrário, teremos que gastar um tempo extra para ir ao supermercado (memória principal).

Cachês L1 e L2 Um primeiro nível de cache (L1) geralmente é integrado ao chip do processador tendo uma capacidade de armazenamento muito limitada. A cache L1 também é conhecida com cache *primária*. Um segundo nível de cache (cache L2) é um chip de memória separado ou um conjunto de chips externos ao processador e geralmente tem uma capacidade maior que um cache L1. O cache L2 também é conhecido como cache *secundária*. Alguns sistemas podem ter cachês de níveis maiores (L3, L4, etc.), mas L1 e L2 são os mais comuns. Além disso, alguns sistemas usam um cache de disco para melhorar o desempenho do disco rígido porque a DRAM, embora muito mais lenta que a SRAM, é muito mais rápida que um drive de disco rígido. A Figura 10–15 ilustra as memórias cache L1 e L2 num sistema de computador.

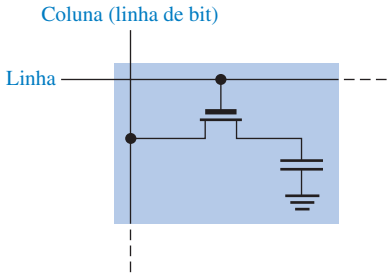


◀ **FIGURA 10–15**
Diagrama em bloco mostrando as memórias cache L1 e L2 num sistema de computador.

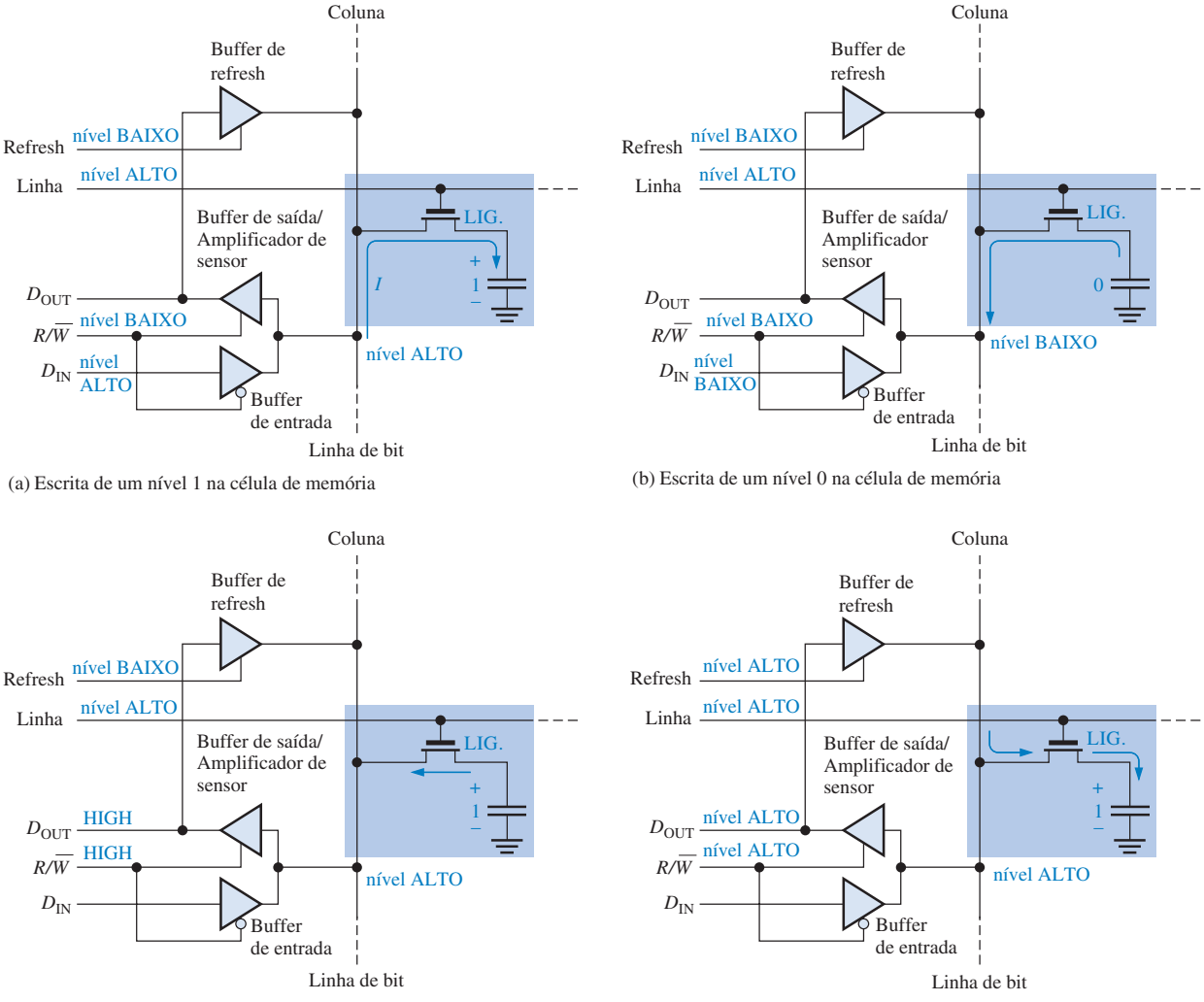
Células de Memória RAM Dinâmica (DRAM)

Células de **memória dinâmica** armazenam um bit de dado num pequeno capacitor em vez de um latch. A vantagem desse tipo de célula é que ela é muito simples, permitindo portanto que sejam construídos arranjos de memórias muito maiores num chip com um menor custo por bit. A desvantagem é que o capacitor não pode manter sua carga por um grande tempo, perdendo o bit de dado armazenado, a menos que a carga seja renovada (operação de refresh) periodicamente. A operação de refresh requer um circuito adicional na memória e complica a operação da **DRAM**. A Figura 10–16 mostra uma célula de DRAM típica que consiste de um único transistor MOS (MOSFET) e um capacitor.

Nesse tipo de célula, o transistor funciona como uma chave. A operação básica simplificada é ilustrada na Figura 10–17 e é apresentada a seguir. Um nível BAIXO na linha $R/\overline{W}$ (modo ES-



▶ **FIGURA 10–16**
Uma célula de uma DRAM MOS.



▲ **FIGURA 10–17**
Operação básica de uma célula DRAM.

CRITA) habilita o buffer de entrada tristate e desabilita o buffer de saída. Para um nível 1 ser escrito na célula, a linha D_{IN} tem que ser nível ALTO e o transistor tem que ser ligado por um nível ALTO na LINHA da matriz da memória. O transistor funciona como uma chave fechada conectando o capacitor à linha de bit. Essa conexão permite que o capacitor seja carregado com uma tensão positiva, conforme mostra a Figura 10–17(a). Quando um nível 0 é armazenado, um nível BAIXO é aplicado na linha D_{IN}. Se o capacitor estiver armazenando um nível 0, ele permanece inalterado, ou se ele estiver armazenando um nível 1, o capacitor se descarrega conforme indicado na Figura 10–17(b). Quando a LINHA volta para nível BAIXO, o transistor desliga e desconecta o capacitor da linha de bit, "prendendo" portanto a carga (1 ou 0) no capacitor.

Para ler a célula, a linha $R/\overline{W}$ (Read/$\overline{\text{Write}}$) deve ser nível ALTO, habilitando o buffer de saída e desabilitando o buffer de entrada. Quando a LINHA é colocada em nível ALTO, o transistor liga e conecta o capacitor na linha de bit e assim ao buffer de saída (amplificador sensor), portanto o bit de dado aparece na linha de saída de dados (D_{OUT}). Esse processo é ilustrado na Figura 10–17(c).

Para o refresh na célula de memória, a linha $R/\overline{W}$ deve ser nível ALTO, a LINHA da matriz deve ser nível ALTO e a linha de refresh deve ser também nível ALTO. O transistor liga, conectando o capacitor na linha de bit. O buffer de saída é habilitado e o bit de dado armazenado é aplicado na entrada do buffer de refresh, que é habilitado pelo nível ALTO na entrada refresh. Isso gera uma tensão na linha de bit que corresponde ao bit armazenado repondo a carga, se for o caso, do capacitor. Isso está ilustrado na Figura 10–17(d).

Organização Básica de uma DRAM

A principal aplicação das DRAMs é como memória principal de computadores. A diferença entre DRAMs e SRAMs está no tipo da célula da memória. Conforme vimos, a célula de uma memória DRAM consiste de um transistor e um capacitor sendo muito mais simples que a célula de uma SRAM. Isso permite uma densidade maior no chip de uma DRAM resultando em maiores capacidades de bits para uma dada área no chip, embora com um tempo de acesso bem menor.

Novamente, por causa da perda de carga armazenada pelo capacitor, a célula de uma DRAM necessita de operações de refresh freqüentes para preservar o bit de dado armazenado. Esse requisito implica num circuito com maior complexidade que o de uma SRAM. Discutiremos agora várias características comuns à maioria das DRAMs usando uma DRAM de 1M × 1 bit como exemplo.

Multiplexação de Endereço As DRAMs usam uma técnica denominada de *multiplexação de endereço* para reduzir o número de linhas de endereço. A Figura 10–18 mostra o diagrama em blo-

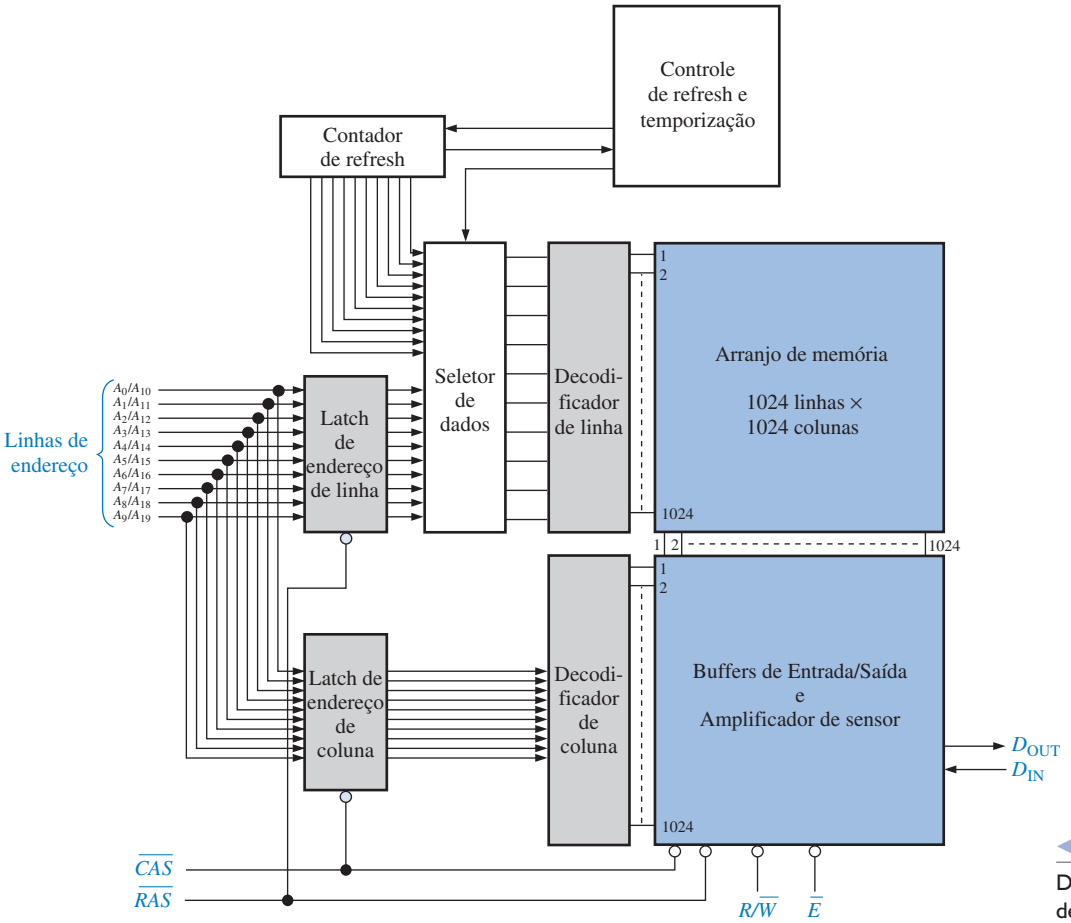
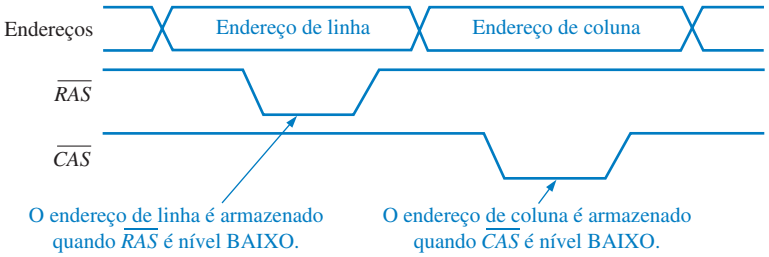


FIGURA 10–18
Diagrama em bloco simplificado de uma DRAM de 1M × 1.

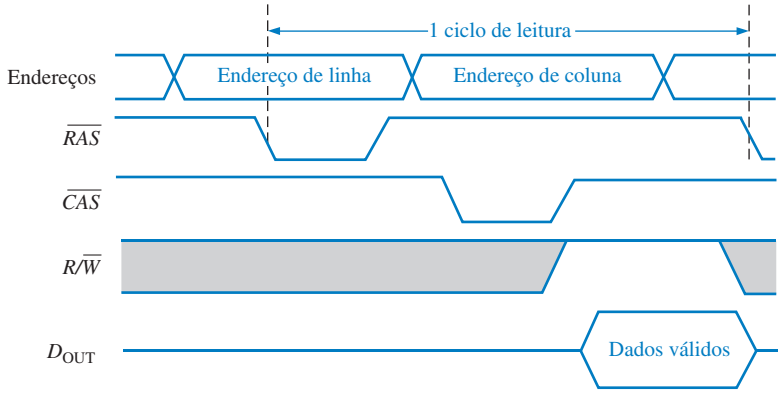
co de uma DRAM com 1.048.576 bits (1 Mbit) sendo a organização de 1M × 1. Vamos nos concentrar nos blocos cor de laranja para ilustrar a multiplexação de endereço. Os blocos em branco representam o circuito lógico de refresh.

As dez linhas de endereço são multiplexadas no tempo no início do ciclo da memória através da seleção de endereço de linha $(\overline{RAS})$ e seleção de endereço de coluna $(\overline{CAS})$ em dois campos separados de endereço de 10 bits. Primeiro os 10 bits de endereço de linha são armazenados no latch de endereço de linha. Em seguida, os 10 bits de endereço de coluna são armazenados no latch de endereço de coluna. O endereço de linha e o endereço de coluna são decodificados para selecionar um dos 1.048.576 endereços (2^{20} = 1.048.576) no arranjo de memória. A temporização básica para a operação de multiplexação de endereço é mostrada na Figura 10–19.

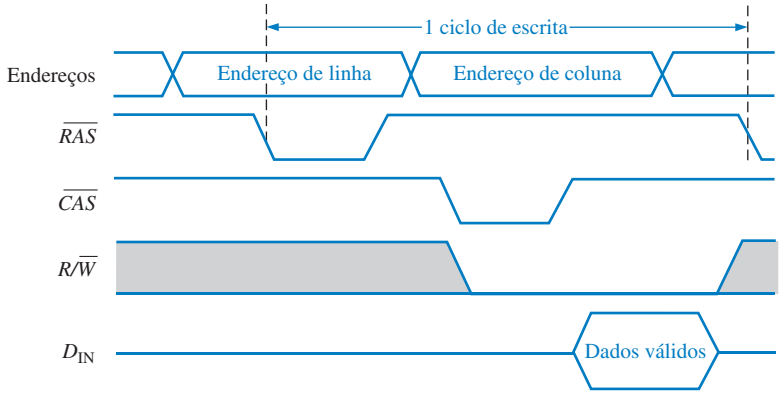


► FIGURA 10–19
Temporização básica para a multiplexação de endereço.

Ciclos de Leitura e Escrita No início de cada ciclo de leitura ou escrita na memória, $\overline{RAS}$ e $\overline{CAS}$ vão para o estado ativo (nível BAIXO) para multiplexar os endereços de linhas e colunas nos latches e decodificadores. Para cada ciclo de leitura, a entrada $R/\overline{W}$ é nível ALTO. Para um ciclo de escrita, a entrada $R/\overline{W}$ é nível BAIXO. Isso está ilustrado na Figura 10–20.



(a) Ciclo de leitura



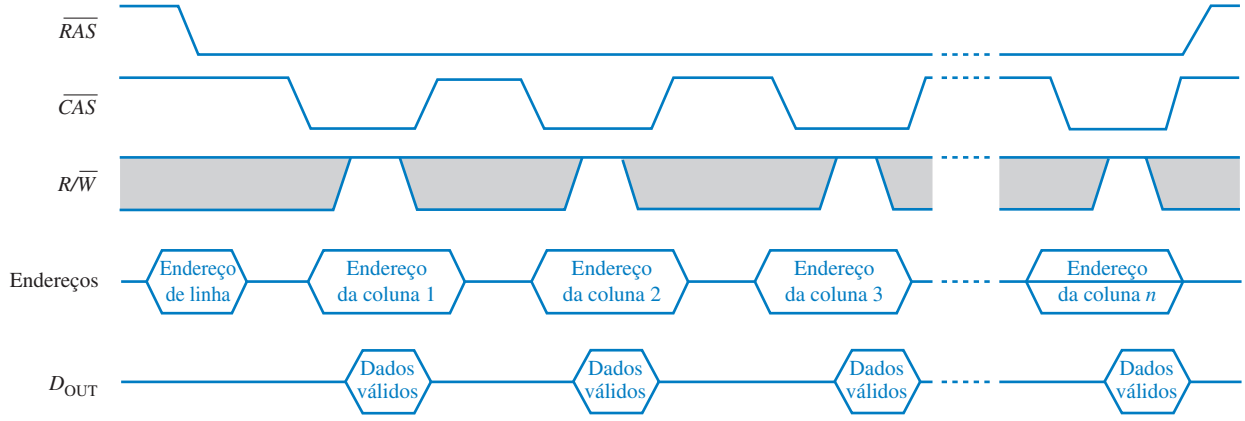
(b) Ciclo de escrita

► FIGURA 10–20
Temporização dos ciclos de leitura e escrita normais.

Modo de Página Rápida No ciclo de leitura ou escrita normal descrito anteriormente, o endereço de linha de uma posição particular de memória é carregado primeiro ativando a entrada $\overline{RAS}$ (com um nível BAIXO) e em seguida o endereço de coluna para a posição é carregado ativando a entrada $\overline{CAS}$. (com um nível BAIXO). A próxima posição é selecionada por um outro $\overline{RAS}$ seguido de um $\overline{CAS}$., e assim por diante.

Uma "página" é uma seção da memória disponível para uma única linha de endereço e consiste de todas as colunas em uma linha. O modo de página rápida permite operações rápidas de leitura ou escrita sucessivas em cada endereço de coluna numa linha selecionada. Um endereço de linha é carregado primeiro fazendo $\overline{RAS}$ nível BAIXO e mantendo assim enquanto $\overline{CAS}$. comuta entre nível ALTO e nível BAIXO. Uma única linha de endereço é selecionada e permanece selecionada enquanto $\overline{RAS}$ é ativado. Cada $\overline{CAS}$. sucessivo seleciona uma outra coluna na linha selecionada. Assim, após um ciclo no modo de página rápida, todos os endereços na linha selecionada serão lidos ou escritos, dependendo de $R/\overline{W}$. Por exemplo, um ciclo no modo de página rápida para a DRAM dada na Figura 10–18 requer que $\overline{CAS}$. seja ativado 1024 vezes para cada linha selecionada por $\overline{RAS}$.

A operação no modo de página rápida para leitura é ilustrada por um diagrama de temporização que é visto na Figura 10–21. Quando $\overline{CAS}$. vai para o seu estado ativo (nível ALTO), desabilita a saída de dados. Portanto, a transição de $\overline{CAS}$. para nível ALTO tem que ocorrer apenas após os dados válidos serem armazenados pelo sistema externo.



▲ **FIGURA 10–21**

Temporização do modo de página rápida para a operação de leitura.

Ciclos de Refresh Conforme sabemos, as DRAMs são baseadas no armazenamento de carga em capacitor para cada bit no arranjo da memória. Essa carga se degrada (fuga de carga) com o tempo e a temperatura, assim cada bit tem que ser periodicamente reavivado (recarregado) para manter o estado correto do bit. Tipicamente, uma DRAM tem que ser reavivada entre 8 ms a 16 ms, embora o refresh para alguns dispositivos possa exceder 100 ms.

Uma operação de leitura automaticamente reaviva todos os endereços na linha selecionada. Entretanto, em aplicações típicas, não podemos prever sempre quando haverá um ciclo de leitura e dessa forma não podemos depender que um ciclo de leitura ocorra numa freqüência suficiente para evitar a perda de dados. Portanto, ciclos de refresh especiais têm que ser implementados em sistemas de DRAM.

Os dois modos básicos de refresh são o *refresh em rajada* e o *refresh distribuído*. No refresh em rajada, todas as linhas no arranjo da memória são reavivadas consecutivamente a cada período de refresh. Para uma memória com um período de refresh de 8 ms, um refresh em rajada de todas as linhas ocorre uma vez a cada 8 ms. As operações de leitura e escrita normais são suspensas durante um ciclo de refresh em rajada. No refresh distribuído, cada linha é reavivada em intervalos intercalados entre ciclos de leitura e escrita normais. Por exemplo, a memória vista na Figura 10–18 tem 1024 linhas. Como exemplo, para um período de refresh de 8 ms, cada linha tem que ser reavivada a cada 8 ms/1024 = 7,8μs quando for usado o refresh distribuído.

Os dois tipos de operações de refresh são *refresh apenas com* $\overline{RAS}$ e *refresh* $\overline{CAS}$ *antes de* $\overline{RAS}$. O refresh apenas com $\overline{RAS}$ consiste de uma transição de um $\overline{RAS}$ para o estado BAIXO

(ativo), o qual armazena o endereço de uma linha para ser reavivada enquanto $\overline{CAS}$ permanece em nível ALTO (inativo) ao longo do ciclo. Um contador externo é usado para prover os endereços de linha para esse tipo de operação.

O refresh de $\overline{CAS}$ antes de $\overline{RAS}$ é iniciado por um $\overline{CAS}$ levado ao nível BAIXO antes $\overline{RAS}$ ir para nível BAIXO. Essa seqüência ativa um contador de refresh interno que gera o endereço da linha a ser reavivada. Esse endereço é comutado pelo seletor de dados no decodificador de endereço.

Tipos de DRAMs

Agora que aprendemos os conceitos básicos das DRAMs, vamos conhecer de forma resumida os principais tipos, os quais são: *DRAM com Modo de Página Rápida (FPM)*, *DRAM com Saída de Dados Estendida (EDO)*, *DRAM com Saída de Dados Estendida em Rajada (BEDO)* e *DRAM Síncrona (S)*.

DRAM FPM A operação de modo de página rápida foi descrita anteriormente. Esse tipo de DRAM foi tradicionalmente o mais comum, usado em computadores antes do desenvolvimento da DRAM EDO. Lembre que uma página na memória é todos os endereços de coluna contidos dentro de um endereço de linha.

Essa idéia básica da **DRAM FPM** é baseada na probabilidade de que os próximos endereços de memória a serem acessados estão na mesma linha (na mesma página). Felizmente, isso acontece em grande parte do tempo. A memória FPM economiza tempo em relação a um acesso aleatório puro porque na memória FPM o endereço de linha é especificado apenas uma vez para o acesso a diversos endereços sucessivos de coluna ao passo que para um acesso aleatório puro, um endereço de linha é especificado para cada endereço de coluna.

Lembre-se de que na operação de modo de página rápida, o sinal $\overline{CAS}$ tem que esperar até que o dado válido de um determinado endereço seja aceito (armazenado) por um sistema externo (CPU) antes que ele possa ir para o seu estado desativado. Quando $\overline{CAS}$ vai para o estado desativado, as saídas de dados são desabilitadas. Isso significa que o próximo endereço de coluna não pode acontecer até que o dado do endereço de coluna seja transferido para a CPU. Isso limita a taxa na qual as colunas dentro de uma página podem ser endereçadas.

DRAM EDO A DRAM com saída de dados estendida, denominada algumas vezes de *DRAM de modo de página hiper*, é similar à DRAM FPM. A principal diferença é que o sinal $\overline{CAS}$ na **DRAM EDO** não desabilita o dado de saída quando ela vai para o estado desativado porque o dado válido do endereço atual pode ser mantido até que $\overline{CAS}$ seja ativado novamente. Isso significa que o próximo endereço de coluna pode ser acessado antes de o sistema externo aceitar o dado válido atual. A idéia é acelerar o tempo de acesso.

DRAM BEDO A DRAM com Saída de Dados Estendida em Rajada é uma DRAM EDO com capacidade de endereçamento em rajada. Lembre-se da discussão sobre a SRAM em rajada síncrona em que a característica de endereço em rajada permite que até quatro endereços sejam gerados internamente a partir de um único endereço externo, o que economiza algum tempo de acesso. Esse mesmo conceito se aplica à **DRAM BEDO**.

SDRAM DRAMs rápidas são necessárias para acompanhar a velocidade sempre crescente dos microprocessadores. A DRAM síncrona é uma forma de acompanhar essa evolução. Assim como a RAM estática síncrona discutida anteriormente, a operação da **SDRAM** é sincronizada com clock do sistema, o qual também a operação do microprocessador num sistema de computador. A mesma idéia básica descrita em relação à SRAM em rajada síncrona, também se aplica à SDRAM.

Essa operação sincronizada torna a SDRAM totalmente diferente dos outros tipos de DRAMs assíncronas. Com memórias assíncronas, o microprocessador tem que esperar para que a DRAM complete as suas operações internas. Entretanto, com operações síncronas, a DRAM armazena endereços, dados e informações de controle a partir do processador sob o controle do clock do sistema. Isso permite ao processador manusear outras tarefas enquanto as operações de leitura ou escrita na memória estão acontecendo, em vez de ter que esperar pela memória fazer essas coisas conforme no caso dos sistemas assíncronos.

SEÇÃO 10–2 REVISÃO

1. Cite os dois tipos de SRAM.
2. O que é uma memória cache?
3. Explique como as SRAMs e as DRAMs se diferem entre si.
4. Descreva a operação de refresh na DRAM.
5. Cite os quatro tipos de DRAM.

10-3 MEMÓRIAS APENAS DE LEITURA (ROMs)

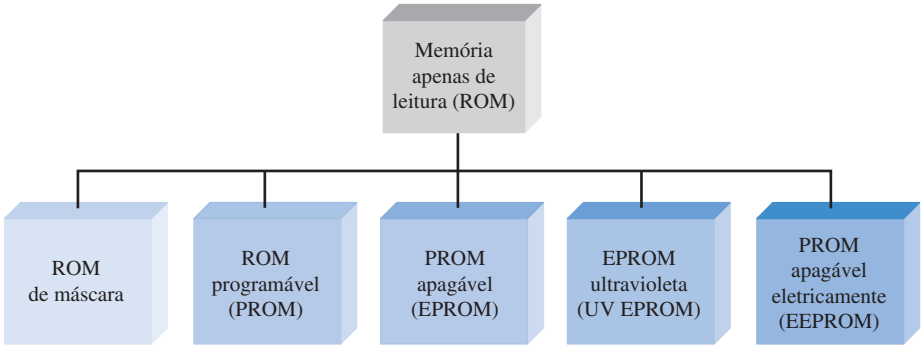
Uma ROM contém permanentemente ou semi-permanentemente dados armazenados, que podem ser lidos da memória, mas não podem ser alterados ou não podem ser alterados sem um equipamento especializado. Uma ROM armazena dados que são usados repetidamente em aplicações tais como tabelas, conversões ou instruções programadas para a inicialização e operação de um sistema. As ROMs retêm os dados armazenados quando a alimentação é desligada, sendo portanto, memórias não-voláteis.

Ao final do estudo desta seção você deverá ser capaz de:

■ Citar os tipos de ROMs ■ Descrever uma célula básica de armazenamento de uma ROM de máscara ■ Explicar como os dados são lidos a partir de uma ROM ■ Discutir a organização interna de uma ROM típica ■ Discutir as aplicações de algumas ROMs

Família de ROMs

A Figura 10–22 mostra como ROMs semicondutoras são categorizadas. A ROM de máscara é o tipo no qual os dados são permanentemente armazenados na memória durante o processo de fabricação. A memória **PROM**, ou ROM programável, é o tipo no qual os dados são armazenados eletricamente através do uso de um equipamento especializado. As memórias ROM de máscara e PROM podem ser de tecnologia MOS ou bipolar. A **EPROM**, ou PROM apagável, é estritamente um dispositivo MOS. A **UV EPROM** é programável eletricamente pelo usuário, porém os dados armazenados têm que ser apagados através da exposição da memória à luz ultravioleta por um período de vários minutos. A PROM apagável eletricamente (**EEPROM** ou E^2PROM) pode ser apagada em alguns milissegundos.



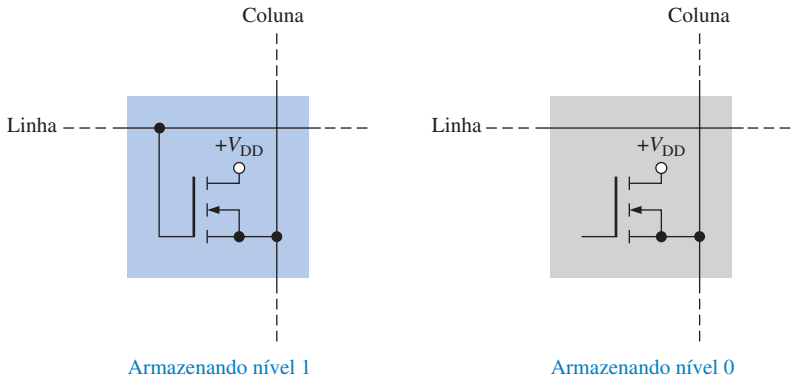
◀ **FIGURA 10–22**
Família de ROMs.

ROM de máscara

A ROM de máscara é geralmente chamada simplesmente de ROM. Ela é programada permanentemente durante o processo de fabricação para prover funções padronizadas usadas amplamente, como conversões populares ou funções de uso especificado. Uma vez que a memória é programada, ela

não pode ser alterada. A maioria dos CIs do tipo ROM utiliza a presença ou ausência de uma conexão a transistor numa junção linha/coluna para representar um nível 1 ou 0.

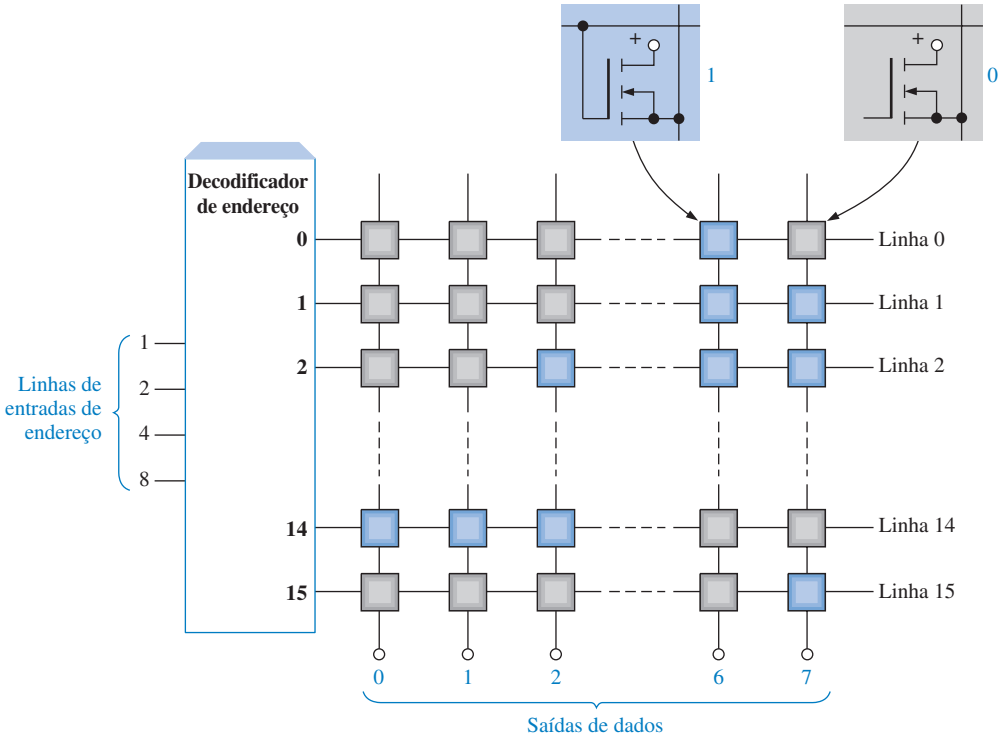
A Figura 10–23 mostra células ROM MOS. A presença de uma conexão de uma linha para a porta de um transistor representa um nível 1 naquela posição porque quando a linha é colocada em nível ALTO, todos os transistores com uma conexão de porta para a linha ligam e conectam o nível ALTO (1) à coluna associada. Na junção linha/coluna onde não existe conexão de porta, a coluna permanece em nível BAIXO (0) quando a linha é endereçada.



▶ **FIGURA 10–23**
Células de ROM.

Uma ROM Simples

Para ilustrar o conceito de ROM, a Figura 10–24 mostra um pequeno arranjo simplificado de uma ROM. Os quadrados na cor laranja representam nível 1 armazenado e os quadrados cinza representam nível 0 armazenado. A operação básica de leitura é a seguinte: quando um código de en-



▲ **FIGURA 10–24**
Uma representação de um arranjo de uma ROM de 16 × 8 bits.

dereço binário é aplicado às linhas de entrada de endereço, a linha correspondente vai para nível ALTO. Esse nível ALTO é conectado à coluna através do transistor em cada junção (célula) onde um nível 1 está armazenado. Em cada célula onde um nível 0 está armazenado, a coluna permanece em nível BAIXO por causa do resistor de terminação. A coluna forma a saída de dados. Os oito bits de dados armazenados na linha selecionada aparecem na saída.

Como podemos ver, o exemplo da ROM dado na Figura 10–24 é organizado em 16 endereços, cada um dos quais armazena 8 bits de dados. Portanto, essa é uma ROM de 16×8 (16×8) e sua capacidade total é 128 bits ou 16 bytes. As ROMs podem ser usadas como tabelas de busca para conversões de códigos e geração de funções lógicas.

EXEMPLO 10–1

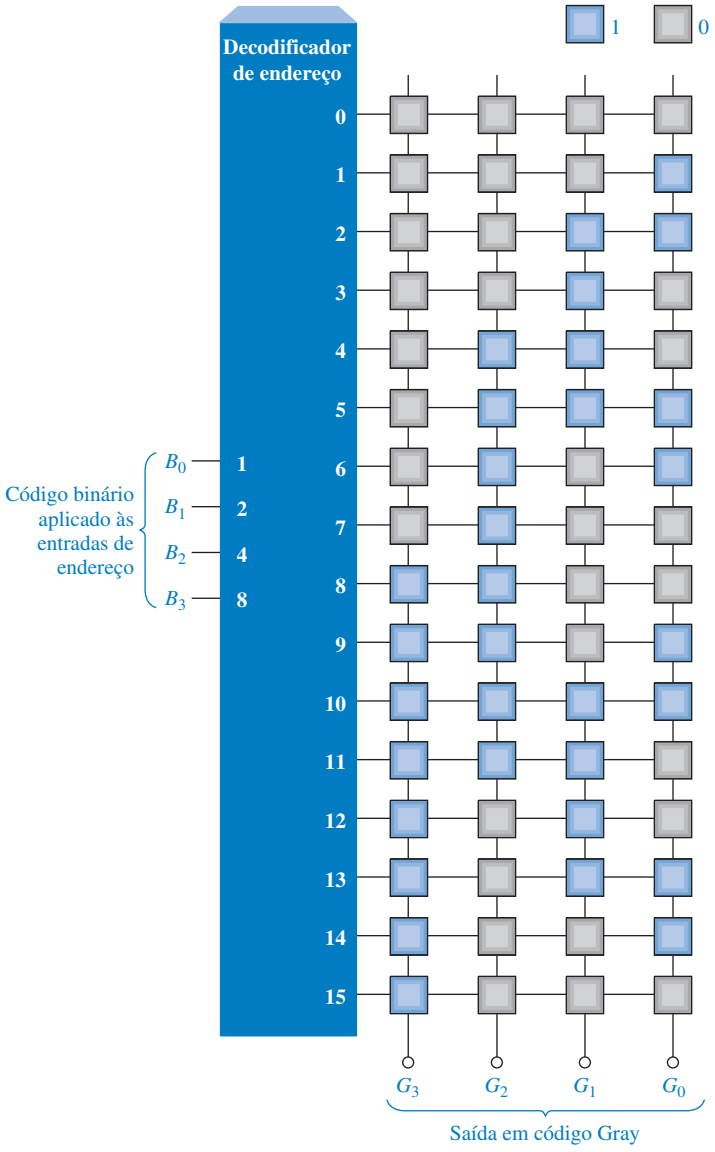
Mostre uma ROM básica, semelhante à da Figura 10–24, programada para conversão de binário para Gray.

Solução Reveja o Capítulo 2 para relembrar o código Gray. A Tabela 10–1 foi desenvolvida para uso na programação da ROM.

TABELA 10–1

BINÁRIO				GRAY			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

O arranjo resultante da ROM 16×4 é mostrado na Figura 10–25. Podemos ver que um código binário na entrada de endereço produz o código Gray correspondente na saída (colunas). Por exemplo, quando o número binário 0110 é aplicado na entrada de endereço, o endereço 6, que armazena o código Gray 0101, é selecionado.



▶ **FIGURA 10–25**

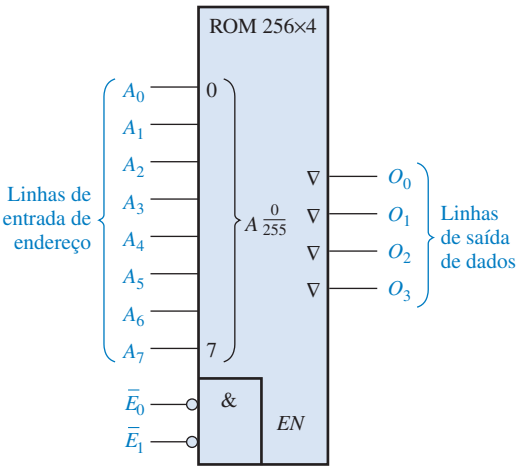
Representação de uma ROM programada como um conversor de código de binário para Gray.

Problema relacionado* Usando a Figura 10–25, determine a saída do código Gray quando o código binário 1011 for aplicado na entrada de endereço.

* As respostas estão no final do capítulo.

Organização Interna de uma ROM

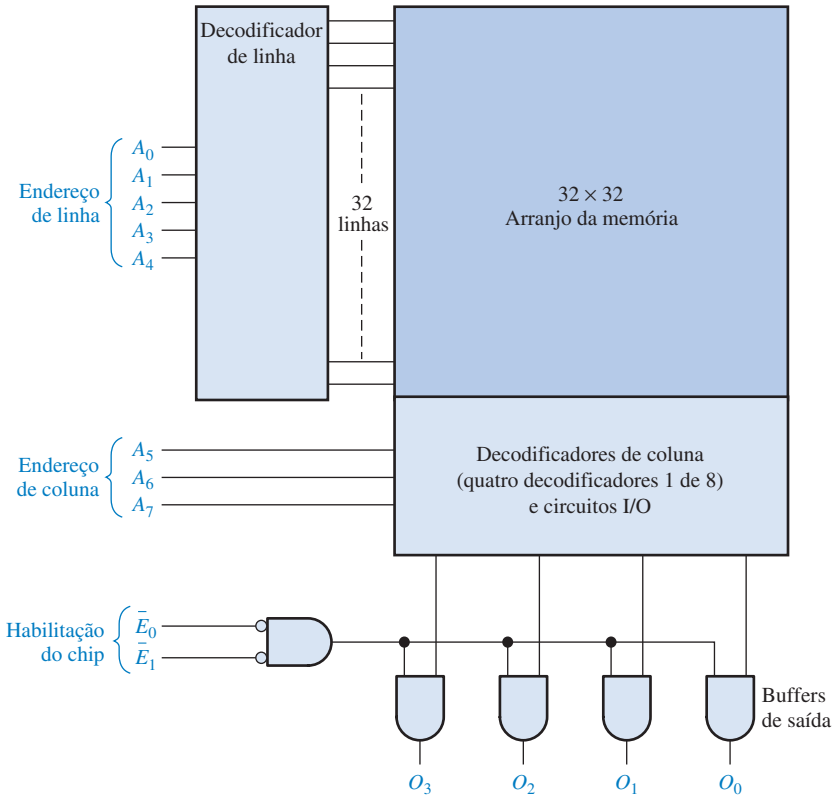
A maioria dos CIs de ROM tem uma organização interna mais complexa que o exemplo básico apresentado anteriormente. Para ilustrar como um CI de ROM é estruturado, vamos usar um dispositivo de 1024 bits com uma organização de 256 × 4. O símbolo lógico é mostrado na Figura 10–26. Quando qualquer um dos 256 códigos binários (oito bits) é aplicado às entradas de endereço, quatro bits de dados aparecem na saída se a entrada de habilitação do chip estiver em nível BAIXO. (256 endereços necessitam de oito bits de endereço.)



◄ FIGURA 10–26

Símbolo lógico de uma ROM de 256 × 4. O indicador $A\frac{0}{255}$ significa que o código de endereço de 8 bits seleciona os endereços de 0 a 255.

Embora a organização 256 × 4 desse dispositivo implique que existem 256 linhas e 4 colunas no arranjo de memória, esse não é o caso real. O arranjo de células de memória é na realidade uma matriz de 32 × 32 (32 linhas e 32 colunas), conforme mostra o diagrama em bloco dado na Figura 10–27.



◄ FIGURA 10–27

Uma ROM de 1024 bits com organização de 256 × 4 baseada num arranjo 32 × 32.

A ROM mostrada na Figura 10–27 funciona da seguinte forma: cinco das oito linhas de endereço (A_0 a A_4) são decodificadas por um decodificador de linha (freqüentemente denominado decodificador *Y*) para selecionar uma das 32 linhas. Três das oito linhas de endereço (A_5 a A_7) são decodificadas por um decodificador de coluna (freqüentemente chamado de decodificador *X*) para selecionar 4 das 32 colunas. Na realidade, o decodificador de coluna consiste em quatro decodificadores (seletores de dados) 1 de 8, como mostra a Figura 10–27.

O resultado dessa estrutura é que quando um código de endereço de 8 bits (A_0 a A_7) for aplicado, uma palavra de dados de 4 bits aparece nas saídas de dados quando as linhas de habilitação

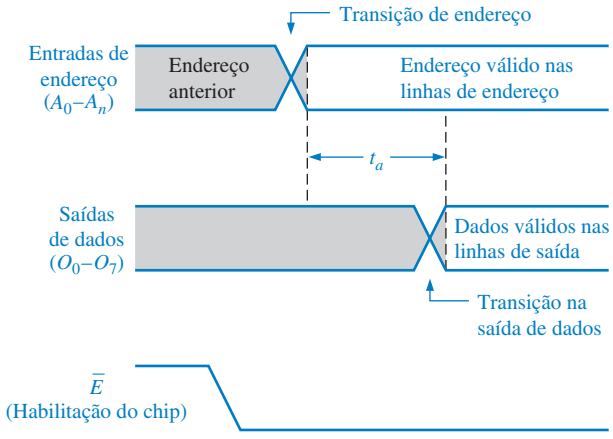
NOTA: COMPUTAÇÃO

Uma ROM é usada num computador pessoal para armazenar o BIOS (*Basic Input/Output System*). Esses programas são usados para realizar a supervisão fundamental e as funções de suporte para o computador. Por exemplo, os programas BIOS armazenados na ROM controlam certas funções do monitor de vídeo e de impressão, provêem a formatação do disco, escaneiam o teclado verificando acionamento de teclas.

do chip $\overline{E}_0$ e $\overline{E}_1$ estão em nível BAIXO para habilitar os buffers de saída. Esse tipo de organização interna (arquitetura) é típica de CIs de ROMs de várias capacidades.

Tempo de Acesso de uma ROM

Um diagrama de temporização típico que ilustra o tempo de acesso de uma ROM é mostrado na Figura 10–28. O **tempo de acesso** (t_a) de uma ROM é o tempo a partir da aplicação de um endereço válido nas linhas de entrada até o surgimento de um dado válido na saída. O tempo de acesso também pode ser medido a partir da ativação da entrada de habilitação do chip ($\overline{E}$) até a ocorrência de um dado de saída válido quando um endereço válido já estiver presente nas linhas de entrada.



► **FIGURA 10–28**

Tempo de acesso (t_a) de uma ROM a partir da mudança de endereço para a saída de dados com a habilitação do chip já ativa.

SEÇÃO 10–3 REVISÃO

1. Qual é a capacidade de armazenamento de bit de uma ROM com uma organização de 512 × 8?
2. Faça uma lista dos tipos de memórias apenas de leitura.
3. Quantos bits de endereço são necessários para uma memória de 2048 bits organizados como uma memória de 256 × 8?

10-4 ROMs PROGRAMÁVEIS (PROMs E EPROMs)

As PROMs são basicamente as mesmas memórias que as ROMs de máscara, uma vez que elas foram programadas. Conforme aprendemos, as ROMs são um tipo de dispositivo lógico programável. A diferença é que as PROMs vêm do fabricante sem programação e são programadas pelo usuário de acordo com a sua necessidade.

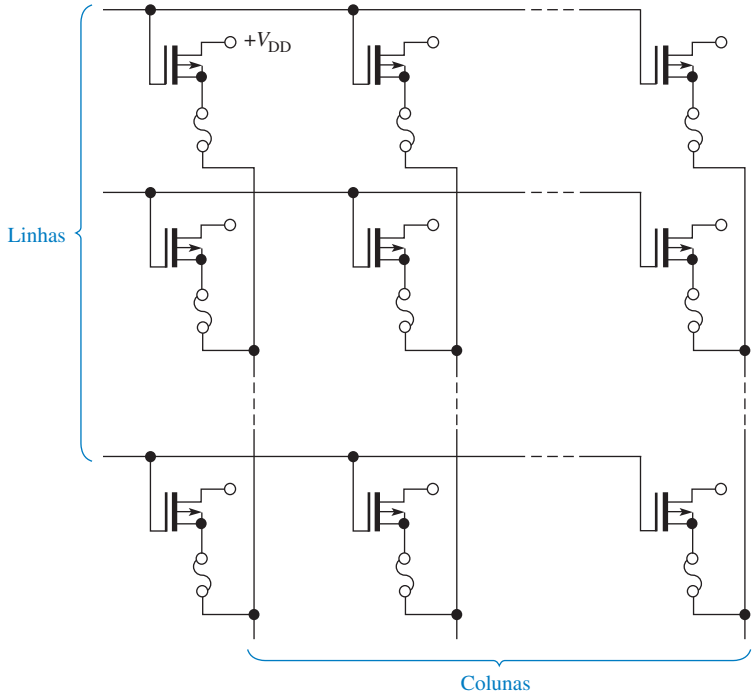
Ao final do estudo desta seção você deverá ser capaz de:

■ Distinguir entre uma ROM de máscara e uma PROM ■ Descrever uma célula de memória PROM básica ■ Discutir EPROMs incluindo UV EPROMs e EEPROMs ■ Analisar o ciclo de programação de uma EPROM

PROMs

Uma **PROM** usa um tipo de processo que faz uso de fusíveis para armazenar bits, no qual uma *conexão a fusível* é "queimada" ou deixada intacta para representar 0 ou 1. O processo de ruptura do fusível é irreversível; uma vez programada a PROM, ela não pode ser alterada.

A Figura 10–29 ilustra o arranjo de uma PROM MOS com conexões a fusível. As conexões a fusível são fabricadas na PROM entre a fonte do transistor de cada célula e a coluna. No processo de programação, uma corrente suficiente é injetada na conexão a fusível para "queimar" e criar um nível 0 armazenado. A conexão a fusível é deixada intacta para o armazenamento de um nível 1.



◀ **FIGURA 10–29**
Arranjo de uma PROM MOS com conexões a fusível (todos os drenos geralmente são conectados a V_{DD}).

Três tecnologias a fusível básicas usadas em PROMs são conexões metálicas, conexões com silício e junções *pn*.

1. Conexões metálicas são feitas de um material tal como o níquel-cromo. Cada bit no arranjo de memória é representado por uma conexão separada. Durante a programação, a conexão é "queimada" (aberta) ou deixada intacta. Isso é feito basicamente endereçando primeiro uma determinada célula e então forçando uma corrente suficiente através da conexão a fusível fazendo-a abrir.
2. Conexões com silício são formadas por estreitas tiras entalhadas de silício policristalino. A programação desses fusíveis requer a fusão das conexões passando uma corrente suficiente através delas. O valor da corrente provoca uma alta temperatura no local do fusível que oxida o silício e forma um isolante em torno da conexão agora aberta.
3. A tecnologia de junção em curto-circuito, ou migração por avalanche induzida, consiste basicamente de duas junções *pn* organizadas em sentidos opostos. Durante a programação, uma das junções dos diodos sofre uma avalanche sendo que a tensão resultante e o calor fazem com que íons de alumínio migrem e coloquem a junção em curto-circuito. A junção restante é então usada como um diodo polarizado diretamente para representar um bit de dado.

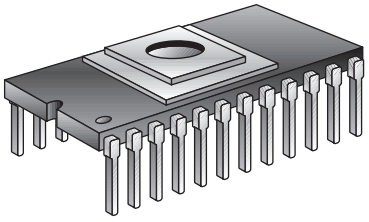
EPROMs

Uma **EPROM** é uma PROM apagável. Diferente de uma PROM comum, uma EPROM pode ser reprogramada se o conteúdo existente no arranjo da memória for apagado primeiro.

Uma EPROM usa um arranjo de MOSFET canal N com uma estrutura de porta isolada. A porta isolada do transistor não tem conexão elétrica, podendo armazenar uma carga elétrica por um período de tempo indefinido. Os bits de dados nesse tipo de arranjo são representados pela presença ou ausência de uma carga armazenada na porta. O apagamento de um bit de dado é um processo que remove a carga da porta.

Dois tipos básicos de PROMs apagáveis são a PROM apagável por ultravioleta (UV EPROM) e a PROM apagável eletricamente (EEPROM).

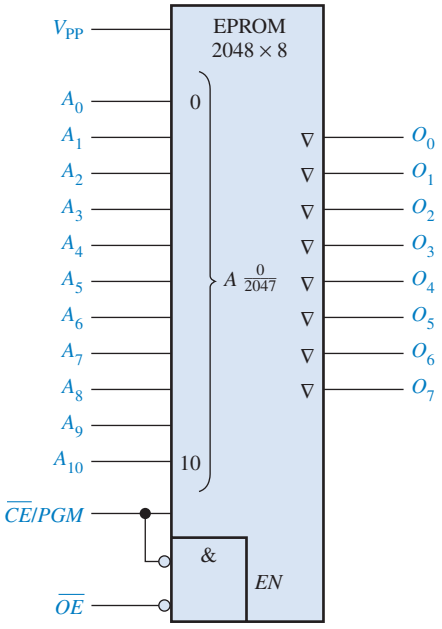
UV EPROMs Podemos reconhecer o dispositivo UV EPROM através de uma "janela" de quartzo transparente no encapsulamento, conforme mostra a Figura 10–30. A porta isolada no **FET** de uma EPROM ultravioleta é "flutuante" dentro de um material de óxido isolante. O processo de programação faz com que elétrons sejam removidos da porta flutuante. O apagamento é feito expondo o chip da memória a uma radiação ultravioleta intensa através da janela de quartzo no topo do encapsulamento. A carga positiva armazenada na porta é neutralizada após vários minutos até uma hora de tempo de exposição.



▶ **FIGURA 10–30**

Encapsulamento de uma PROM apagável por ultravioleta.

Uma UV EPROM é representada na Figura 10–31 por um diagrama lógico. A operação dessa memória é similar a outras UV EPROMs de diversas capacidades. Conforme mostra o diagrama lógico, esse dispositivo tem 2048 endereços ($2^{11} = 2048$), cada um com oito bits. Observe que as oito saídas são tristate (∇).



▶ **FIGURA 10–31**

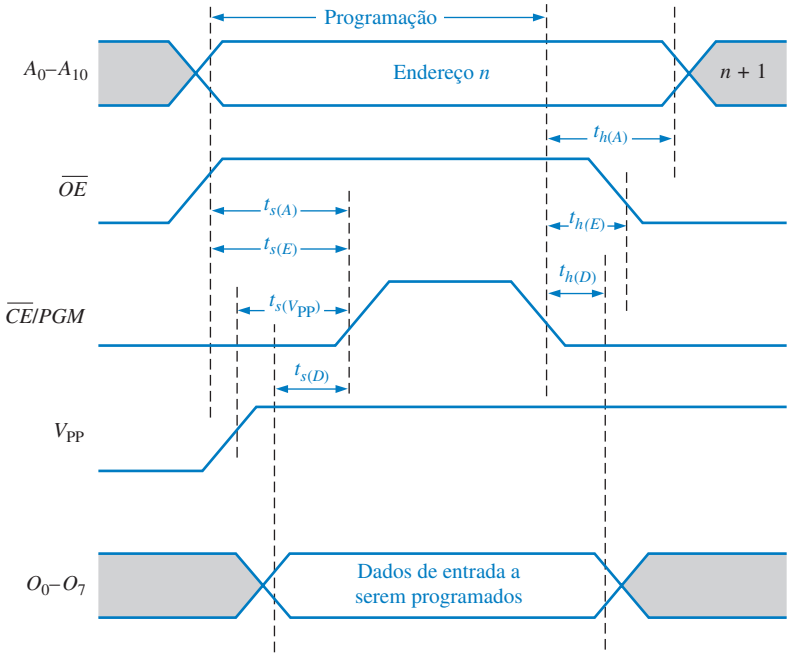
O símbolo lógico para uma UV EPROM de 2048 × 8.

Para ler a memória, a entrada de habilitação da saída ($\overline{OE}$) tem que ser nível BAIXO assim como a entrada de redução de consumo/programação ($\overline{CE}/PGM$). Para apagar os dados armazenados, o dispositivo é exposto a uma luz ultravioleta de alta capacidade através da janela transparente. Uma lâmpada UV típica apaga os dados em cerca de 20 a 25 minutos. Assim como na maioria das UV EPROMs após o apagamento, todos os bits são 1s. A luz ambiente normal contém a luz UV com o comprimento de onda necessário para apagar a memória por um período de tempo. Portanto, a janela transparente no encapsulamento tem que ser mantida protegida contra a luz.

Para programar o dispositivo, uma tensão cc relativamente alta é aplicada em V_{PP} sendo $\overline{OE}$ nível ALTO. Os oito bits de dados a serem programados no endereço determinado são aplicados

nas saídas (O_0 a O_7) e o endereço é aplicado nas entradas A_0 a A_{10}. Em seguida, um pulso de nível ALTO é aplicado na entrada $\overline{CE}/PGM$. Os endereços programados podem ser em qualquer ordem.

A Figura 10–32 mostra um diagrama de temporização para a programação. Esses sinais são geralmente produzidos por um equipamento chamado de programador de EPROM.



◀ FIGURA 10–32

Diagrama de temporização para um ciclo de programação de uma UV EPROM de 2048 × 8 com os tempos críticos de *setup* (t_s) e *hold* (t_h) indicados.

EEPROMs Uma PROM apagável eletricamente pode ser tanto apagada quanto programada com pulsos elétricos. Visto que ela pode ser escrita eletricamente e apagada eletricamente, a EEPROM pode ser rapidamente programada e apagada no próprio circuito (*in-circuit*) para uma reprogramação.

As memórias MOS de porta flutuante e as de silício com nitrito de óxido metálico (MNOS) são os dois tipos de EEPROMs. A aplicação de uma tensão na porta de controle dentro da estrutura da porta flutuante permite o armazenamento ou a remoção de carga da porta flutuante.

SEÇÃO 10–4 REVISÃO

1. Em que as PROMs diferem das ROMs?
2. Após o apagamento, qual é o nível lógico (1 ou 0) de todos os bits de uma EPROM típica?
3. Qual é o modo normal da operação de uma PROM?

10-5 MEMÓRIAS FLASH

Uma memória ideal tem uma alta capacidade de armazenamento, é não-volátil, tem capacidade de leitura e escrita no sistema, é compatível com operações rápidas e tem um custo eficaz. As tecnologias tradicionais de memória como ROM, PROM, EPROM, EEPROM, SRAM e DRAM exibem individualmente um ou mais dessas características, porém nenhuma apresenta todas essas características, exceto a memória flash.

Ao final do estudo desta seção você deverá ser capaz de:

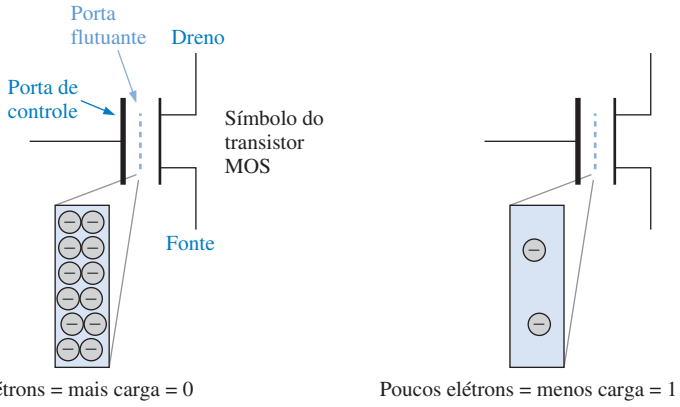
■ Discutir as características básicas de uma memória flash ■ Descrever a operação básica de uma célula de memória flash ■ Comparar memórias flash com outros tipos de memórias

As **memórias flash** são memórias de leitura/escrita de alta densidade (alta densidade se traduz em alta capacidade de armazenamento de bits) não-voláteis, o que significa que os dados podem ser mantidos armazenados indefinidamente sem alimentação. Algumas vezes elas são usadas no lugar de drives de disquetes ou de discos rígidos de pequenas capacidades em computadores portáteis.

Alta densidade significa que um grande número de células pode ser agrupado numa determinada área na superfície de um chip; ou seja, quanto maior a densidade, mais bits podem ser armazenados numa área de mesmo tamanho. Essa alta densidade é conseguida nas memórias flash com o uso de uma célula de armazenamento que consiste de um único transistor MOS de porta flutuante. Um bit de dado é armazenado como uma carga ou a ausência dessa carga na porta flutuante dependendo se for armazenado um nível 0 ou um nível 1.

Célula de Memória Flash

Uma célula de um único transistor numa memória flash é representada na Figura 10–33. O transistor MOS de duas portas em questão consiste de uma porta de controle e uma porta flutuante além do dreno e da fonte. A porta flutuante armazena elétrons (carga) como resultado de uma tensão suficiente aplicada na porta de controle. Um *nível 0 é armazenado quando existe mais cargas e um nível 1 é armazenado quando existe menos ou nenhuma carga*. A quantidade de carga presente na porta flutuante determina se o transistor estará ligado e conduzindo corrente do dreno para a fonte quando uma tensão de controle for aplicada durante uma operação de leitura.



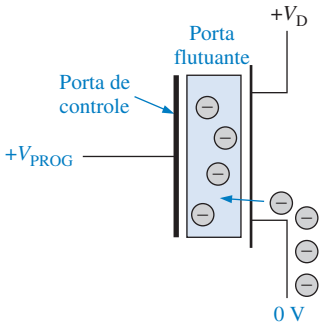
▶ **FIGURA 10–33**
A célula de armazenamento em uma memória flash.

Operação Básica da Memória Flash

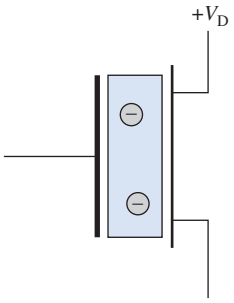
Existem três operações principais numa memória flash: a operação de *programação*, a operação de *leitura* e a operação de *apagamento*.

Programação Inicialmente, todas as células estão no estado 1 porque as cargas foram removidas de cada célula numa operação prévia de apagamento. A operação de programação acrescenta elétrons (carga) na porta flutuante daquelas células que armazenarão um nível 0. Nenhuma carga é acrescentada àquelas células que estão armazenando um nível 1. A aplicação de uma tensão positiva na porta de controle em relação à fonte durante a programação atrai elétrons para a porta flutuante, conforme indicado na Figura 10–34. Uma vez programada, a célula pode reter a carga por até 100 anos sem alimentação externa.

Leitura Durante uma operação de leitura, uma tensão positiva é aplicada na porta de controle. A quantidade de carga presente na porta flutuante de uma célula determina se a tensão aplicada na porta de controle ligará ou não o transistor. Se um nível 1 estiver armazenado, a tensão na porta de controle será suficiente para ligar o transistor. Se um nível 0 estiver armazenado, o transistor não será ligado porque a tensão na porta de controle não será suficiente para se sobrepor à carga negativa armazenada na porta flutuante. Pense na carga na porta flutuante como uma fonte de tensão que se opõe à tensão aplicada na porta de controle durante uma operação de leitura. Assim, a carga na por-



Para armazenar um nível 0, uma tensão positiva suficiente é aplicada na porta de controle em relação à fonte para acrescentar carga à porta flutuante durante a programação.



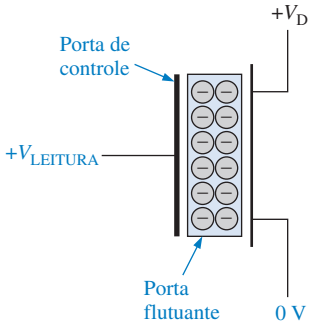
Para armazenar um nível 1, nenhuma carga é acrescentada e a célula é deixada na condição de apagada.

◄ **FIGURA 10–34**

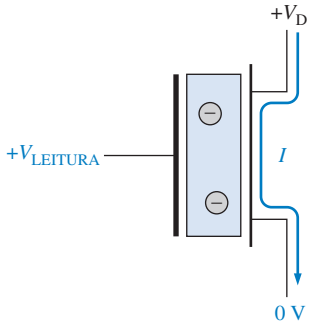
Ilustração simplificada do armazenamento dos níveis 0 e 1 numa célula flash durante a operação de programação.

ta flutuante, associada a um nível 0 armazenado, evita que a tensão na porta de controle consiga atingir a tensão de limiar, ao passo que uma carga nula ou pequena, que é associada a um nível 1 armazenado, permite que a tensão na porta de controle exceda a tensão de limiar que liga o transistor.

Quando o transistor liga, existe uma corrente do dreno para a fonte no transistor da célula. A presença dessa corrente é detectada indicando um nível 1 e a ausência dela é detectada indicando um nível 0. Essa idéia básica está ilustrada na Figura 10–35.



Quando um nível 0 é lido, o transistor permanece desligado porque a carga na porta flutuante evita que a tensão de leitura exceda a tensão de limiar que liga o transistor.

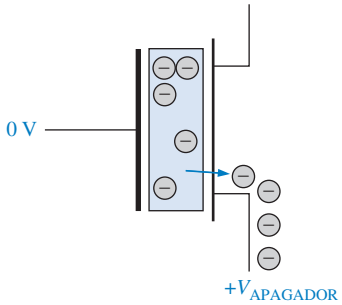


Quando um nível 1 é lido, o transistor liga porque a ausência de carga na porta flutuante permite que a tensão de leitura exceda à tensão de limiar que liga o transistor.

◄ **FIGURA 10–35**

A operação de leitura de uma célula flash num arranjo.

Apagamento Durante uma operação de apagamento, a carga é removida de todas as células de memória. Uma tensão positiva suficiente é aplicada na fonte do transistor em relação à porta de controle. Esta tensão tem polaridade oposta a que é usada na programação. Essa tensão atrai elétrons da porta flutuante e a depleta de carga, conforme ilustra a Figura 10–36. Uma memória flash é sempre apagada antes de ser reprogramada.



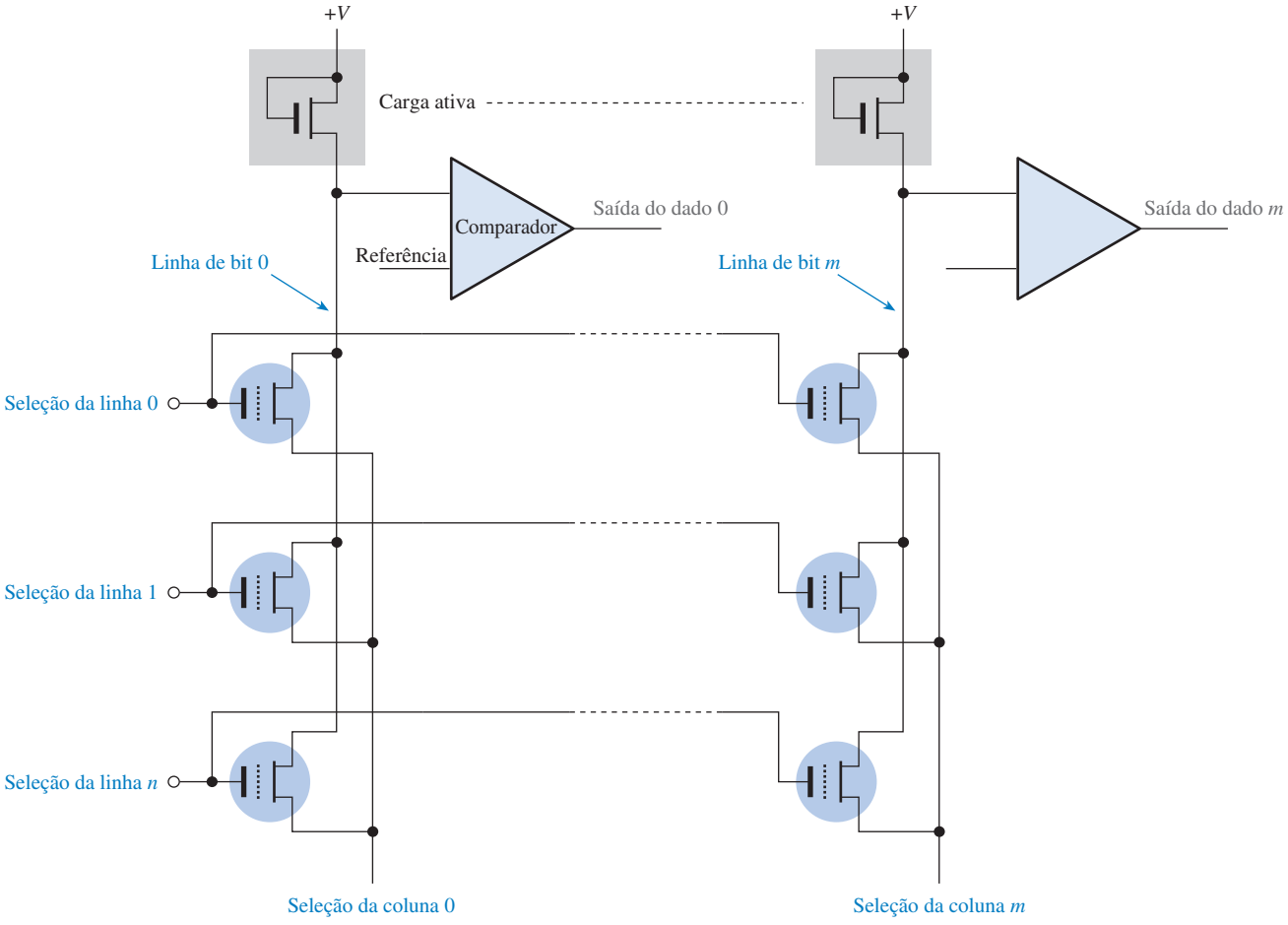
Para apagar uma célula, uma tensão positiva suficiente é aplicada no terminal fonte em relação à porta de controle para remover cargas da porta flutuante durante a operação de apagamento.

◄ **FIGURA 10–36**

Ilustração simplificada da remoção de cargas de uma célula durante o apagamento.

Arranjo Básico de Memória Flash

A Figura 10–37 mostra um arranjo simplificado de células de memória flash. Apenas uma linha é acessada de cada vez. Quando uma célula numa linha de um determinado bit liga (nível 1 armazenado) durante uma operação de leitura, existe uma corrente através da linha de bit, a qual produz uma queda de tensão na carga ativa. Essa queda de tensão é comparada com uma tensão de referência através de um circuito comparador, o qual gera uma saída que indica um nível 1. Se um nível 0 estiver armazenado, existirá uma corrente nula ou pequena na linha de bit produzindo um nível lógico oposto na saída do comparador.



▲ **FIGURA 10–37**

Arranjo básico de memória flash.

O *memory stick* é um dispositivo de armazenamento que usa tecnologia de memória flash numa configuração menor que um tablete de goma de mascar. Os *memory sticks* são comercializados em capacidades de 4 MB, 8 MB, 16 MB, 32 MB, 64 MB e 128 MB e como um kit com um adaptador de cartão para PC. Devido ao seu projeto compacto, ele é ideal para uso em pequenos produtos eletrônicos digitais, como computadores laptop e câmeras digitais.

Comparação entre Memórias Flash e outras Memórias

Vamos comparar as memórias flash com outros tipos de memórias tendo como referência o que aprendemos.

Flash* versus *ROM, EPROM* e *EEPROM As memórias apenas de leitura são dispositivos de alta densidade e não-voláteis. Entretanto, uma vez programada, o conteúdo de uma ROM nunca pode ser alterado. Além disso, a programação inicial é um processo caro e que consome tempo.

Embora a EPROM seja uma memória de alta densidade e não-volátil, ela pode ser apagada apenas removendo-a do sistema e usando uma luz ultravioleta. Ela pode ser reprogramada apenas com equipamentos especializados.

A EEPROM tem uma estrutura de células mais complexa que a ROM ou a EPROM e assim a densidade não é tão alta, embora ela possa ser reprogramada sem ser removida do sistema. Devido à sua baixa densidade, o custo por bit é maior que para as ROMs e EPROMs.

Uma memória flash pode ser reprogramada facilmente no sistema porque ela é essencialmente um dispositivo de LEITURA/ESCRITA. A densidade de uma memória flash se compara com a de uma ROM e de uma EPROM porque ambas tem células de transistor único. Uma memória flash (assim como uma ROM, EPROM ou EEPROM) é não-volátil, o que permite que os dados sejam armazenados indefinidamente mesmo sem alimentação.

Flash* versus *SRAM Conforme aprendemos, as memórias estáticas de acesso aleatório são dispositivos de LEITURA/ESCRITA voláteis. Uma SRAM requer uma alimentação constante para manter os dados armazenados. Em muitas aplicações, uma bateria de backup é usada para evitar a perda de dados se a fonte de alimentação principal for desligada. Entretanto, como é sempre possível ocorrer uma falha no circuito da bateria, a retenção indefinida dos dados armazenados numa SRAM não pode ser garantida. Devido a célula de memória numa SRAM ser basicamente um flip-flop que consiste de diversos transistores, a densidade é relativamente baixa.

Uma memória flash também é uma memória de LEITURA/ESCRITA, porém diferentemente da SRAM ela não é volátil. Além disso, uma memória flash tem uma densidade muito maior que uma SRAM.

Flash* versus *DRAM As memórias dinâmicas de acesso aleatório são dispositivos de LEITURA/ESCRITA voláteis e de alta densidade. As DRAMs necessitam não somente de uma alimentação constante para manter os dados, mas também que os dados armazenados têm que ser freqüentemente reavivados. Em muitas aplicações, um armazenamento de backup, tal como um disco rígido, tem que ser usado em conjunto com uma DRAM.

As memórias flash apresentam densidades maiores que as DRAMs porque uma célula de memória flash consiste de um transistor e não precisa de refresh, ao passo que uma célula de DRAM é constituída de um transistor mais um capacitor e precisa ser reavivada. Tipicamente, uma memória flash consome muito menos potência que uma DRAM equivalente e pode ser usada como substituição a um disco rígido em muitas aplicações.

A Tabela 10–2 fornece um resumo das comparações entre as tecnologias de memórias.

▼ TABELA 10–2

Comparações entre tipos de memórias

TIPO DE MEMÓRIA	NÁO-VOLÁTIL	ALTA DENSIDADE	CÉLULA DE UM TRANSISTOR	ESCRITA NO PRÓPRIO SISTEMA
Flash	Sim	Sim	Sim	Sim
SRAM	Não	Não	Não	Sim
DRAM	Não	Sim	Sim	Sim
ROM	Sim	Sim	Sim	Não
EPROM	Sim	Sim	Sim	Não
EEPROM	Sim	Não	Não	Sim

SEÇÃO 10–5 REVISÃO

1. Quais são os tipos de memórias não-voláteis?
2. Qual é a principal vantagem de uma memória flash sobre uma SRAM ou uma DRAM?
3. Faça uma lista com os três modos de operação de uma memória flash.

10-6 EXPANSÃO DE MEMÓRIA

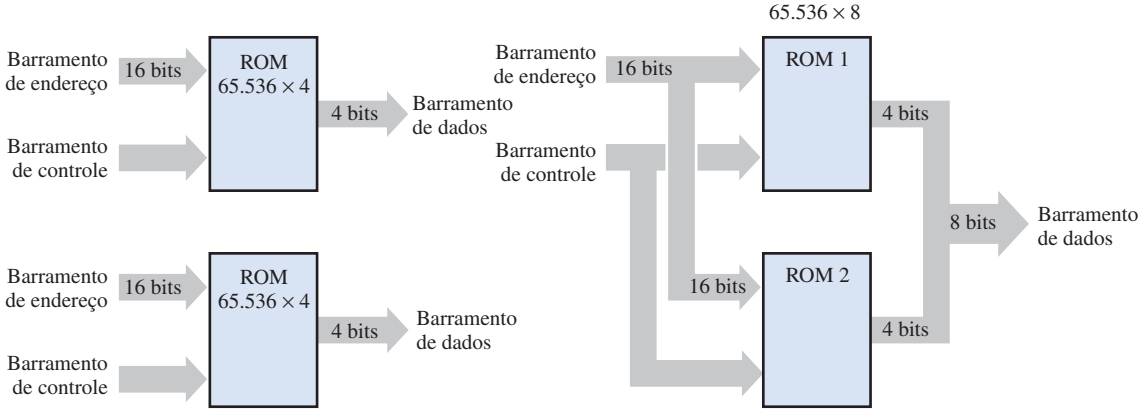
A partir das memórias disponíveis podemos implementar expansões que aumentem o tamanho da palavra (número de bits em cada endereço) ou a capacidade de palavras (número de endereços diferentes) ou ainda ambos. Uma expansão de memória é realizada acrescentando um número apropriado de chips de memória aos barramentos de endereço, dados e controle. Nesse momento são apresentados os módulos de expansão de memória SIMMs, DIMMs e RIMMs.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir *expansão do tamanho da palavra* ■ Mostrar como expandir o tamanho da palavra de uma memória ■ Definir *expansão de capacidade de palavras* ■ Mostrar como expandir a capacidade de palavras de uma memória

Expansão do Tamanho da Palavra

Para aumentar o **tamanho da palavra** de uma memória, o número de bits no barramento de dados tem que ser aumentado. Por exemplo, um tamanho de palavra de 8 bits pode ser conseguido usando duas memórias, cada uma com palavras de 4 bits, conforme ilustrado na Figura 10–38(a). Como podemos ver na parte (b), o barramento de endereço de 16 bits é geralmente conectado nas duas memórias de forma que as memórias combinadas ainda tenham o mesmo número de endereços (2^{16} = 65.536) conforme cada memória individual. O barramento de dados de 4 bits das duas memórias são combinados para formar um barramento de 8 bits. Agora quando um endereço for selecionado, oito bits são colocados no barramento de dados – quatro bits para cada memória. O Exemplo 10–2 mostra os detalhes da expansão de 65.536 × 4 para 65.5536 × 8.



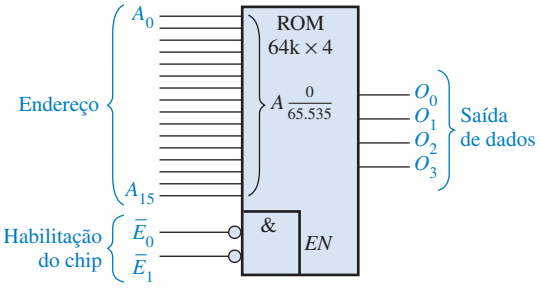
(a) Duas ROMs independentes de 65.536 × 4 (b) Uma ROM de 65.536 8 a partir de duas ROMs de 65.536 × 4

▲ **FIGURA 10–38**

Expansão de duas ROMs de 65.536 × 4 para uma ROM de 65.536 × 8 para ilustrar a expansão do tamanho da palavra.

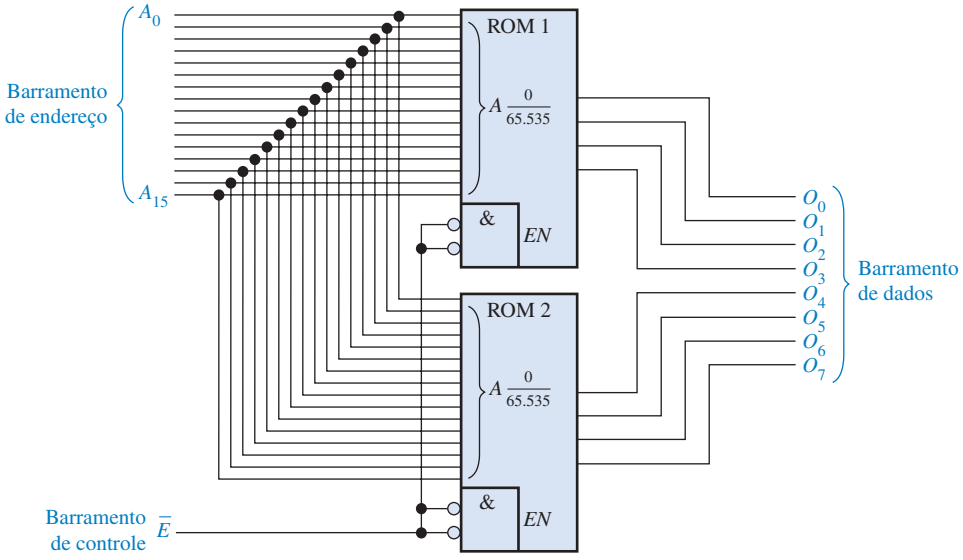
EXEMPLO 10–2

Faça a expansão de memória a partir da ROM de 65.536 × 4 (64k × 4) dada na Figura 10–39 para formar uma ROM de 64k × 8. Observe que "64k" é uma forma reduzida de expressar 65.536. Porque não expressamos como sendo 65k? Talvez seja porque 64 seja também uma potência de 2.



▶ **FIGURA 10–39**
Uma ROM de 64k × 4.

Solução Duas ROMs de 64k × 4 são conectadas como mostra a Figura 10–40. Observe que um endereço específico é acessado na ROM 1 e na ROM 2 ao mesmo tempo. Os quatro bits do endereço selecionado na ROM 1 e os quatro bits do endereço selecionado na ROM 2 saem em paralelo formando uma palavra de 8 bits no barramento de dados. Além disso observe que um nível BAIXO na linha de habilitação do chip $\overline{E}$, que forma um simples barramento de controle, habilita as *duas* memórias.



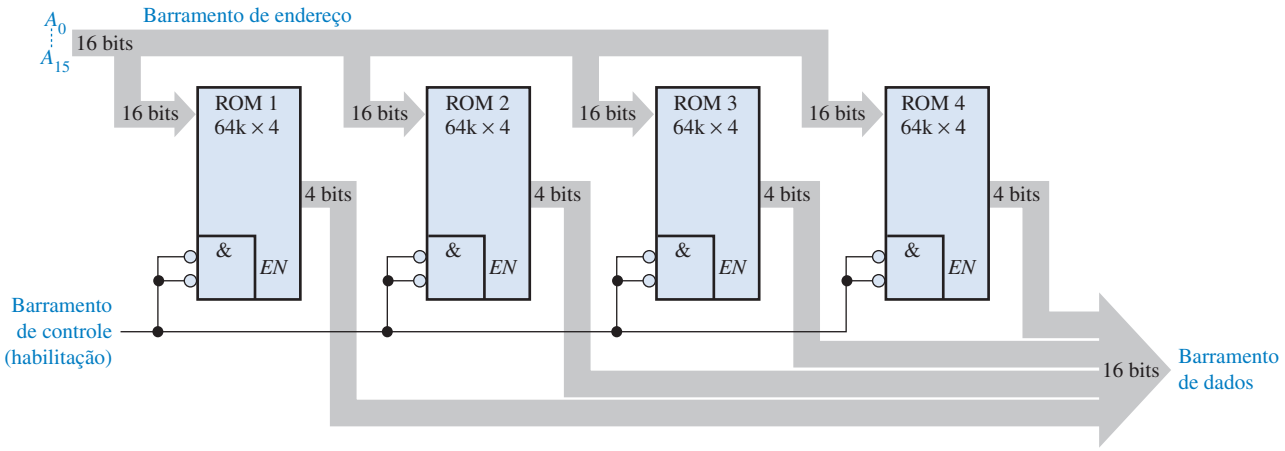
▲ **FIGURA 10–40**

Problema relacionado Descreva como poderia ser feita a expansão de memória de uma ROM de 64k × 1 para uma de 64k × 8.

EXEMPLO 10–3

Use a memória dada no Exemplo 10–2 para formar uma ROM de 64k × 16.

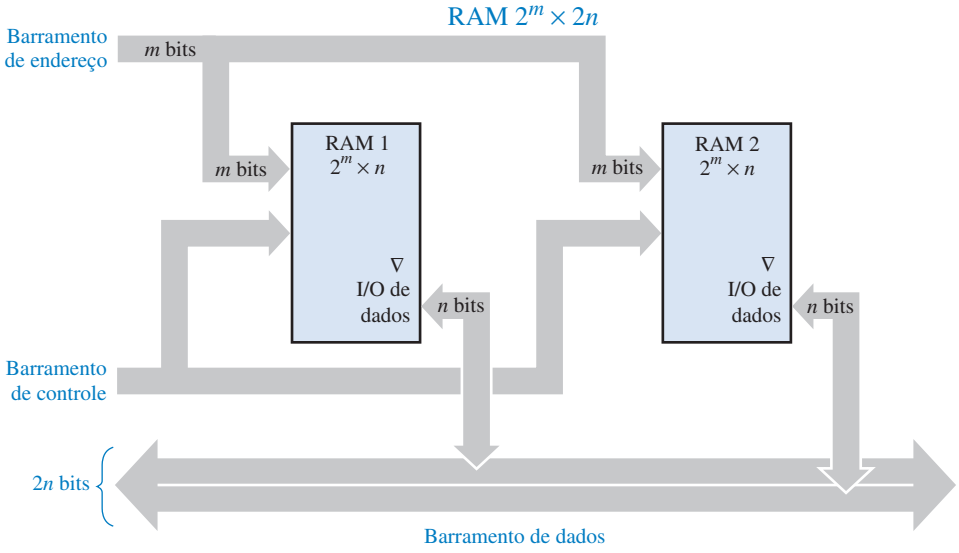
Solução Nesse caso, precisamos de uma memória que armazene 65.536 palavras de 16 bits. Quatro ROMs de 64k × 4 são necessárias para realizar o trabalho, como mostra a Figura 10–41.



▲ **FIGURA 10–41**

Problema relacionado Quantas ROMs de 64k × 1 seriam necessárias para implementar a memória mostrada na Figura 10–41?

Uma ROM tem apenas saídas de dados, porém uma RAM tem entradas e saídas de dados. Para expandir o tamanho da palavra numa RAM (SRAM ou DRAM), as entradas *e* saídas de dados formam o barramento de dados. Como as mesmas linhas são usadas para entrada e saída de dados, são necessários buffers tristate. A maioria das RAM possui circuito tristate interno. A Figura 10–42 ilustra uma expansão de RAM para aumentar o tamanho da palavra.



▶ **FIGURA 10–42**

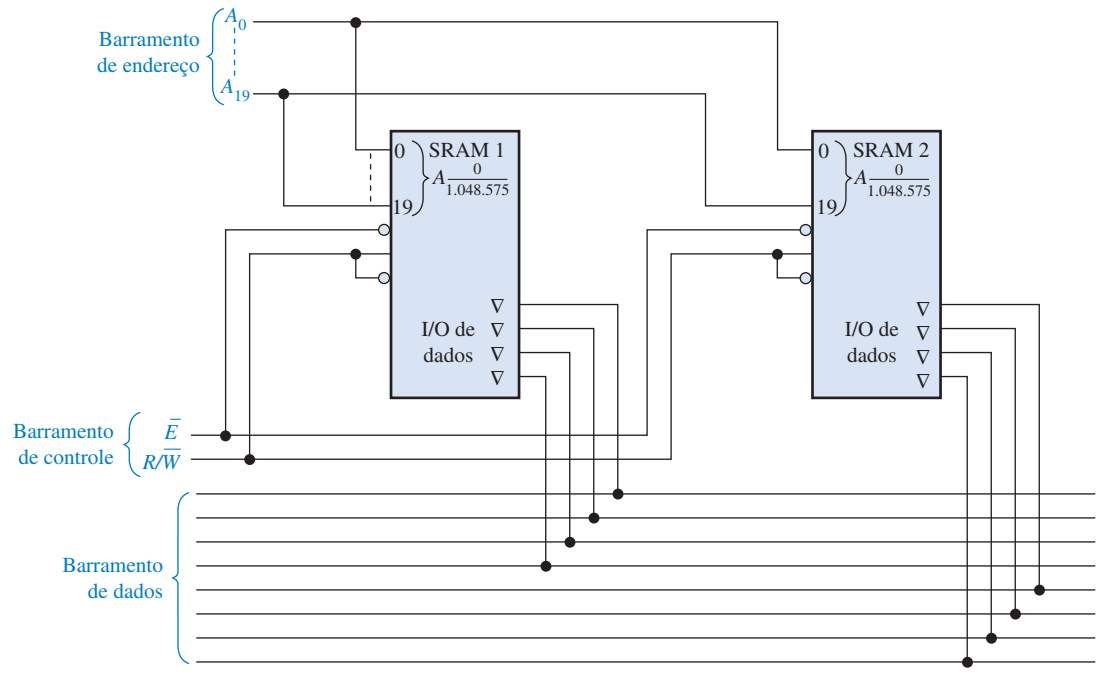
Ilustração da expansão do tamanho da palavra com duas RAMs de $2^m \times n$ formando uma RAM de $2^m \times 2n$.

EXEMPLO 10–4

Use SRAMs de 1M × 4 para criar uma SRAM de 1M × 8.

Solução Duas SRAMs de 1M × 4 são conectadas, como mostra o diagrama em bloco simplificado na Figura 10–43.

Problema relacionado Use SRAMs de 1M × 8 para criar uma SRAM de 1M × 16.

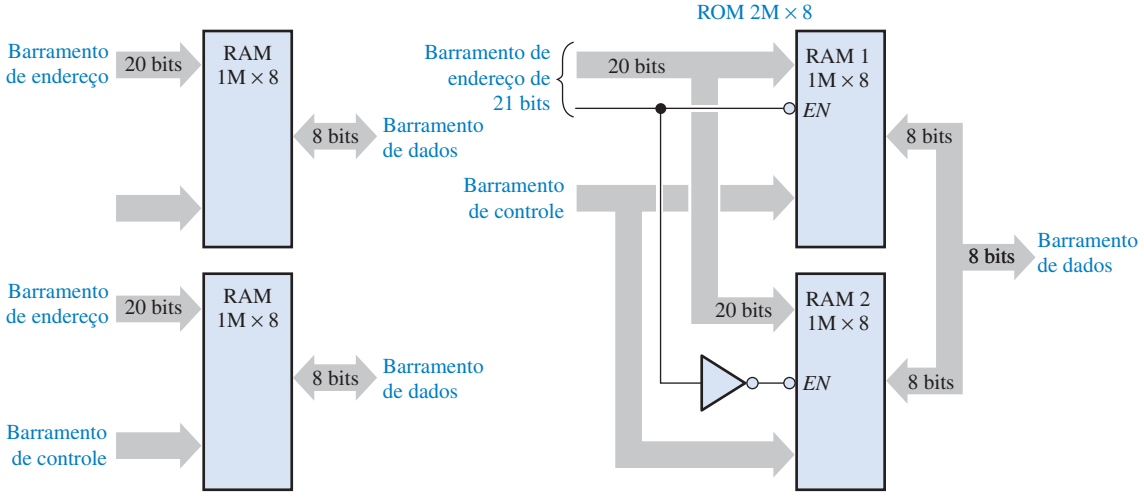


▲ FIGURA 10–43

Expansão da Capacidade de Palavras

Quando as memórias são expandidas para aumentar a **capacidade de palavras**, o *número de endereços é aumentado*. Para conseguir esse aumento, o número de bits de endereços tem que ser aumentado, conforme ilustra a Figura 10–44, (onde duas RAMs de 1M × 8 são expandidas para formar uma memória de 2M × 8).

Cada memória individual tem 20 bits de endereço para selecionar 1.048.576 endereços, como mostra a parte (a) da figura. A memória expandida tem 2.097.152 endereços necessitando portan-



(a) Cada memória individual armazena 1.048.576 palavras de 8 bits

(b) Memórias expandidas para formar RAM de 2M × 8 necessitando de um barramento de endereço de 21 bits

▲ FIGURA 10–44

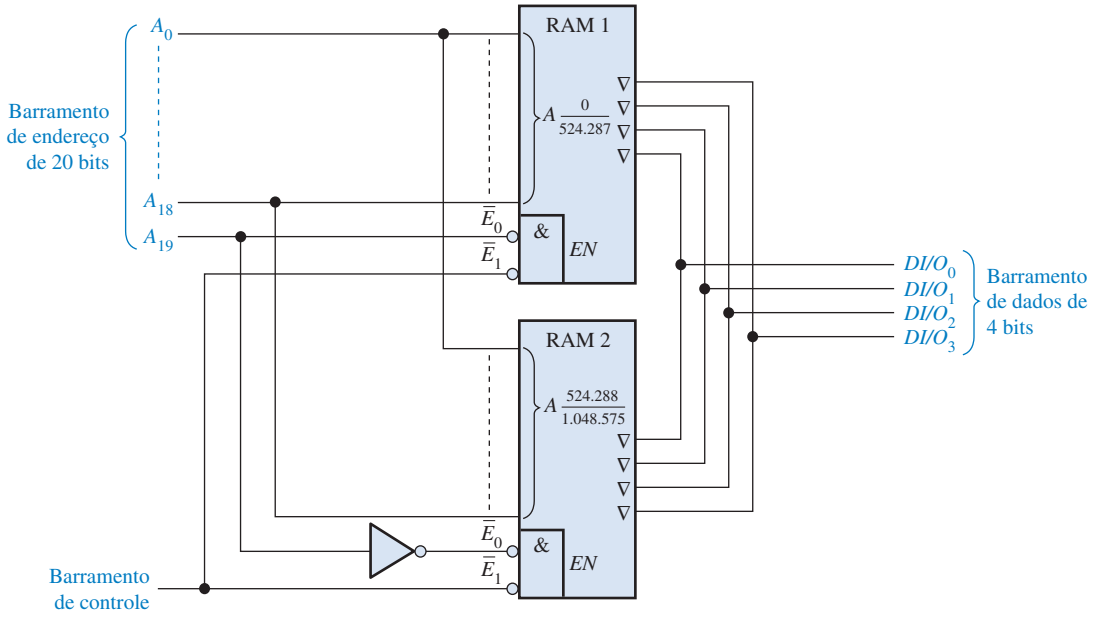
Ilustração da expansão da capacidade de palavras.

to de 21 bits de endereço, como mostra a parte (b) da figura. O vigésimo primeiro bit de endereço é usado para o chip de memória próprio. O barramento de dados para a memória expandida permanece com a largura de 8 bits. Detalhes dessa expansão estão ilustrados no Exemplo 10–5.

EXEMPLO 10–5

Use RAMs de 512k × 4 para implementar uma memória de 1M × 4.

Solução Para expandir o endereçamento é necessário conectar a entrada de habilitação do chip ($\overline{E}_0$) no vigésimo bit de endereço (A_{19}), como mostra a Figura 10–45. A entrada $\overline{E}_1$ é usada como uma entrada de habilitação comum às duas memórias. Quando o vigésimo bit de endereço (A_{19}) for nível BAIXO, a RAM 1 é selecionada (a RAM 2 é desabilitada) e os dezenove bits de endereço menos significativos ($A_0 - A_{18}$) acessam cada um dos endereços na RAM 1. Quando o vigésimo bit de endereço (A19) for nível ALTO, a RAM 2 é habilitada por um nível BAIXO na saída do inversor (a RAM é desabilitada) e os dezenove bits menos significativos ($A_0 - A_{18}$) acessam cada um dos endereços da RAM 2.



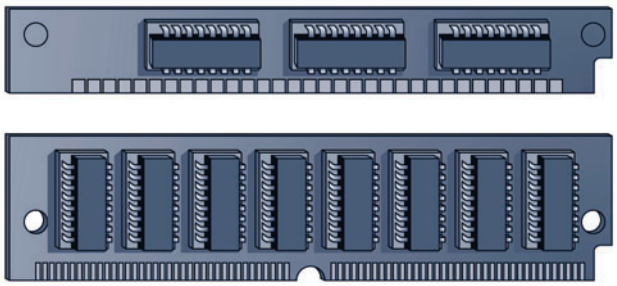
▲ **FIGURA 10–45**

Problema relacionado Quais são as faixas de endereços na RAM 1 e na RAM 2 vistas na Figura 10–45?

Módulos de Memória

As RAMs normalmente fornecidas como módulos de memória em linha única (**SIMMs**) ou como módulos de memória em linha dupla (**DIMMs**). SIMMs e DIMMs são pequenas placas de circuito impresso nas quais os chips de memórias (CIs) são montados com entradas e saídas conectadas no conector de borda na parte inferior da placa. DIMMs são geralmente mais rápidas, porém eles só podem ser instalados em máquinas que foram projetadas para usar esse tipo de módulo.

As duas classificações dos módulos SIMMs são os de 30 pinos e os de 72 pinos. Esses são ilustrados na Figura 10–46. Embora as capacidades de memórias para módulos SIMMs variem de 256 kB a 32 MB, a principal diferença nas duas configurações de pinos é o tamanho da via de dados. Geralmente, os módulos SIMMs de 30 pinos são projetados para barramentos de dados de 8 bits, sendo necessário mais módulos SIMMs para operar com mais bits de dados. Os módulos SIMMs de 72 bits podem acomodar um barramento de dados de 32 bits, assim, é necessário um par de módulos SIMMs para um barramento de dados de 64 bits.

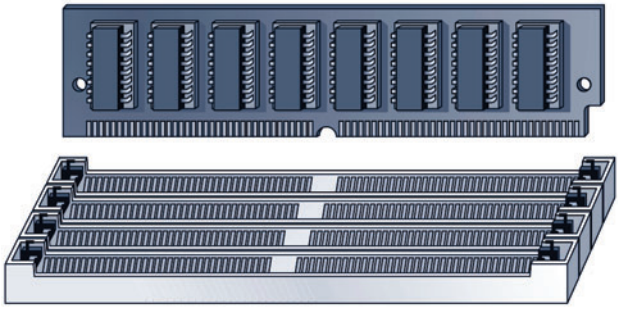


◀ **FIGURA 10–46**
Módulos SIMMs de 30 e 72 pinos.

Os módulos DIMMs são similares aos SIMMs, mas aqueles apresentam um aumento na densidade de memória com apenas um pequeno aumento relativo no tamanho físico. A principal diferença é que os módulos DIMMs distribuem os pinos de entrada e saída nos dois lados da placa PC, ao passo que os módulos SIMMs usam apenas um lado. As configurações DIMM comuns são de 72 pinos, 100 pinos, 144 pinos e 168 pinos que acomodam vias de dados de 32 e 64 bits. Geralmente, a faixa de capacidade de módulos DIMMs é de 4 MB a 512 MB.

Os módulos SIMMs e DIMMs são conectados em soquetes na placa do sistema como ilustrado na Figura 10–47, onde vários soquetes estão geralmente disponíveis para expansão de memória. Os soquetes para os módulos SIMMs e DIMMs são, obviamente, diferentes e não intercambiáveis.

Um outro módulo de memória padrão similar ao DIMM, mas com um barramento de maior velocidade, é o RIMM (módulo de memória rambus em uma linha). Além disso, muitos computadores laptop usam uma variação do módulo DIMM denominada SODIMM, a qual possui dimensões menores, 144 pinos e capacidade de até 256 MB.



◀ **FIGURA 10–47**
Um módulo SIMM/DIMM é inserido no soquete na placa de um sistema.

DICA PRÁTICA

Os componentes de memória são extremamente sensíveis à eletricidade estática. Siga as seguintes precauções quando manusear chips de memória ou módulos tais como os SIMMs e DIMMs:

- Antes do manuseio, descarregue a carga estática do seu corpo tocando numa superfície aterrada ou, se disponível, use uma pulseira própria aterrada através de um resistor de alto valor. Um terra conveniente e confiável é obtido da tomada elétrica (tomada com terra disponível).
- Não retire os componentes da embalagem anti-estática até que esteja pronto para instalá-los.
- Não repouse os componentes sobre a embalagem anti-estática porque apenas a parte interna é anti-estática.
- Quando manusear módulos SIMMs ou DIMMs, segure-os pelas bordas ou pelo suporte de montagem metálico. Não toque os componentes nem a borda dos pinos do conector.
- Nunca deslize qualquer parte sobre qualquer superfície.
- Evite plástico, vinil, espuma e nylon no local de trabalho.

Quando instalar módulos SIMMs e DIMMs, siga os passos a seguir:

1. Alinhe os recortes de encaixe das placas SIMM ou DIMM com os recortes no soquete da memória.
2. Empurre firmemente o módulo até que esteja seguramente encaixado no soquete.
3. Geralmente, as travas nos dois lados do soquete encaixam no local quando o módulo estiver completamente inserido. Essas travas também liberam o módulo, assim ele pode ser removido do soquete.

SEÇÃO 10–6 REVISÃO

1. Quantas RAMs de 16k × 1 são necessárias para conseguir uma memória com uma capacidade de palavras de 16k e um tamanho de palavra de 8 bits?
2. Para expandir a memória de 16k × 8 dada na questão 1 para uma organização de 32k × 8, quantas memórias de 16k × 1 são necessárias?
3. O que significa SIMM?
4. O que significa DIMM?
5. O que significa o termo RIMM?

10-7 TIPOS ESPECIAIS DE MEMÓRIAS

Nesta seção, são abordadas a memória FIFO (*first in-first out* – o primeiro a entrar é o primeiro a sair), a memória LIFO (*last in-first out* – o último a entrar é o primeiro a sair), a memória pilha e a memória com dispositivo acoplado por carga.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a memória FIFO ■ Descrever a memória LIFO ■ Discutir a memória pilha ■ Explicar como usar uma parte de uma RAM como memória pilha ■ Descrever uma memória básica CCD

Memória FIFO

Esse tipo de memória é formado por um arranjo de registradores de deslocamento. O termo **FIFO** se refere à operação básica desse tipo de memória, no qual o primeiro bit de dado escrito na memória é o primeiro a ser lido.

Uma importante diferença entre um registrador convencional e um registrador FIFO é ilustrada na Figura 10–48. Num registrador convencional, um bit de dado se move através do registro apenas quando um novo bit de dado é inserido; num registrador FIFO, um bit de dado passa imediatamente através do registrador para a posição do bit mais à direita que está vazio.

Registrador de deslocamento convencional

Entrada	X	X	X	X	Saída
0	0	*X*	*X*	*X*	→
1	1	0	*X*	*X*	→
1	1	1	0	*X*	→
0	0	1	1	1	→

X = bits de dados desconhecidos.
Em um registrador de deslocamento convencional, os dados permanecem à esquerda até que sejam "forçados" a se deslocar em função de dados adicionais.

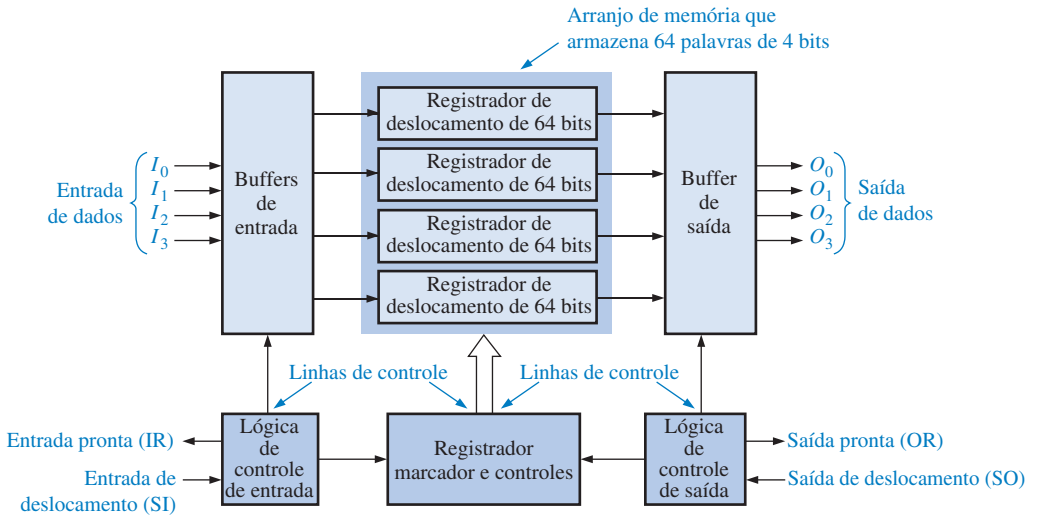
Registrador de deslocamento FIFO

Entrada	—	—	—	—	Saída
0	—	—	—	0	→
1	—	—	1	0	→
1	—	1	1	0	→
0	0	1	1	0	→

— = posições vazias.
Em um registrador de deslocamento FIFO, os dados passam direto.

▲ **FIGURA 10–48**
Comparação da operação de um registrador convencional e um FIFO.

A Figura 10–49 mostra o diagrama em bloco de uma memória serial FIFO. Essa memória em particular tem quatro registradores seriais de dados de 64 bits e um registrador de controle (registrador de marca) de 64 bits. Quando os dados são inseridos por um pulso de deslocamento para dentro (*shift in*), eles se movem automaticamente sob o controle do registrador de marca para a posição vazia mais próxima da saída. Os dados não podem avançar nas posições ocupadas. Entretan-



◀ **FIGURA 10–49**
Diagrama em bloco de uma típica memória serial FIFO.

to, quando um bit de dado é deslocado para fora através de um pulso de deslocamento, os bits restantes nos registradores se movem automaticamente para a próxima posição em direção à saída. Num registrador FIFO assíncrono, os dados são deslocados para fora independente da entrada de dados, com o uso de dois clocks separados.

Aplicações de Registradores FIFO

Uma área de aplicação importante para o registrador FIFO é o caso no qual dois sistemas de taxas de dados diferentes precisam se comunicar. Os dados podem ser inseridos num registrador FIFO a uma taxa e retirados a outra. A Figura 10–50 ilustra como um registrador FIFO pode ser usado nessas situações.

(a) Dados telemétricos irregulares podem ser armazenados e retransmitidos a uma taxa constante

(b) Entrada de dados via teclado a uma taxa lenta que podem ser armazenados e transferidos a uma taxa maior para processamento.

(c) Entrada de dados a uma taxa constante que podem ser armazenados e enviados em rajadas.

(d) Dados de entrada em rajadas podem ser armazenados e reformatados numa saída com taxa constante.

◀ **FIGURA 10–50**
Exemplos de registradores FIFO em aplicações tipo buffer que separam taxas de dados.

Memórias LIFO

A memória **LIFO** é usada em aplicações que envolvem microprocessadores e outros sistemas de computação. Ela permite que dados sejam armazenados e lidos na ordem inversa; ou seja, o último byte de dados a ser armazenado é o primeiro byte de dados a ser recuperado.

Pilhas de Registradores Uma memória LIFO é geralmente referida como uma pilha que "empurra para baixo". Em alguns sistemas, ela é implementada com um grupo de registradores, como mostra a Figura 10–51. Uma pilha pode consistir em um número qualquer de registradores, porém o registrador na parte superior é chamado de *topo da pilha*.

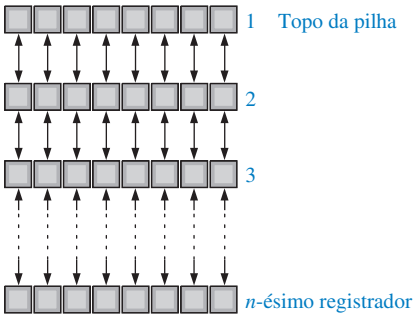


FIGURA 10–51
Pilha de registradores

Para ilustrar o princípio, um byte de dados é carregado em paralelo no topo da pilha. Cada byte sucessivo "empurra" o byte anterior para baixo para o próximo registrador. Esse processo é ilustrado na Figura 10–52. Observe que o novo byte de dado é sempre carregado no registrador do topo e os bytes armazenados anteriormente mais para baixo na pilha. O nome *pilha que empurra para baixo* vem dessa característica.

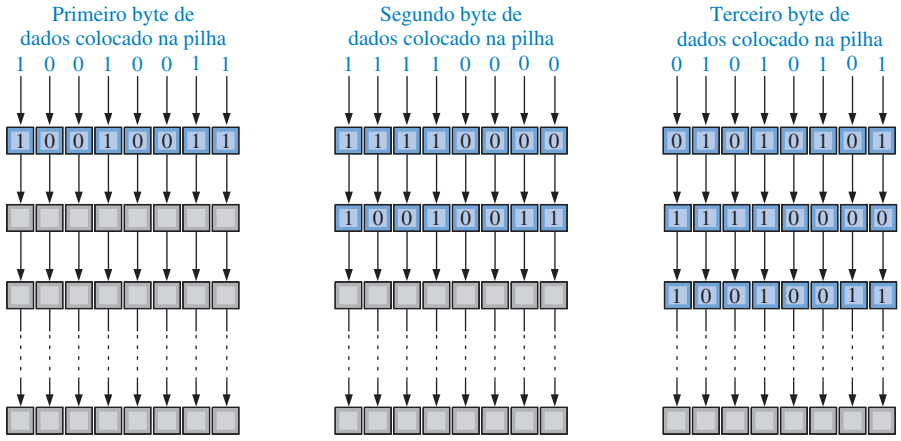


FIGURA 10–52
Ilustração simplificada da inserção de dados na pilha.

Os bytes de dados são recuperados na ordem inversa. O último byte inserido está sempre no topo da pilha, assim, quando ele é retirado, os outros bytes sobem para a próxima posição mais alta. Esse processo é ilustrado na Figura 10–53.

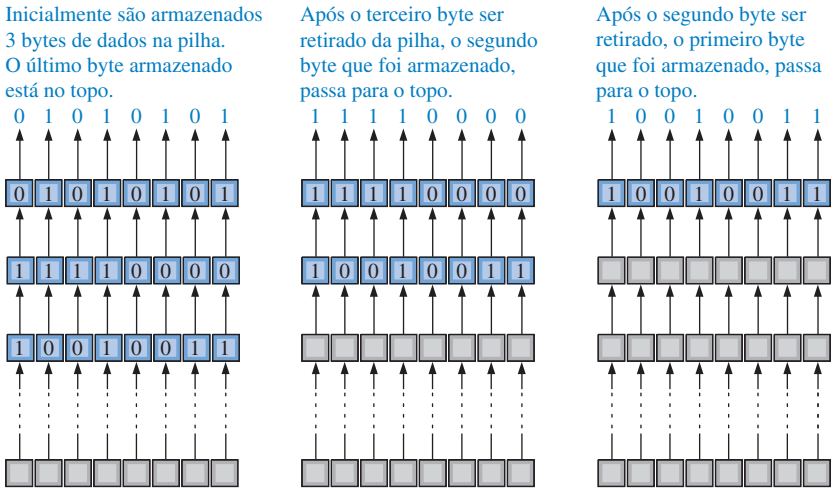
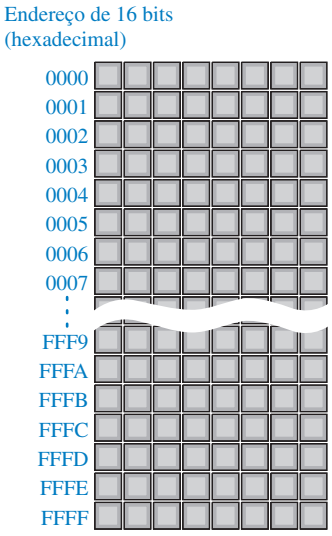


FIGURA 10–53
Ilustração simplificada da retirada de dados de uma pilha.

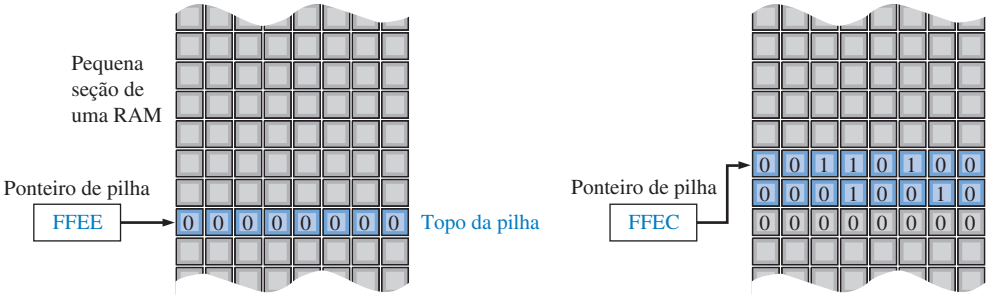
Pilha RAM Uma abordagem da memória LIFO usada em sistemas microprocessados é a alocação de uma seção da RAM como pilha em vez de usar um conjunto dedicado de registradores. Como já vimos, para uma pilha de registradores os dados se movem para cima ou para baixo de uma posição para a próxima. Numa pilha na RAM, os dados não se movem por conta própria mas o topo da pilha se move sob o controle de um registrador denominado de ponteiro de pilha (*stack pointer*).

Considere uma memória de acesso aleatório que é organizada em bytes – ou seja, uma memória na qual cada endereço contém 8 bits – como ilustrado na Figura 10–54. O endereço em binário 0000000000001111, por exemplo, pode ser escrito como 000F em hexadecimal. Um endereço de 16 bits tem um valor hexadecimal *mínimo* de 0000_{16} e um valor *máximo* de $FFFF_{16}$. Com essa notação, um arranjo de memória de 64 kB pode ser representado como mostra a Figura 10–54. O menor endereço de memória é 0000_{16} e o maior endereço de memória é $FFFF_{16}$.



◄ FIGURA 10–54
Representação de uma memória de 64 kB com os endereços de 16 bits expressos em hexadecimal.

Agora, considere uma seção de RAM separada para uso como pilha. Um registrador separado especial, o ponteiro de pilha, contém o endereço do topo da pilha, conforme ilustrado na Figura 10–55. Uma representação hexadecimal de 4 dígitos é usada para o endereço binário. Na figura, os endereços foram escolhidos para fins de ilustração.



◄ FIGURA 10–55
Ilustração da operação de armazenamento (PUSH) numa pilha em memória RAM.

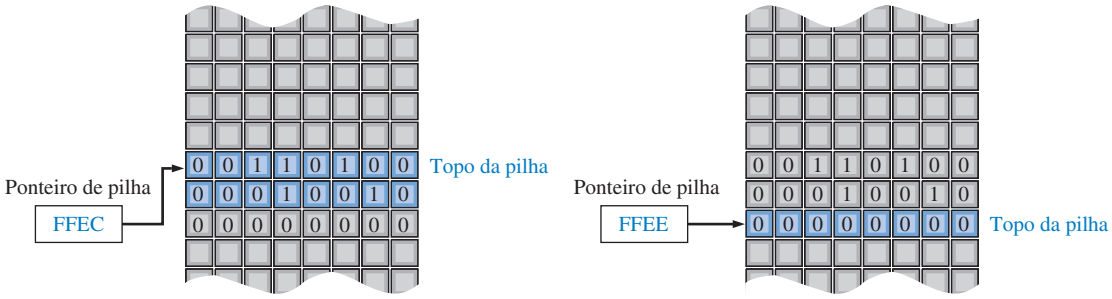
(a) O ponteiro de pilha está inicialmente com FEEE antes que a palavra de dados 0001000010000110100 (1234) seja armazenada.

(b) O ponteiro de pilha é decrementado de dois e a palavra de dados 0001000010000110100 (1234) é armazenada nas duas posições anteriores à posição original indicada pelo ponteiro.

Agora vamos ver como os dados são empurrados para a pilha. O ponteiro de pilha é inicializado com o endereço $FFEE_{16}$, que é o topo da pilha, como mostra a Figura 10–55(a). O ponteiro é então decrementado (diminuído) de dois passando para o valor $FFEC_{16}$. Isso move o topo da pilha para um endereço menor, como mostra a Figura 10–55(b). Observe que o topo da pilha não é estacionário como nos registradores fixos, mas se move para baixo (para um endereço menor) na RAM conforme as palavras de dados são armazenadas. A Figura 10–55(b) mostra que dois bytes

(uma palavra de dados) são então empurrados para a pilha. Após a palavra de dados ser armazenada, o topo da pilha estará em $FFEC_{16}$.

A Figura 10–56 ilustra a operação POP (puxar) para uma pilha na RAM. A última palavra de dados armazenada na pilha é lida primeiro. O ponteiro de pilha que está em $FFEC_{16}$ é incrementado (aumentado) de dois passando para o valor $FFEE_{16}$ e a operação POP é realizada conforme mostra a parte (b) da figura. Tenha em mente que o conteúdo das RAMs não é destruído quando lido, assim, a palavra de dados ainda permanece na memória após a operação POP. Uma palavra de dados é destruída apenas quando uma nova palavra é escrita sobre ela.



(a) O ponteiro está em FFEC antes da palavra de dados ser copiada da pilha.

(b) O ponteiro é incrementado de dois e a última palavra de dados armazenada é copiada a partir da pilha.

▲ **FIGURA 10–56**

Ilustração da operação de leitura (POP) numa pilha em memória RAM.

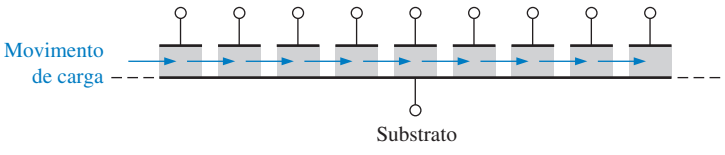
Uma pilha em RAM pode ter uma profundidade qualquer, dependendo do número de endereços contínuos de memória associados para essa finalidade.

Memórias CCD

A memória **CCD** (dispositivo acoplado por carga) armazena dados como cargas em capacitores. Entretanto, diferentemente da DRAM, a célula de armazenamento não inclui um transistor. A principal vantagem das CCDs é a alta densidade.

A memória CCD consiste em uma longa linha de capacitores em semicondutor, denominado *canais*. Os dados são inseridos num canal de forma serial depositando nos capacitores uma pequena carga para um nível 0 e uma grande carga para um nível 1. Esses pacotes de carga são então deslocados ao longo do canal através de sinais de clock conforme mais dados são inseridos.

Assim como com a DRAM, as cargas têm que ser reavivadas periodicamente. Esse processo é feito deslocando os pacotes de carga de forma serial através de um circuito de refresh. A Figura 10–57 mostra o conceito básico de um canal CCD. Como os dados são deslocados de forma serial através dos canais, a memória CCD tem um tempo de acesso relativamente longo. Arranjos de CCD são usados em algumas câmeras modernas para captura de imagens de vídeo na forma de carga induzida por luz.



▶ **FIGURA 10–57**

Um canal CCD (dispositivo acoplado por carga).

SEÇÃO 10–7 REVISÃO

1. O que é uma memória FIFO?
2. O que é uma memória LIFO?
4. Explique a operação PUSH na pilha em memória.
3. Explique a operação POP na pilha em memória.
5. O que significa o termo *CCD*?

10-8 ARMAZENAMENTO MAGNÉTICO E ÓPTICO

Nesta seção, são apresentadas as bases dos discos magnéticos, da fita magnética, dos discos magneto-ópticos e dos discos ópticos. Esses meios de armazenamento são muito importantes, particularmente em aplicações de computadores, onde são usados para armazenamento não-volátil em massa de dados e programas.

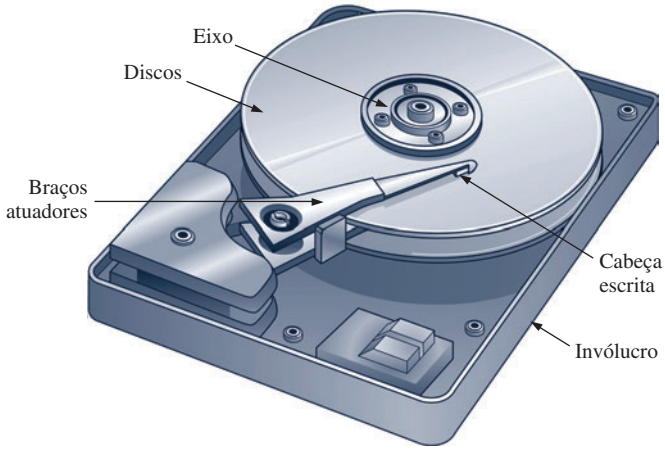
Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever um disco rígido magnético ■ Descrever um disquete ■ Discutir discos rígidos removíveis ■ Explicar o princípio dos discos magneto-ópticos ■ Discutir os discos CD-ROM, CD-R e CD-RW ■ Descrever o WORM ■ Discutir o DVD-ROM

Armazenamento Magnético

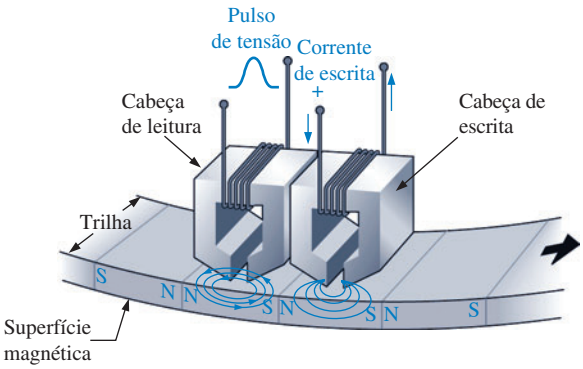
Discos Rígidos Magnéticos Os computadores usam discos rígidos como um meio de armazenamento em massa interno. Os **discos rígidos** são discos ("pratos") rígidos feitos de uma liga de alumínio ou uma mistura de vidro e cerâmica cobertos com uma camada magnética. Os drivers de discos rígidos se dividem principalmente em dois tamanhos, 5,25 polegadas e 3,5 polegadas de diâmetro, embora os diâmetros de 2,5 e 1,75 polegadas também sejam comercializados. Um drive de disco rígido é hermeticamente selado para manter o disco livre de poeira.

Tipicamente, dois ou mais discos são empilhados uns sobre os outros com um eixo comum em torno do qual gira o conjunto em vários milhares de rpm. Uma separação entre cada disco permite o acesso de uma cabeça de leitura/escrita magnética que é montada na extremidade de um braço atuador, conforme mostra a Figura 10–58. Existe uma cabeça de leitura/escrita para os dois lados de cada disco visto que os dados são gravados nos dois lados da superfície do disco. O braço atuador do drive sincroniza todas as cabeças de leitura/escrita mantendo-as em perfeito alinhamento conforme elas "voam" sobre a superfície do disco com uma separação de apenas uma fração de um milímetro do disco. Uma pequena partícula de sujeira poderia fazer com que uma cabeça danifique ("crach") a superfície do disco.



◀ **FIGURA 10–58**
Um drive de disco rígido.

Princípios Básicos da Cabeça de Leitura/Escrita O drive de disco rígido é um dispositivo de acesso aleatório porque ele pode recuperar dados armazenados em qualquer local do disco e em qualquer ordem. Um diagrama simplificado da operação de leitura/escrita na superfície magnética é mostrado na Figura 10–59. A direção ou polarização dos domínios magnéticos na superfície do disco é controlada pela direção das linhas de fluxo magnético (campo magnético) produzido pela cabeça de escrita de acordo com a direção do pulso de corrente no enrolamento. Esse fluxo magnético magnetiza um pequeno ponto na superfície do disco na direção do campo magnético. Um ponto magnetizado com uma polaridade representa um binário 1 e um ponto magnetizado com polaridade oposta representa um binário 0. Uma vez que um ponto na superfície do disco foi magnetizado, ele permanece até que ocorra uma escrita sobre esse ponto com um campo magnético oposto.



▶ **FIGURA 10–59**
Operação simplificada da cabeça de leitura/escrita.

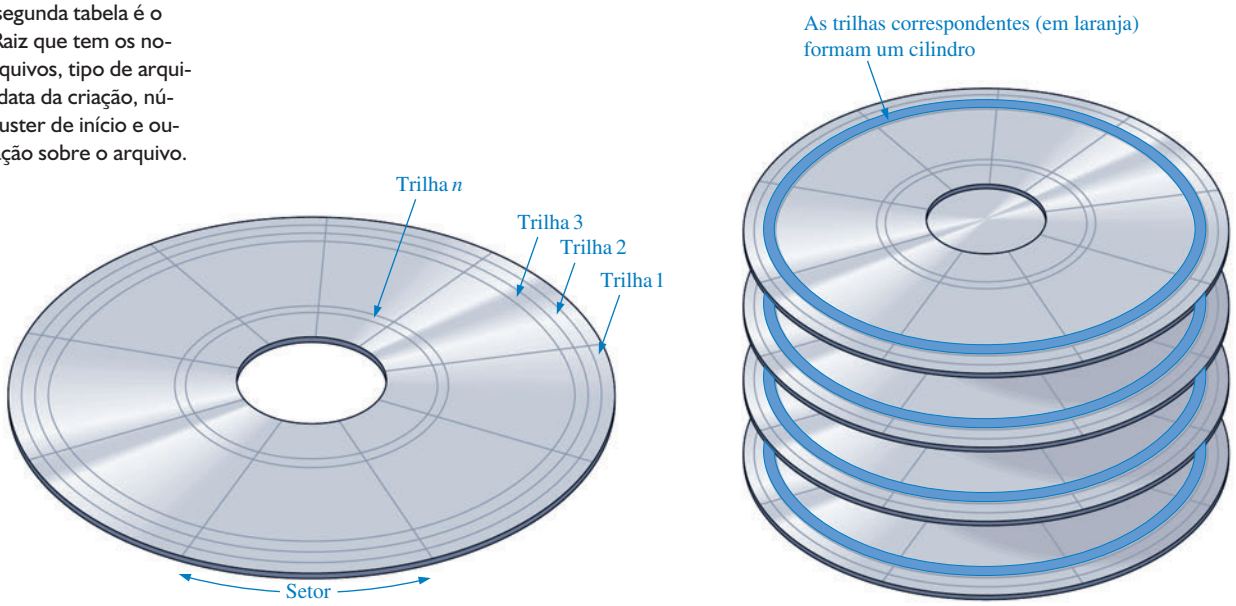
NOTA: COMPUTAÇÃO

Os dados são armazenados num disco rígido na forma de arquivos. Se manter na trilha da localização de um arquivo é o trabalho do dispositivo driver que gerencia o drive de disco rígido (algumas vezes chamado de BIOS do drive de disco rígido). O dispositivo driver e o sistema operacional do computador podem acessar duas tabelas para obter a trilha dos arquivos e os nomes dos arquivos. A primeira tabela é denominada de FAT (*File Alocation Table* – Tabela de Alocação de Arquivos). A FAT mostra o que está associado aos arquivos específicos e mantém uma identificação dos setores abertos e dos setores com problemas. A segunda tabela é o Diretório Raiz que tem os nomes dos arquivos, tipo de arquivo, hora e data da criação, número do cluster de início e outra informação sobre o arquivo.

Armazenamento Magnético

Discos Rígidos Magnéticos Os computadores usam discos rígidos como um meio de armazenamento em massa interno. Os **discos rígidos** são discos ("pratos") rígidos feitos de uma liga de alumínio ou uma mistura de vidro e cerâmica cobertos com uma camada magnética. Os drivers de discos rígidos se dividem principalmente em dois tamanhos, 5,25 polegadas e 3,5 polegadas de diâmetro, embora os diâmetros de 2,5 e 1,75 polegadas também sejam comercializados. Um drive de disco rígido é hermeticamente selado para manter o disco livre de poeira.

Tipicamente, dois ou mais discos são empilhados uns sobre os outros com um eixo comum em torno do qual gira o conjunto em vários milhares de rpm. Uma separação entre cada disco permite o acesso de uma cabeça de leitura/escrita magnética que é montada na extremidade de um braço atuador, conforme mostra a Figura 10–58. Existe uma cabeça de leitura/escrita para os dois lados de cada disco visto que os dados são gravados nos dois lados da superfície do disco. O braço atuador do drive sincroniza todas as cabeças de leitura/escrita mantendo-as em perfeito alinhamento conforme elas "voam" sobre a superfície do disco com uma separação de apenas uma fração de um milímetro do disco. Uma pequena partícula de sujeira poderia fazer com que uma cabeça danifique ("crach") a superfície do disco.



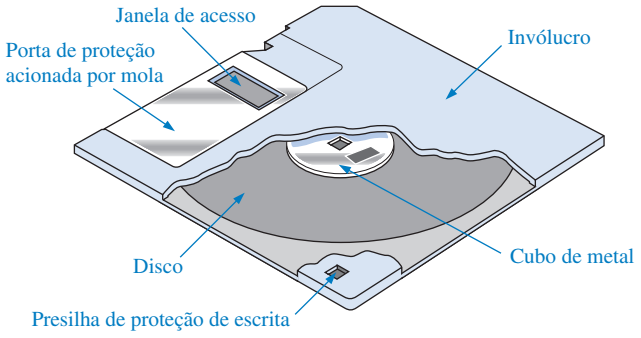
▲ **FIGURA 10–60**
Organização e formatação de um disco rígido.

Desempenho de um Disco Rígido Diversos parâmetros básicos determinam o desempenho de um driver de disco rígido. Uma operação de *busca* é o movimento da cabeça de leitura/escrita para a trilha desejada. O **tempo de busca** é a duração média dessa operação. Tipicamente, os drivers de discos rígidos têm um tempo médio de busca de vários milissegundos, dependendo do drive em particular.

O **período de latência** é o tempo para que o setor desejado gire sob a cabeça uma vez que esta esteja posicionada na trilha desejada. A pior situação ocorre quando o setor desejado acabou de passar pela posição da cabeça e está se distanciando dela. O setor tem que girar quase uma revolução completa para posicionar a cabeça. O *período de latência médio* considera que o disco tenha que fazer metade de uma revolução. Obviamente, o período de latência depende da velocidade rotacional constante do disco. As velocidades de rotação dos discos são diferentes para drivers distintos, mas, em geral, são de 3600 rpm, 4500 rpm, 5400 rpm e 7200 rpm. Alguns drivers de disco giram a 10.033 rpm e têm um período de latência médio de menos de 3 ms.

A soma do tempo de busca médio e do período de latência médio é o tempo de acesso para o drive de disco rígido.

Disquetes O disquete é uma tecnologia mais antiga, e o seu nome deriva do meterial com o qual é feito, poliéster flexível com uma camada magnética nos dois lados. Os primeiros disquetes foram de 5,25 polegadas de diâmetro e eram encapsulados num envelope semi-flexível. Os **disquetes** atuais (ou floppy disks) são de 3,5 polegadas de diâmetro e encapsulados num envelope de plástico rígido, como mostrado na Figura 10–61. Uma porta de proteção acionada por mola cobre a janela de acesso que permanece fechada até que o disco seja inserido no drive de disco. Um cubo de metal tem um furo para centralizar o disco e um outro para girá-lo dentro do envelope de proteção. Obviamente, os disquetes são discos removíveis, ao passo que os discos rígidos não são. Os disquetes são formatados em trilhas e setores de forma similar aos discos rígidos, exceto pelo número de trilhas e setores. Os disquetes de alta densidade (1,44 MB) têm 80 trilhas por lado com 18 setores.



◀ **FIGURA 10–61**
Disco flexível de 3,5 polegadas (disquete).

Zip™ O Zip drive, como é conhecido, é um tipo de dispositivo removível de armazenamento magnético que têm substituído os disquetes de capacidade limitada. Semelhante ao disquete, o cartucho de **disco Zip** é constituído de um disco flexível alojado num invólucro rígido de tamanho semelhante ao disquete, porém mais espesso. O drive de zip típico é muito mais rápido que o drive de disquete porque ele gira a 3000 rpm comparado à taxa de 300 rpm do disquete. O zip drive tem uma capacidade de armazenamento de até 250 MB, 173 vezes superior à capacidade de 1,44 MB do disquete.

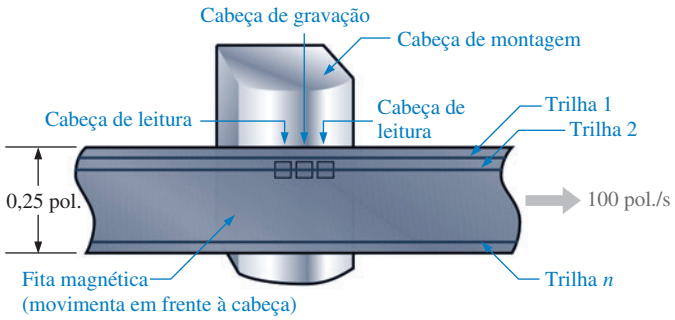
Jaz™ Um outro tipo de dispositivo removível de armazenamento magnético é o Jaz drive, similar ao drive de disco rígido, exceto que os dois discos internos são alojados num cartucho removível protegido por uma janela à prova de poeira. Os **cartuchos Jaz** são comercializados com capacidades de 1 ou 2 GB.

Disco Rígido Removível Além dos populares discos removíveis Zip e Jaz, drives de disco rígido removíveis com capacidades de 80 GB a 250 GB são comercializados. Tenha em mente que a

tecnologia muda tão rápido que provavelmente avanços estejam acontecendo enquanto você se dedica a este texto.

Fita Magnética A fita é usada para backup de dados a partir de dispositivos de armazenamento em massa e tipicamente é mais lenta que os discos porque os dados na fita são acessados de forma serial em vez de aleatória. Existem vários tipos comercializados, incluindo QIC, DAT, 8 mm e DLT.

QIC é uma abreviação para *quarter-inch cartridge* (cartucho de um quarto de polegada) e se parece muito com a fita cassete de áudio cm dois carretéis internos. Vários padrões QIC possuem de 36 a 72 trilhas que podem armazenar de 80 MB a 1,2 GB. Inovações mais recentes no padrão Travan têm prolongado a fita e aumentado sua largura permitindo capacidades de armazenamento de até 4 GB. O drive de fita QIC usa cabeças de leitura/escrita que tem uma única cabeça de escrita e com uma cabeça de leitura em cada lado. Isso permite ao drive de fita verificar dados escritos há pouco quando a fita gira em qualquer direção. No modo de gravação, a fita se move sob as cabeças de leitura/escrita a aproximadamente 100 polegadas/segundo, conforme indicado na Figura 10–62.



▲ **FIGURA 10–62**

Fita QIC.

DAT, abreviação de *digital audio tape* (fita de áudio digital), usa uma técnica denominada gravação por varredura helicoidal. Os sistemas DAT oferecem capacidades de armazenamento acima de 12 GB, mas são mais caros que o sistema QIC.

Um terceiro formato de fita, a fita de 8 mm, foi desenvolvida originalmente pela indústria de vídeo, mas tem sido adotada pela indústria de computadores como uma forma confiável de armazenamento de grandes quantidades de dados de computadores. O padrão 8 mm é similar ao DAT, mas oferece capacidades de armazenamento de até 25 GB.

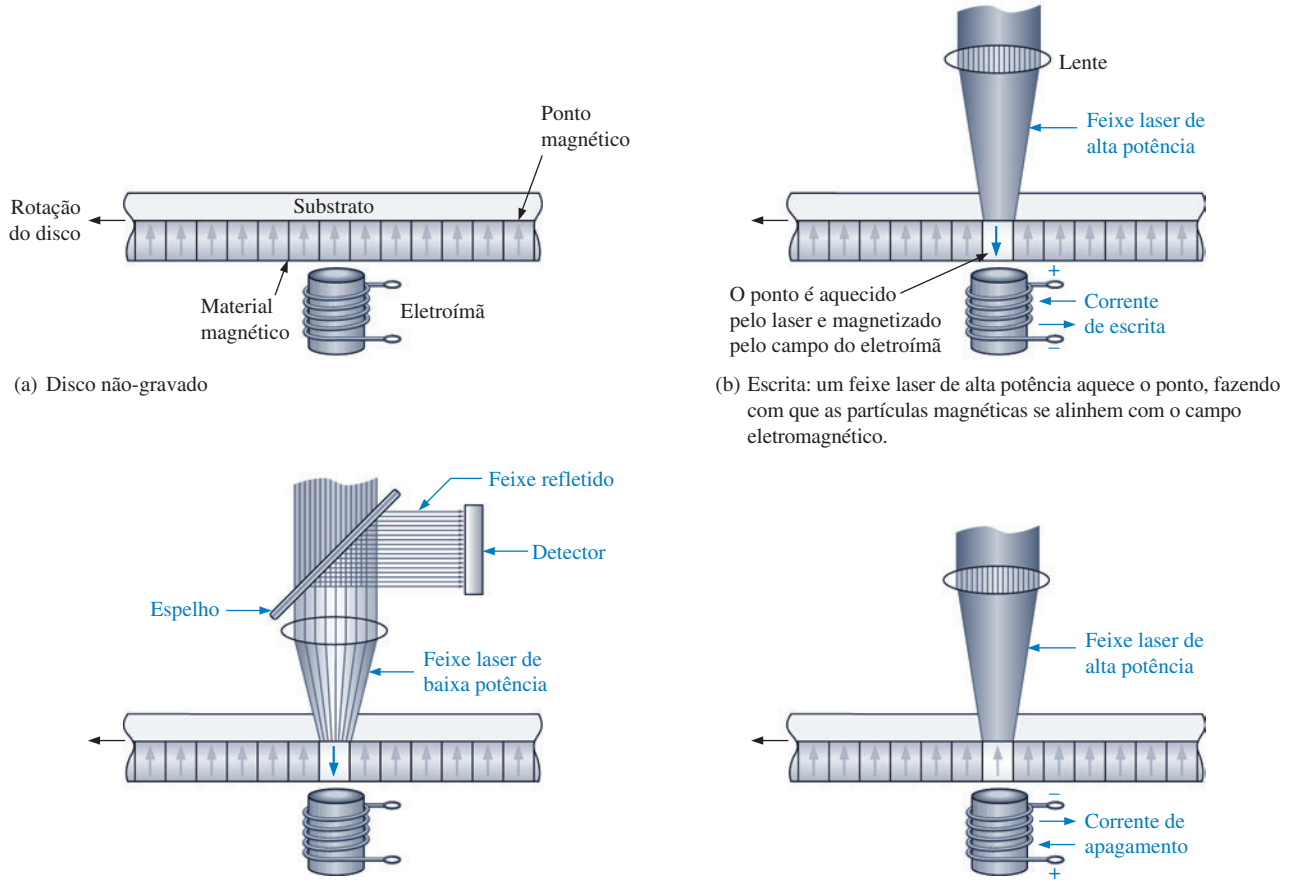
DLT é uma abreviação de *digital linear tape* (fita linear digital). DLT é uma fita com largura de meia polegada, que é 60% mais larga que a fita de 8 mm e, é claro, duas vezes a largura do padrão QIC. Basicamente, a fita DLT difere na forma com que o mecanismo de acionamento da fita trabalha para minimizar o desgaste da fita em comparação com outros sistemas. O padrão DLT oferece a maior capacidade de armazenamento de todos os formatos de fita com capacidades acima de 35 GB.

Armazenamento Magneto-Óptico

Como o nome sugere, os dispositivos de armazenamento magneto-óptico (MO) usam uma combinação das tecnologias magnética e óptica (laser). Um **disco magneto-óptico** é formatado em trilhas e setores de forma similar aos discos magnéticos.

A diferença básica entre um disco puramente magnético e um disco MO é que a cobertura magnética usada no disco MO requer um aquecimento para alterar a polarização magnética. Portanto, a tecnologia MO é extremamente estável à temperatura ambiente, fazendo dos dados inalteráveis. Para escrever um bit de dado, um feixe de laser de alta potência é focalizado num minúsculo ponto do disco, sendo a temperatura daquele minúsculo ponto aumentada acima do nível de temperatura denominado de ponto Curie (cerca de 200ºC). Uma vez aquecido, as partícu-

las magnéticas nesse ponto podem facilmente ter a sua direção (polarização) alterada por um campo magnético gerado pela cabeça de escrita. A informação é lida a partir do disco com um laser de potência menor que o usado para a escrita, fazendo uso do efeito Kerr onde a polaridade da luz do laser refletido é alterada dependendo da orientação das partículas magnéticas. Os pontos de uma polaridade representam 0s e os de polaridade oposta representam 1s. A operação básica de um disco MO é mostrada na Figura 10–63, que representa uma pequena área de seção reta de um disco.



(a) Disco não-gravado

(b) Escrita: um feixe laser de alta potência aquece o ponto, fazendo com que as partículas magnéticas se alinhem com o campo eletromagnético.

(c) Leitura: um feixe laser de baixa potência reflete das partículas magnéticas polarizadas reversamente e a sua polarização muda. Se as partículas não estiverem reversamente polarizadas, a polarização do feixe refletido não se altera.

(d) Apagamento: o campo eletromagnético é invertido conforme um feixe laser de alta potência aquece o ponto, fazendo com que as partículas magnéticas retornem para a polaridade original.

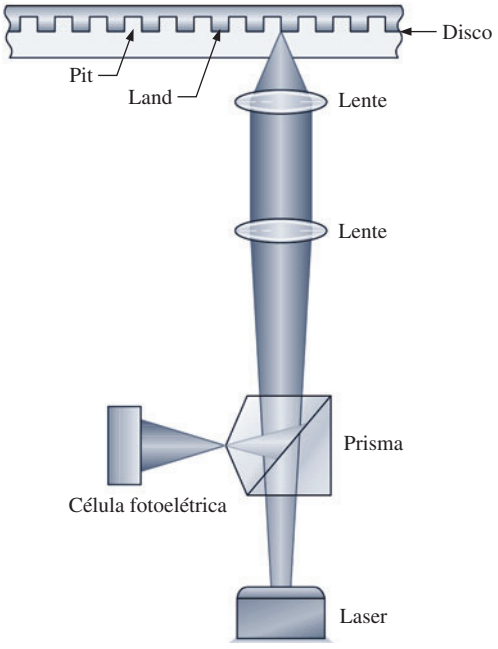
▲ **FIGURA 10–63**

Princípio básico de um disco magneto-óptico.

Armazenamento Óptico

CD-ROM A memória apenas de leitura em disco compacto (CD-ROM) consiste em um disco de 120 mm de diâmetro com uma cobertura de três camadas: um plástico policarbonato na parte inferior, uma fina folha de alumínio para proporcionar reflexão e uma cobertura superior de verniz para proteção. O **CD-ROM** é formatado numa única trilha espiral com setores seqüenciais de 2 kB e tem uma capacidade de 680 MB. Os dados são pré-gravados na fábrica na forma de minuciosos entalhes denominados *pits* (um pequeno fosso) e a área plana ao redor dos *pits* é chamada de *lands* (terra). Os pits são estampados na camada de plástico e não podem ser apagados.

Um aparelho leitor de CD lê os dados a partir da trilha em espiral com um laser infravermelho de baixa potência, conforme ilustrado na Figura 10–64. Os dados estão na forma de *pits* e *lands* como mostrado. A luz do laser refletida de um *pit* é 180º defasada da luz refletida pelos *lands*. À medida que o disco gira, o estreito feixe de laser choca contra uma série de *pits* e *lands* de comprimentos que variam, e um fotodiodo detecta a diferença na luz refletida. O resultado é uma série de 1s e 0s que correspondem à configuração de *pits* e *lands* ao longo da trilha.



▶ **FIGURA 10–64**

Operação básica de leitura de dados num CD-ROM.

WORM O armazenamento óptico do tipo **WORM** (*Write Once/Read Many* – uma escrita e várias leituras) é um armazenamento no qual os dados podem ser escritos uma vez, não podendo ser apagados, e lidos diversas vezes. Para escrever os dados, é usado um laser de baixa potência para criar *pits* microscópicos na superfície do disco. Os 1s e os 0s são representados por áreas com e sem *pits*.

CD-R Esse é essencialmente um tipo de WORM. A diferença é que o CD-Gravável (CD-R) permite múltiplas seções de escrita para diferentes áreas do disco. O **CD-R** tem uma trilha em espiral semelhante ao CD-ROM, mas em vez de imprimir os entalhes mecanicamente no disco para representar os dados, o CD-R usa um laser para "queimar" pontos microscópicos numa tintura orgânica superficial. Quando aquecido além de uma temperatura crítica com um laser durante a leitura, os pontos "queimados" mudam de cor e refletem menos luz que as áreas não "queimadas". Portanto, 1s e 0s são representados num CD-R por áreas "queimadas" e "não-queimadas", ao passo que no CD-ROM eles são representados por *pits* e *lands*. Semelhante ao CD-ROM, os dados não podem ser apagados uma vez escritos.

CD-RW O CD-Regravável (CD-RW) pode ser usado para leitura e escrita de dados. Em vez de uma camada de gravação baseada numa tintura como no CD-R, o **CD-RW** normalmente usa um composto cristalino com uma propriedade especial. Quando ele é aquecido a uma certa temperatura, ele se torna cristalino quando esfria; mas se for aquecido a uma temperatura maior, ele derrete e se torna amorfo quando esfria. Para escrever dados, o feixe de laser focado aquece o material à temperatura de derretimento resultando em um estado amorfo. As áreas amorfas resultantes refletem menos a luz que as áreas cristalinas, permitido que na operação de leitura se detecte 1s e 0s. Os dados podem ser apagados ou sobrescritos aquecendo as áreas amorfas a uma temperatura acima da temperatura de cristalização, porém abaixo da temperatura de derretimento fazendo com que o material amorfo reverta para o estado cristalino.

DVD-ROM Originalmente DVD era uma abreviação para *Digital Video Disk* (Disco de Vídeo Digital), mas eventualmente veio representar *Digital Versatile Disk*. Semelhante ao CD-ROM, os dados no **DVD-ROM** são pré-armazenados no disco. Entretanto, o tamanho do *pit* é menor, permitindo que mais dados sejam armazenados numa trilha. A principal diferença entre o CD-ROM e o DVD-ROM é que o CD é de face única, enquanto o DVD tem dados nos dois lados. Além dos discos DVD serem de dupla face, existem também discos com múltiplas camadas, provendo capacidades de armazenamento de dezenas de gigabytes. Para acessar todas as camadas, o feixe de laser necessita reajustar o foco ao passar de uma camada para outra.

SEÇÃO 10–8 REVISÃO

1. Faça uma lista dos principais tipos de armazenamento magnético.
2. Qual é a capacidade de armazenamento de um disquete?
3. Como é organizado um disco magnético?
4. Como os dados são escritos e lidos em um disco magneto-óptico?
5. Faça uma lista dos tipos de armazenamento óptico.

10-9 ANÁLISE DE DEFEITO

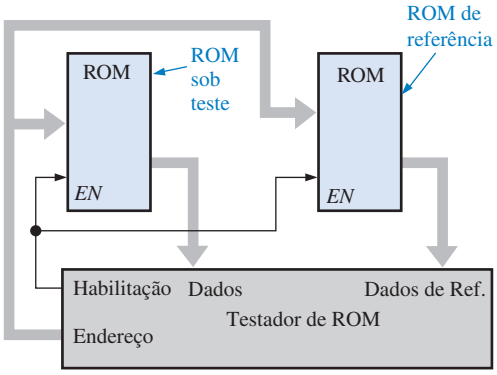
Como as memórias podem conter uma grande quantidade de células de armazenamento, o teste de cada célula pode ser um longo e frustrante processo. Felizmente, o teste de memória é geralmente um processo automatizado realizado com um instrumento de teste programável ou com a ajuda de software para teste no próprio sistema. A maioria dos sistemas microprocessados fornece testes automáticos de memória como parte do software do sistema.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir o método de *checksum* no teste de ROMs ■ Discutir o método do padrão xadrez de teste de RAMs

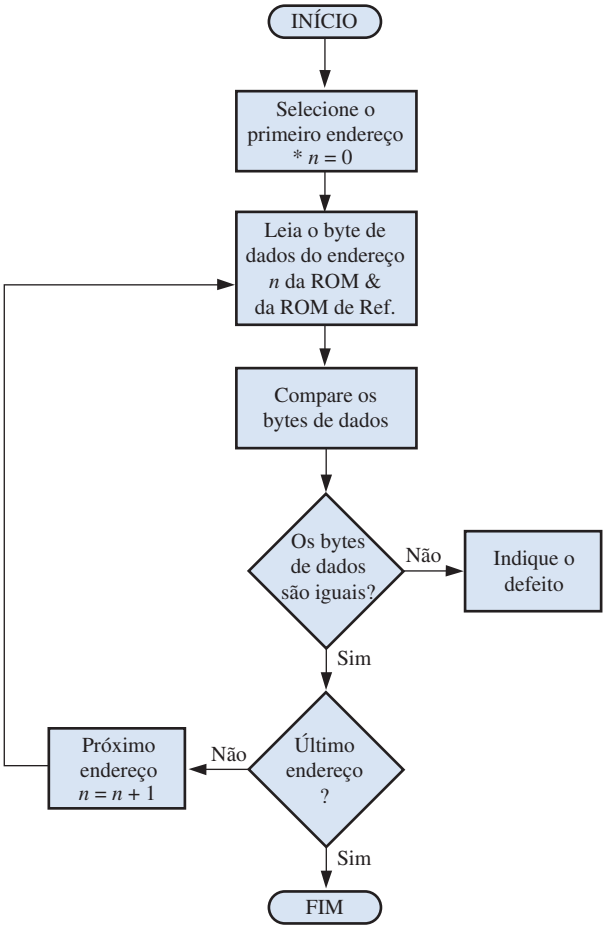
Teste de ROM

Como as ROMs contêm dados desconhecidos, elas podem ser verificadas em relação ao armazenamento correto de dados lendo cada palavra de dado da memória e comparando com uma palavra de dado que se sabe estar correta. Uma forma de fazer isso está ilustrada na Figura 10–65. Esse processo requer uma ROM de referência que contenha os mesmos dados que a ROM sob teste.



◀ **FIGURA 10–65**
Diagrama em bloco para a verificação completa do conteúdo de uma ROM.

Um equipamento especial de teste é programado para ler cada endereço nas duas ROMs simultaneamente e comparar os conteúdos. Um fluxograma, mostrado na Figura 10–66, ilustra a seqüência básica.



* *n* é o número do endereço.

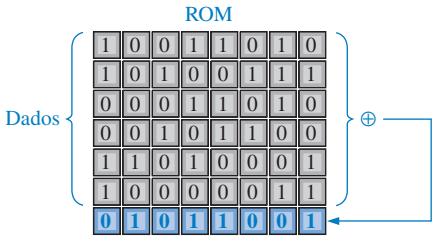
▲ **FIGURA 10–66**

Fluxograma para uma verificação completa do conteúdo de uma ROM.

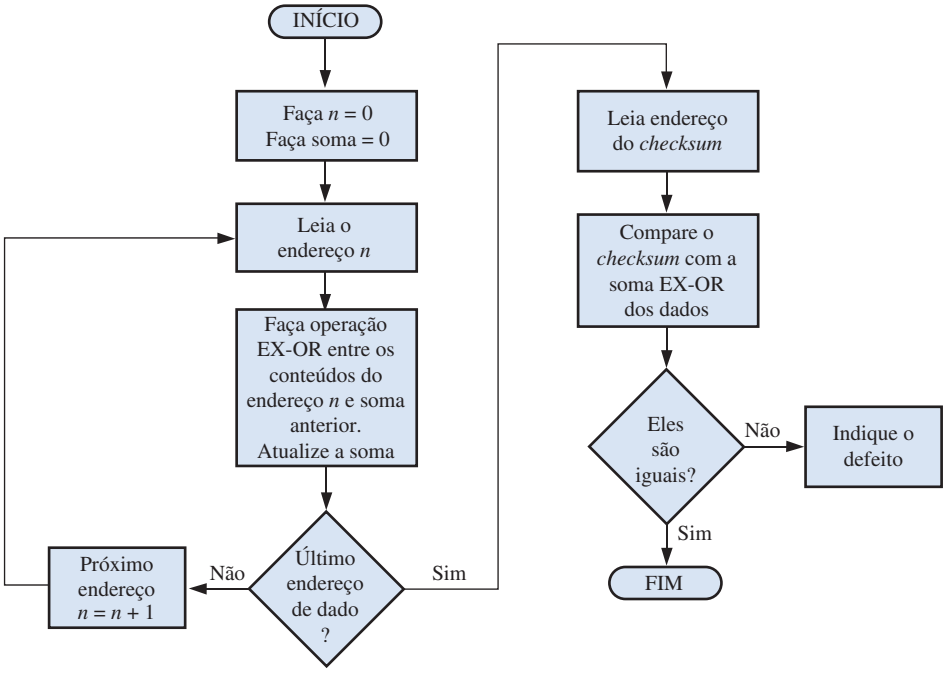
Método Checksum Embora o método anterior verifique cada endereço da ROM em relação ao dado correto, ele tem a desvantagem da necessidade de uma ROM de referência para cada ROM diferente a ser testada. Além disso, uma falha na ROM de referência pode produzir uma indicação de defeito.

No método *checksum*, um número, a soma dos conteúdos de todos os endereços da ROM, é armazenado num determinado endereço da ROM quando a ROM é programada. Para testar a ROM, o conteúdo de todos os endereços, exceto o *checksum*, são somados e o resultado é comparado com o *checksum* armazenado na ROM. Se existir uma diferença, existe definitivamente um defeito. Se os *checksums* forem iguais, a ROM está provavelmente boa. Entretanto, existe uma possibilidade remota que uma combinação de células de memórias ruins poderiam provocar *checksums* iguais.

Esse processo é ilustrado na Figura 10–67 com um exemplo simples. O *checksum* nesse caso é produzido fazendo a soma de cada coluna dos bits de dados e descartando os carries. Isso é na realidade uma soma EX-OR de cada coluna. O fluxograma mostrado na Figura 10–68 ilustra o teste *checksum* básico.



◄ **FIGURA 10–67**
Ilustração simplificada de uma ROM programada com o *checksum* armazenado do endereço indicado.



◄ **FIGURA 10–68**
Fluxograma para um teste *checksum* básico.

O teste *checksum* pode ser implementado com um equipamento especial de teste, ou ele pode ser incorporado como uma rotina de teste embutida (sistema) no software ou sistemas microprocessados. Nesse caso, a rotina de teste da ROM é automaticamente executada ao inicializar o sistema.

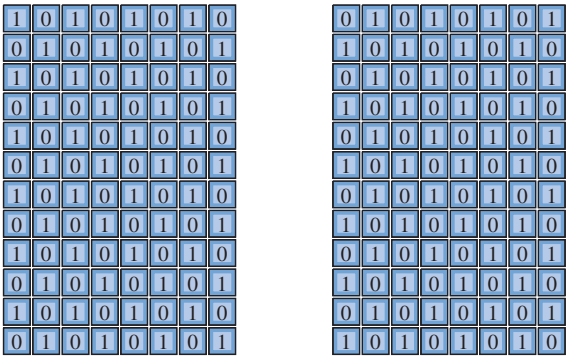
Teste de RAM

Para testar uma RAM verificando sua habilidade de armazenar 0s e 1s em cada célula, primeiro os 0s são escritos em todas as células de cada endereço e em seguida lidos e verificados. Em seguida, os 1s são escritos em todas as células de cada endereço e em seguida lidos e verificados. Esse teste básico detectará uma célula que está fixa no estado 1 ou no estado 0.

Alguns defeitos de memória não podem ser detectados com o teste de todos 0s e 1s. Por exemplo, duas células de memória adjacentes estão em curto-circuito, elas estarão sempre no mesmo estado, ambas em 0 ou ambas em 1. Além disso, o teste de todos 0s e todos 1s será ineficaz se existirem problemas de ruídos internos tal que o conteúdo de um ou mais endereços sejam alterados pela mudança no conteúdo de um outro endereço.

O Teste Padrão Xadrez Uma maneira de testar de forma mais completa uma RAM é usando um padrão xadrez de 1s e 0s, conforme ilustrado na Figura 10–69. Observe que todas as células adjacentes têm bits opostos. Esse padrão verifica a existência de curto-circuito entre células adjacentes; se houver um curto-circuito, duas células estarão no mesmo estado.

Após a RAM ser verificada com o padrão mostrado na Figura 10–69(a), o padrão é invertido, como mostra a parte (b) da figura. Essa inversão verifica a habilidade de todas as células armazenarem tanto 1s quanto 0s.

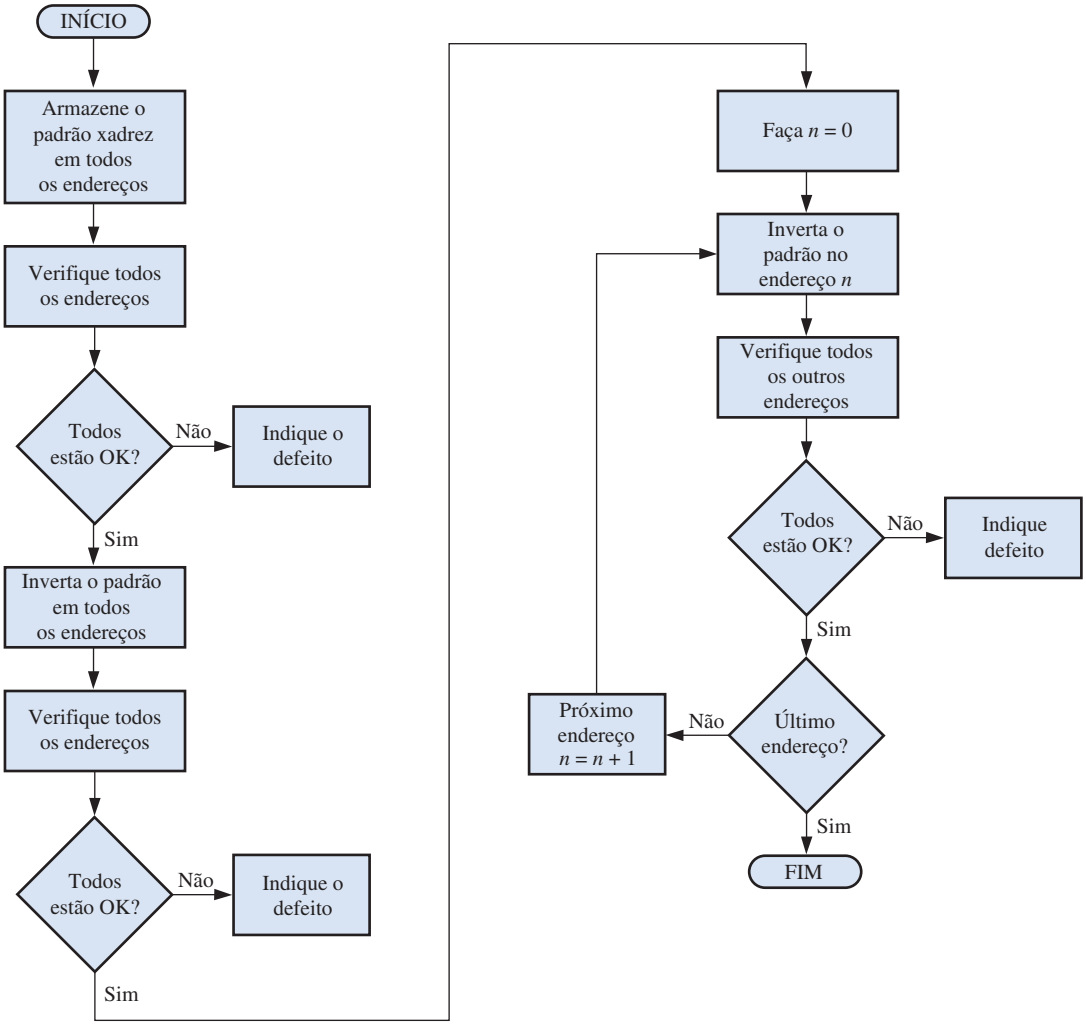


▶ **FIGURA 10–69**

Padrão de teste xadrez para uma RAM.

Um teste posterior é alternar o padrão um endereço de cada vez e verificar todos os outros endereços para o padrão próprio. Esse teste identifica um problema no qual os conteúdos de um endereço são alterados dinamicamente quando os conteúdos de outro endereço mudam.

Um procedimento básico para o teste com padrão xadrez é ilustrado pelo fluxograma dado na Figura 10–70. O procedimento pode ser implementado com o software do sistema num sistema



▲ **FIGURA 10–70**

Fluxograma para o teste do padrão xadrez de uma RAM.

microprocessado de forma que quaisquer testes sejam automáticos quando o sistema for energizado ou podem ser iniciados a partir do teclado.

SEÇÃO 10–9 REVISÃO

1. Descreva o método *checksum* de teste de ROM.
2. Por que o método *checksum* não pode ser aplicado no teste de uma RAM?
3. Faça uma lista com os três defeitos básicos que o teste com o padrão xadrez pode detectar numa RAM.

APLICAÇÕES EM SISTEMAS DIGITAIS

Nessa aplicação, é desenvolvido o circuito lógico da memória do sistema de segurança que foi introduzido no Capítulo 9. O circuito lógico do código de segurança, que foi completado anteriormente, será combinado com o circuito lógico da memória para formar o sistema completo.

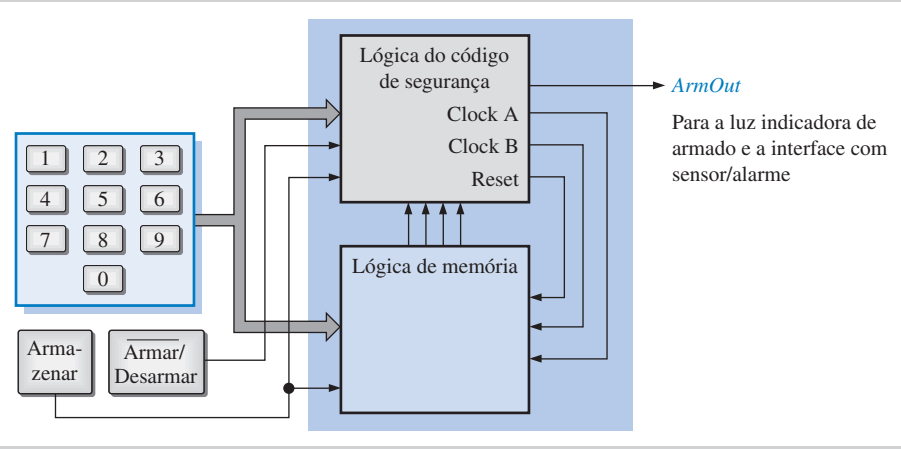
Operação Geral

O diagrama em bloco do sistema de segurança completo é mostrado na Figura 10–71. O circuito lógico da memória armazena um código de segurança de 4 dígitos no formato BCD. No modo *desarmar*, os quatro dígitos são inseridos na memória a partir do teclado. Uma vez armazenados na memória, os quatro dígitos BCD passam a ser o código de segurança de entrada no local. Se for necessário alterar o código, a memória é reprogramada com um novo código.

A memória é programada colocando primeiro o sistema no modo *desarmar* e usando a chave armazenar e o teclado para inserir o código de 4 dígitos desejado. Essa é uma operação de *escrita* na memória. Uma vez que a memória foi programada com o código de segurança, a chave *Armar/Desarmar* é comutada para o modo *armar*, que configura a memória para operações de *leitura*. Um diagrama em bloco do circuito lógico da memória é mostrado na Figura 10–72.

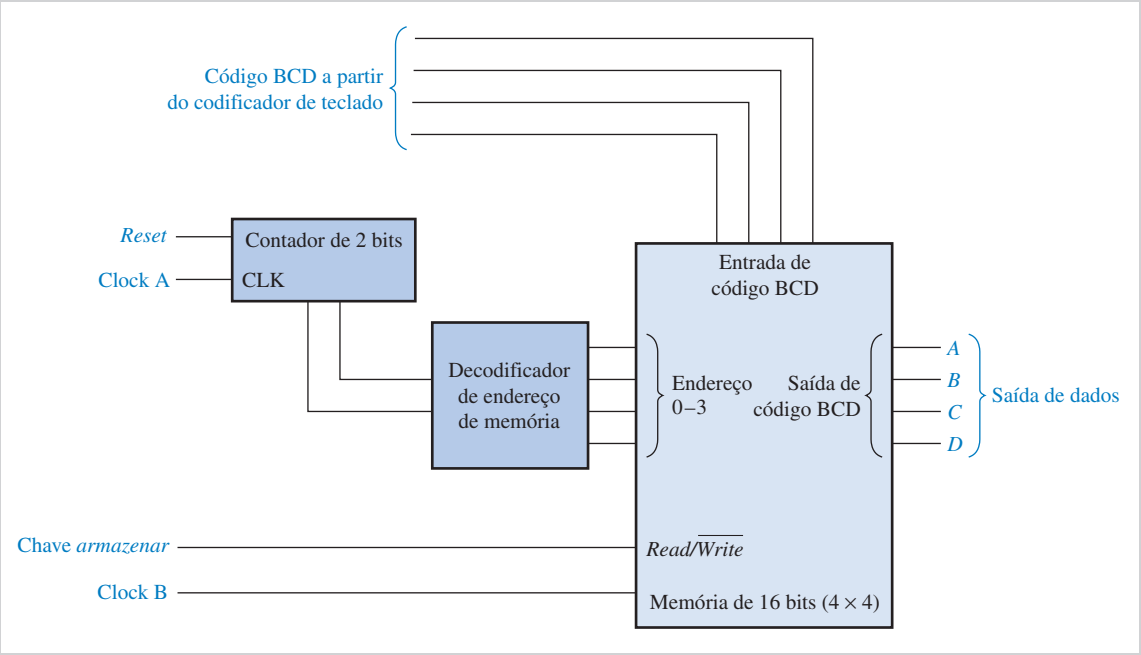
A Célula de Memória

A memória necessita de dezesseis células para armazenar os quatro dígitos BCD do código de segurança. Uma possibilidade de projeto para uma célula de memória é mostrada na Figura 10–73. Um flip-flop J-K é usado como elemento básico de armazenamento e pode ser operado em dois modos (*leitura* e *escrita*). No modo escrita, *AddSel* (seleção de endereço) é nível ALTO e a entrada $R/\overline{W}$ (read/$\overline{\text{write}}$) () é nível BAIXO. As portas G1 e G2 são habilitadas, o bit de entrada é aplicado na entrada *J* e o seu complemento é aplicado na entrada *K*. O bit de entrada é então armazenado na borda positiva do pulso de clock. No modo



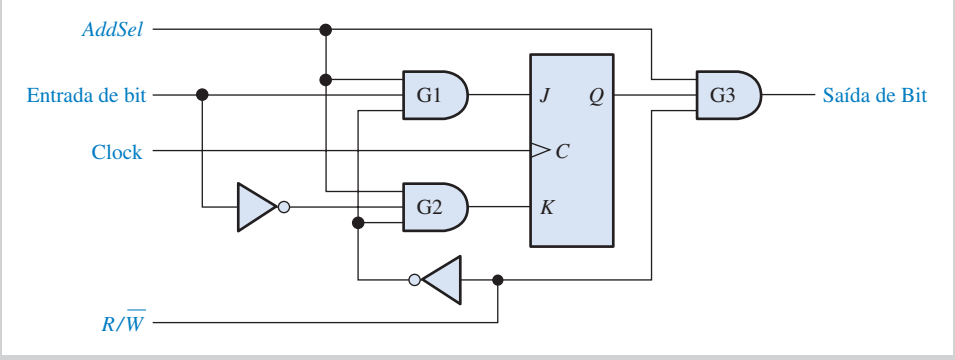
▲ **FIGURA 10–71**

Diagrama em bloco básico do sistema de segurança.



▲ **FIGURA 10–72**

Diagrama em bloco do circuito lógico da memória.



▲ **FIGURA 10–73**

Circuito lógico da célula de memória.

leitura, *AddSel* é nível ALTO e $R/\overline{W}$ é nível ALTO habilitando G3. O bit armazenado na saída *Q* do flip-flop aparece na saída de G3 (saída de bit).

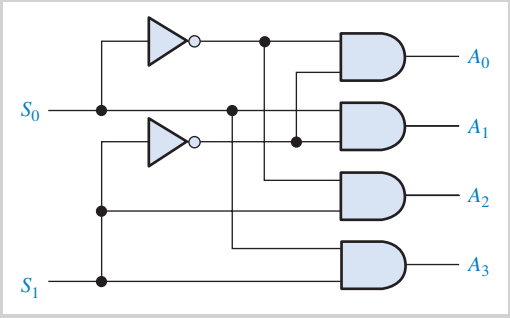
O Decodificador de Endereço da Memória

O circuito lógico do decodificador de endereço da memória é mostrado na Figura 10–74. Uma seqüência binária de 2 bits é aplicada nas entradas de seleção (S_0, S_1) para selecionar cada um dos quatro endereços de memória usando as linhas *AddSel*.

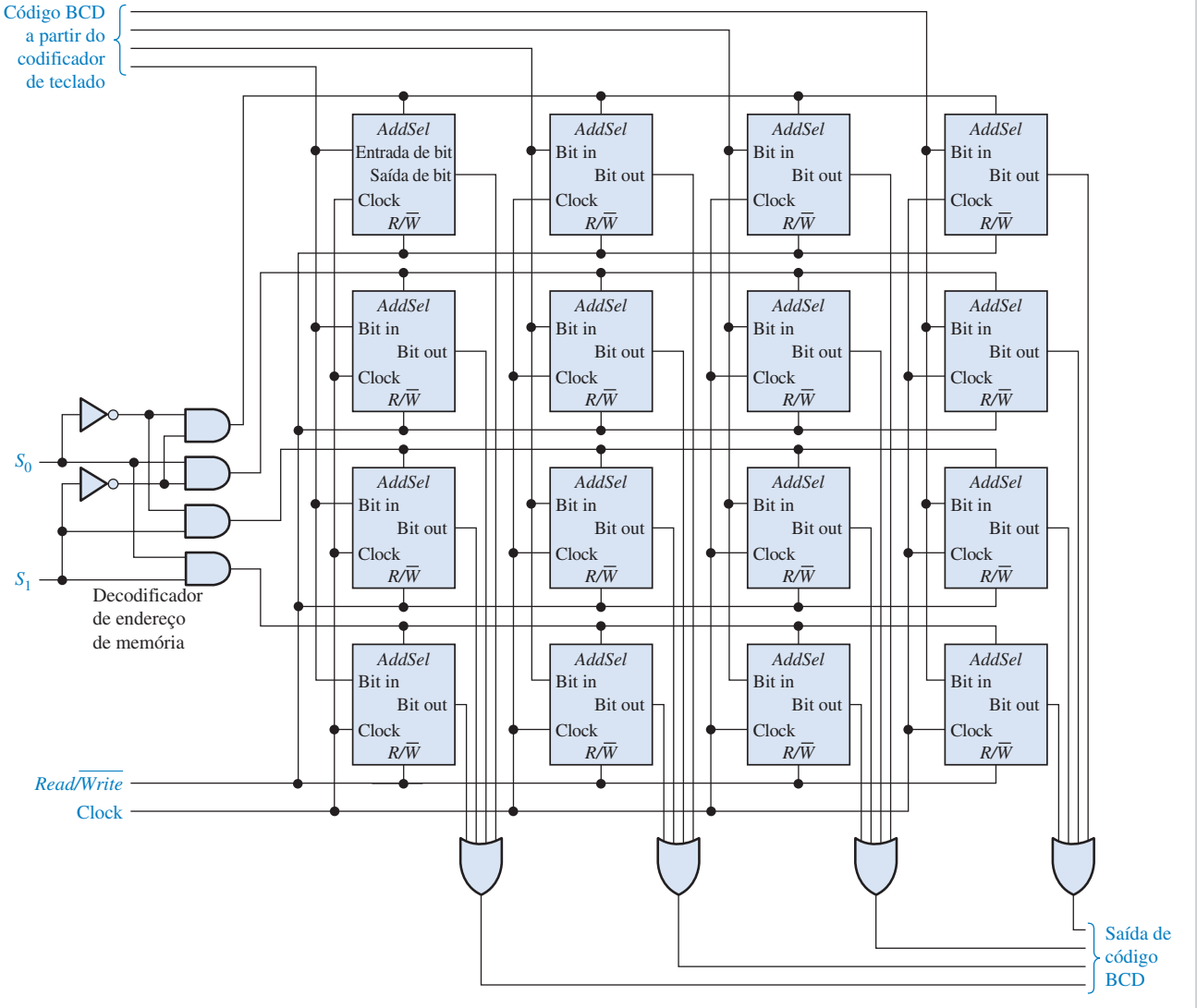
O Arranjo de Memória

A memória tem dezesseis células como mostra a Figura 10–75. Quando uma das linhas da memória é selecionada pelo decodificador de endereço e a entrada leitura/escrita for nível BAIXO, o código de entrada BCD de 4 bits é inserido nas quatro células selecionadas. As entradas do decodificador de endereço são alteradas seqüencialmente passando por cada um dos quatro estados (00, 01, 10 e 11) para selecionar seqüencialmente cada linha da memória.

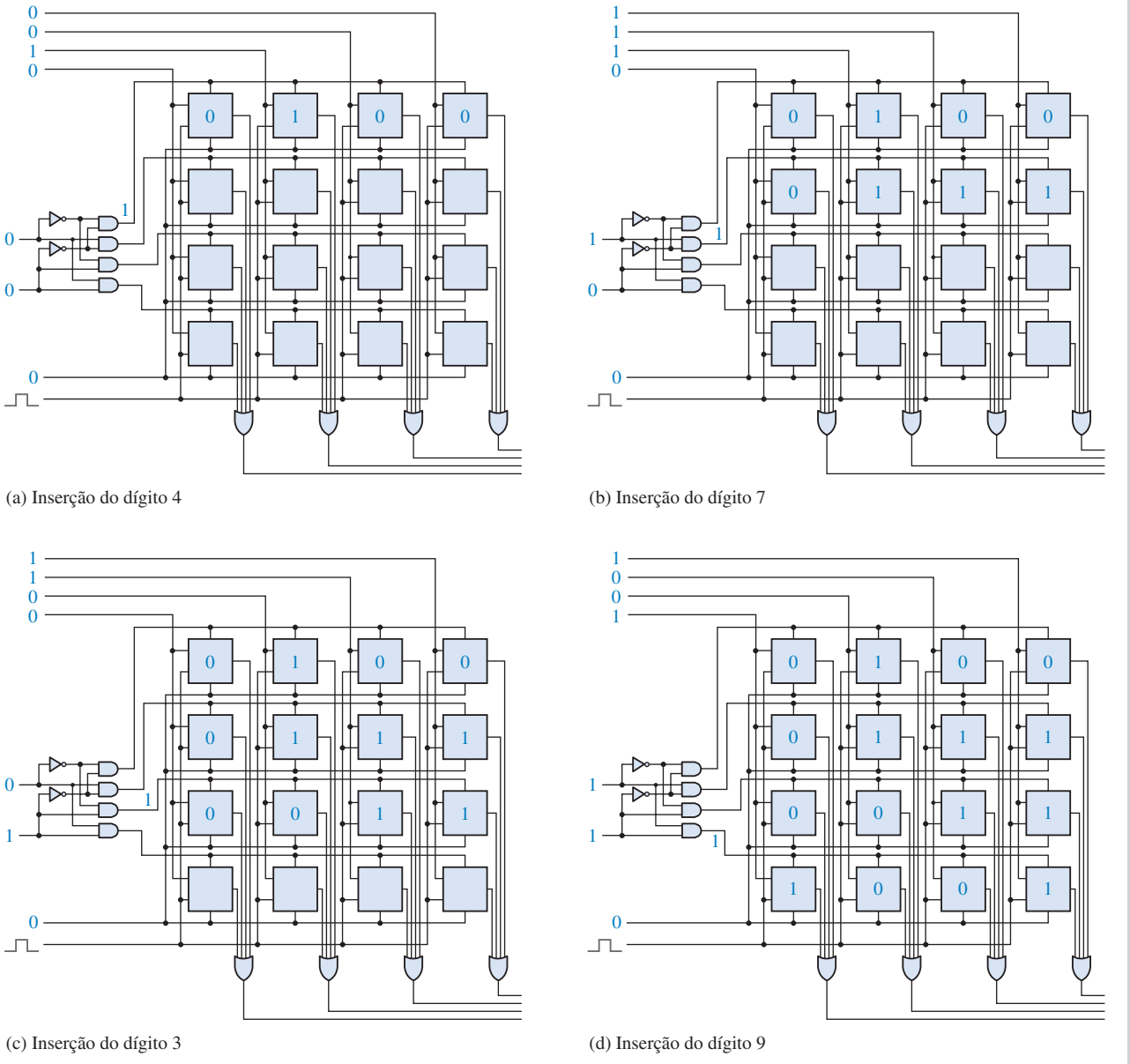
A Figura 10–76 ilustra a programação da memória conforme o código de segurança 4739 é inserido seqüencialmente.



▶ **FIGURA 10–74**
Decodificador de endereço da memória.



▲ **FIGURA 10–75**
Arranjo de memória e decodificador de endereço.



(a) Inserção do dígito 4

(b) Inserção do dígito 7

(c) Inserção do dígito 3

(d) Inserção do dígito 9

▲ **FIGURA 10–76**

Ilustração do armazenamento do código de segurança (4739) na memória.

O Circuito Lógico Completo da Memória

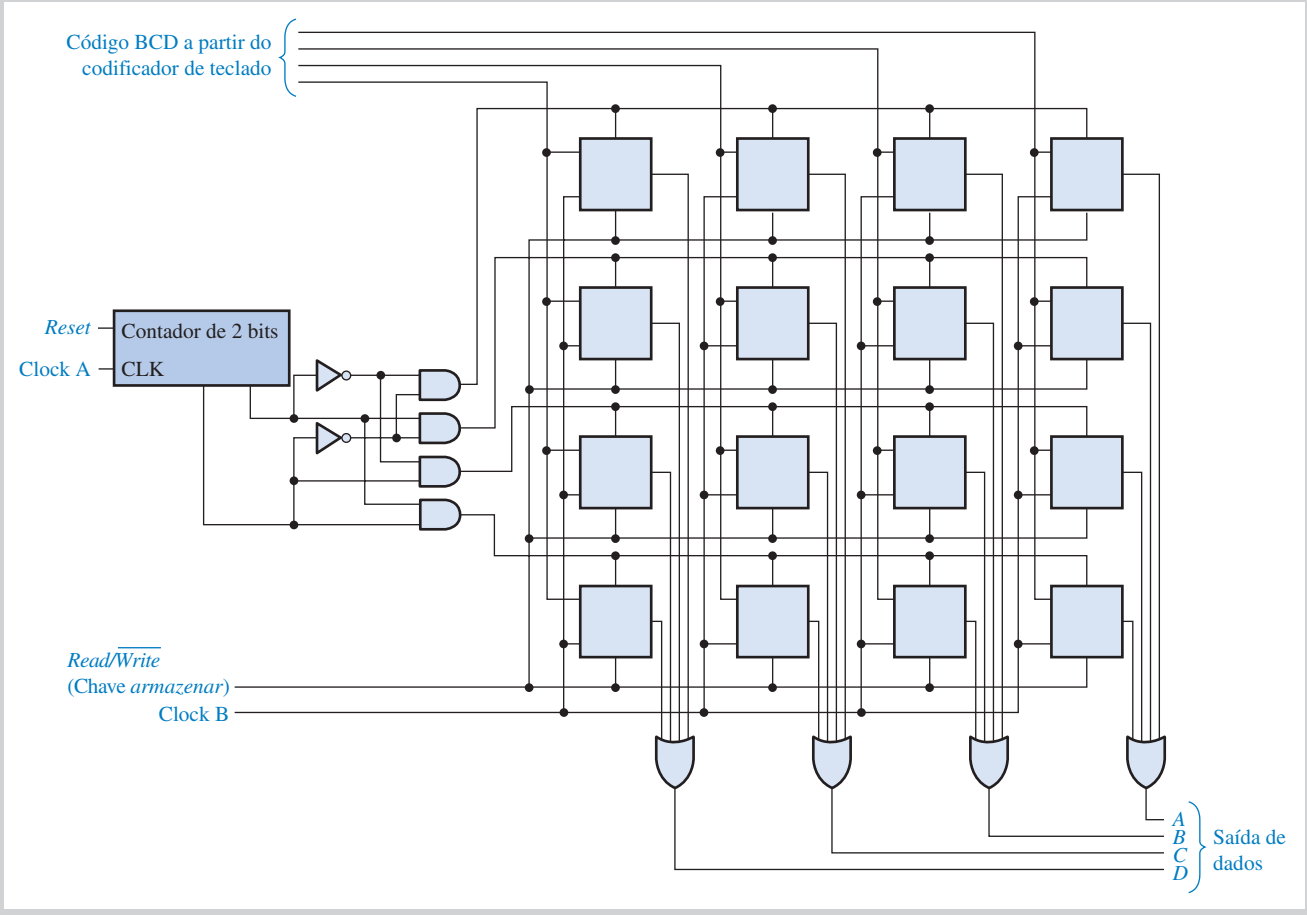
Um codificador de teclado é necessário para converter um tecla pressionada para o código BCD e um contador de 2 bits é usado para produzir a seqüência para seleção dos endereços da memória. Isso é mostrado na Figura 10–77. No início da programação, o contador é resetado para o estado 0 através de uma entrada de reset a partir do circuito lógico de inserção do código e avança pela sua seqüência a cada tecla pressionada.

O Sistema de Segurança Completo

Agora que o circuito lógico da memória está completo, podemos combiná-lo com o circuito lógico do código de segurança a partir do Capítulo 9 mostrado como um diagrama em bloco na Figura 10–78 para formar o sistema de segurança completo mostrado como um diagrama em bloco na Figura 10–79.

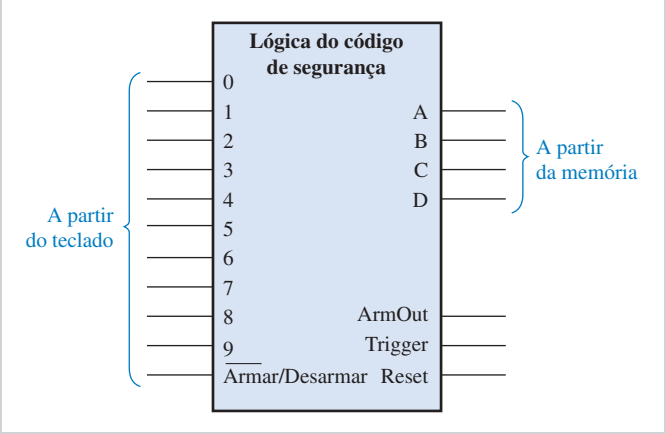
Atribuições do Sistema

- *Atividade 1* Explicar o que o codificador de teclado faz na memória.
- *Atividade 2* Explicar o que o contador de 2 bits faz na memória.
- *Atividade opcional* Construir o sistema de inserção de segurança completo usando dispositivos TTL padrão (74XX) e outros componentes necessários. Teste o sistema.



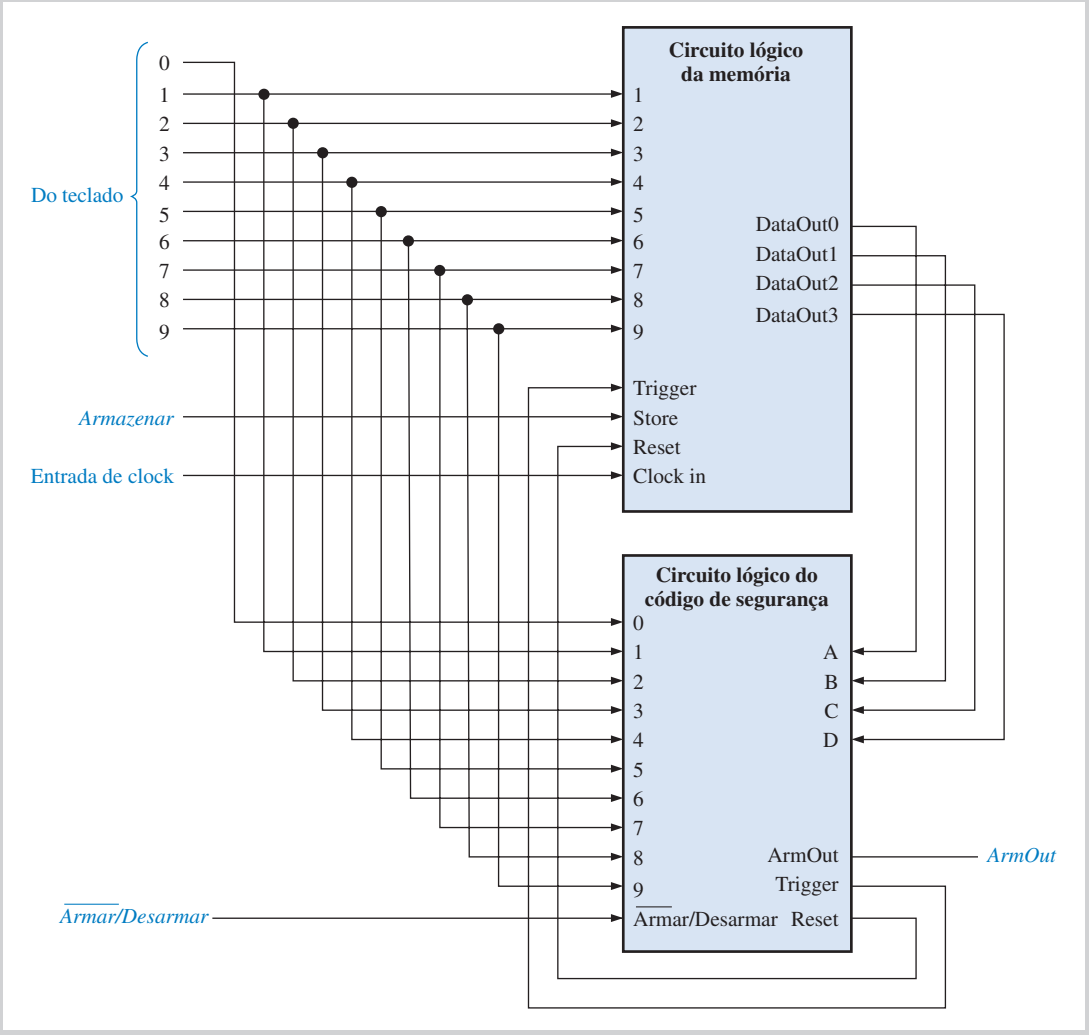
▲ **FIGURA 10–77**

O circuito lógico completo da memória.



▲ **FIGURA 10–78**

Circuito lógico do código de segurança (do Capítulo 11).

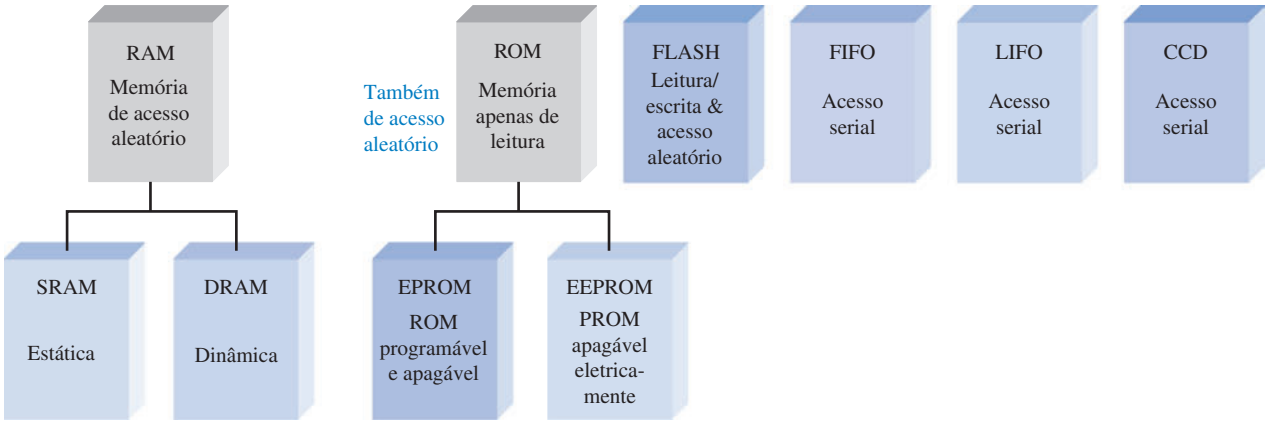


▲ **FIGURA 10–79**

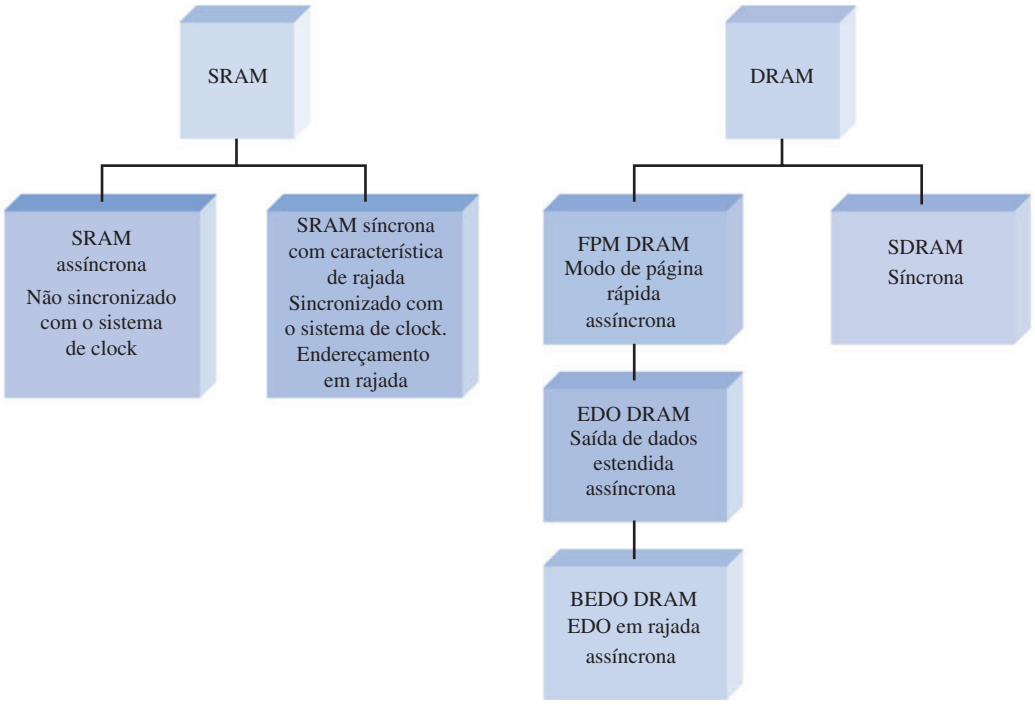
O sistema de segurança completo.

RESUMO

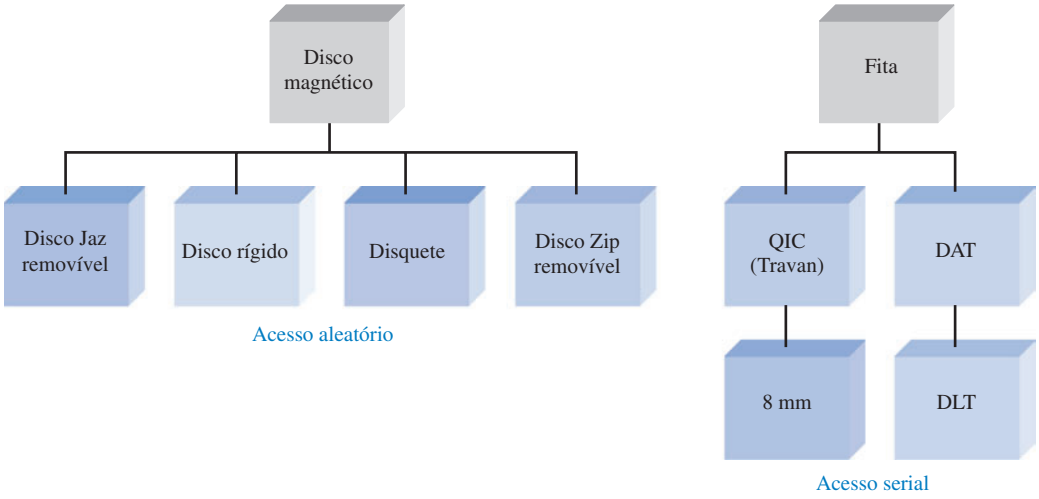
- Tipos de memórias semicondutoras:



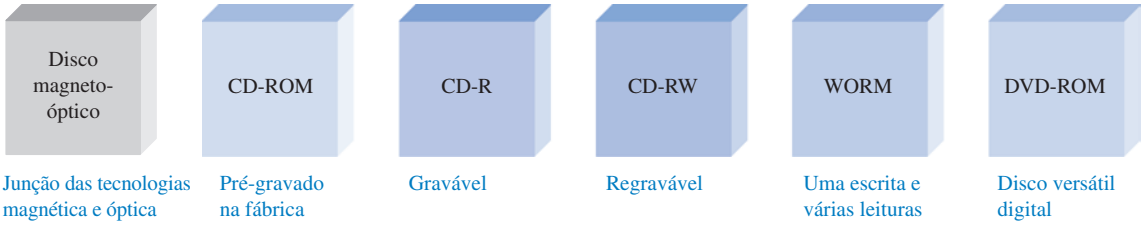
- Tipos de SRAMs (RAMs Estáticas) e DRAMs (RAMs Dinâmicas):



- Tipos de armazenamento magnético:



- Tipos de armazenamento óptico (laser):



TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Barramento Um conjunto de interconexões que faz a interface com um ou mais dispositivos baseado em especificações padronizadas.

Capacidade O número total de unidades de dados (bits, nibbles, bytes, words) que uma memória pode armazenar.

Célula Um elemento único de armazenamento numa memória.

Disco rígido Um dispositivo de armazenamento magnético; tipicamente, uma pilha de um ou mais discos rígidos confinados num invólucro hermeticamente selado.

DRAM Memória de acesso aleatório dinâmica; um tipo e memória semicondutora de leitura/escrita que usa capacitores como elementos de armazenamento sendo volátil.

Endereço A posição de uma determinada célula de armazenamento ou de um grupo de células numa memória.

EPROM Memória apenas de leitura programável e apagável; um tipo de dispositivo de memória semicondutora que usa tipicamente luz ultravioleta para apagar os dados.

Escrita O processo de armazenamento de dados numa memória.

FIFO Memória na qual o primeiro dado a entrar é o primeiro a sair.

Leitura O processo de recuperação de dados armazenados numa memória.

LIFO Memória na qual o primeiro dado a entrar é o último a sair; uma memória pilha.

Memória flash Uma memória semicondutora de acesso aleatório, de leitura/escrita e não-volátil na qual os dados são armazenados como carga na porta flutuante num certo tipo de FET.

PROM Memória apenas de leitura programável; um tipo de memória semicondutora.

RAM Memória de acesso aleatório; uma memória semicondutora de leitura/escrita volátil.

ROM Memória apenas de leitura; uma memória semicondutora de acesso aleatório não-volátil.

SRAM Memória de acesso aleatório estática; um tipo e memória semicondutora de leitura/escrita volátil.

Word Uma unidade completa de dado binário (palavra).

AUTOTESTE

As respostas estão no final do capítulo.

1. A capacidade em bits de uma memória que tem 1024 endereços e pode armazenar 8 bits em cada endereço é
   (a) 1024 **(b)** 8192 **(c)** 8 **(d)** 4096
2. Uma palavra de dados de 32 bits consiste de
   (a) 2 bytes **(b)** 4 nibbles
   (c) 4 bytes **(d)** 3 bytes e 1 nibble
3. Dados são armazenados numa memória de acesso aleatório (RAM) durante
   (a) a operação de leitura **(b)** a operação de habilitação
   (c) a operação de escrita **(d)** a operação de endereçamento
4. O dado que é armazenado num determinado endereço numa memória de acesso aleatório (RAM) é perdido quando
   (a) a alimentação é desligada.
   (b) o dado é lido do endereço.
   (c) um novo dado é escrito no endereço.
   (d) as alternativas (a) e (c) estão corretas.
5. Uma ROM é uma
   (a) memória não-volátil **(b)** memória volátil
   (c) memória de leitura/escrita **(d)** memória organizada em bytes
6. Uma memória com 256 endereços tem
   (a) 256 linhas de endereço **(b)** 6 linhas de endereço
   (c) 1 linha de endereço **(d)** 8 linhas de endereço

7. Uma memória organizada em bytes tem

(a) 1 linha de saída de dados **(b)** 4 linhas de saída de dados
(c) 8 linhas de saída de dados **(d)** 16 linhas de saída de dados

8. A célula de armazenamento de uma SRAM é um

(a) flip-flop **(b)** capacitor
(c) um fusível **(d)** um domínio magnético

9. Uma DRAM tem que ser

(a) substituída periodicamente. **(b)** reavivada periodicamente.
(c) sempre habilitada. **(d)** programada antes de cada uso.

10. Uma memória flash é

(a) volátil **(b)** uma memória apenas de leitura
(c) uma memória de leitura/escrita **(d)** não-volátil
(e) as alternativas (a) e (c) estão corretas. **(f)** as alternativas (c) e (d) estão corretas.

11. Disco rígido, disquete, disco Zip e disco Jaz são todos

(a) dispositivos de armazenamento magneto-óptico
(b) dispositivos de armazenamento semicondutor
(c) dispositivos de armazenamento magnético
(d) dispositivos de armazenamento óptico

12. Os dispositivos de armazenamento óptico empregam

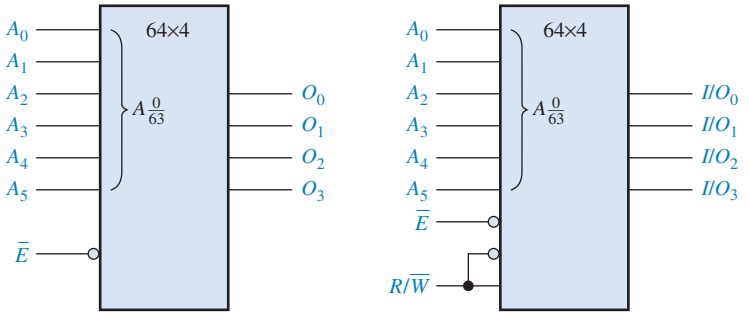
(a) luz ultravioleta **(b)** campos eletromagnéticos
(c) acopladores ópticos **(d)** lasers

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 10–1 Fundamentos de Memória Semicondutora

1. Identifique a ROM e a RAM na Figura 10–80.



▶ FIGURA 10–80

2. Explique por que RAMs e ROMs são memórias de acesso aleatório.

3. Explique a finalidade do barramento de endereço e do barramento de dados.

4. Qual endereço de memória (de 0 a 256) está representado por cada um dos seguintes números hexadecimais:

(a) $0A_{16}$ **(b)** $3F_{16}$ **(c)** CD_{16}

SEÇÃO 10–2 Memórias de Acesso Aleatório (RAMs)

5. Um arranjo de memória estática com quatro linhas similar ao mostrado na Figura 10–9 tem armazenado inicialmente somente 0s. Qual é o conteúdo dele após as seguintes condições? Considere que um nível 1 seleciona a linha.
Linha 0 = 1, Entrada de dado (Bit 0) = 1
Linha 1 = 0, Entrada de dado (Bit 1) = 1
Linha 2 = 1, Entrada de dado (Bit 2) = 1
Linha 3 = 0, Entrada de dado (Bit 3) = 0

6. Desenhe um diagrama lógico básico para uma RAM estática de 512 8 bits, mostrando todas as entradas e saídas.

7. Considerando que uma SRAM de 64k × 8 tenha uma estrutura similar a da SRAM vista na Figura 10–11, determine o número de linhas e colunas de 8 bits no seu arranjo de células de memória.

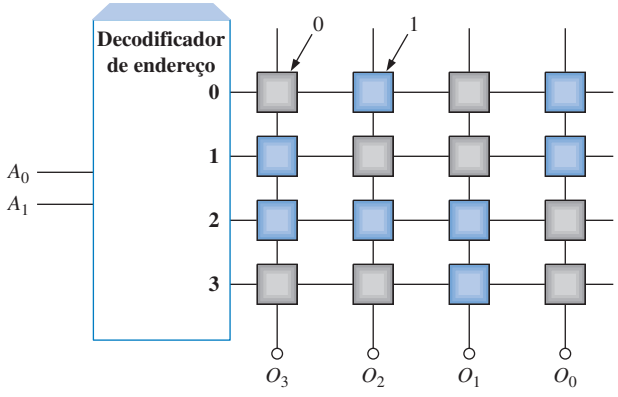
8. Redesenhe o diagrama em bloco mostrado na Figura 10–11 para uma memória de 64k × 8.

9. Explique a diferença entre uma SRAM e uma DRAM.

10. Qual é a capacidade de uma DRAM que tem vinte linhas de endereço?

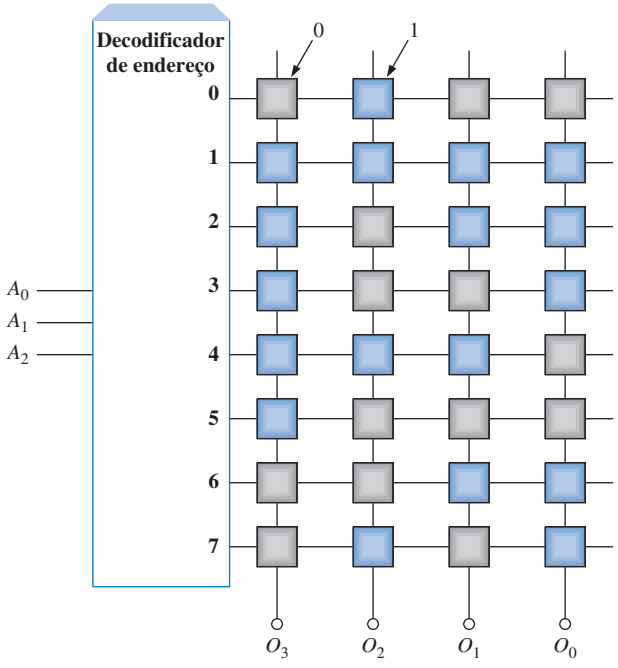
SEÇÃO 10–3 Memórias Apenas de Leitura (ROMs)

11. Para o arranjo de ROM dado na Figura 10–81, determine as saídas para todas as combinações de entrada possíveis e resuma as respostas na forma tabular (célula laranja é nível 1, célula cinza é nível 0).



► FIGURA 10–81

12. Determine a tabela-verdade para a ROM dada na Figura 10–82.



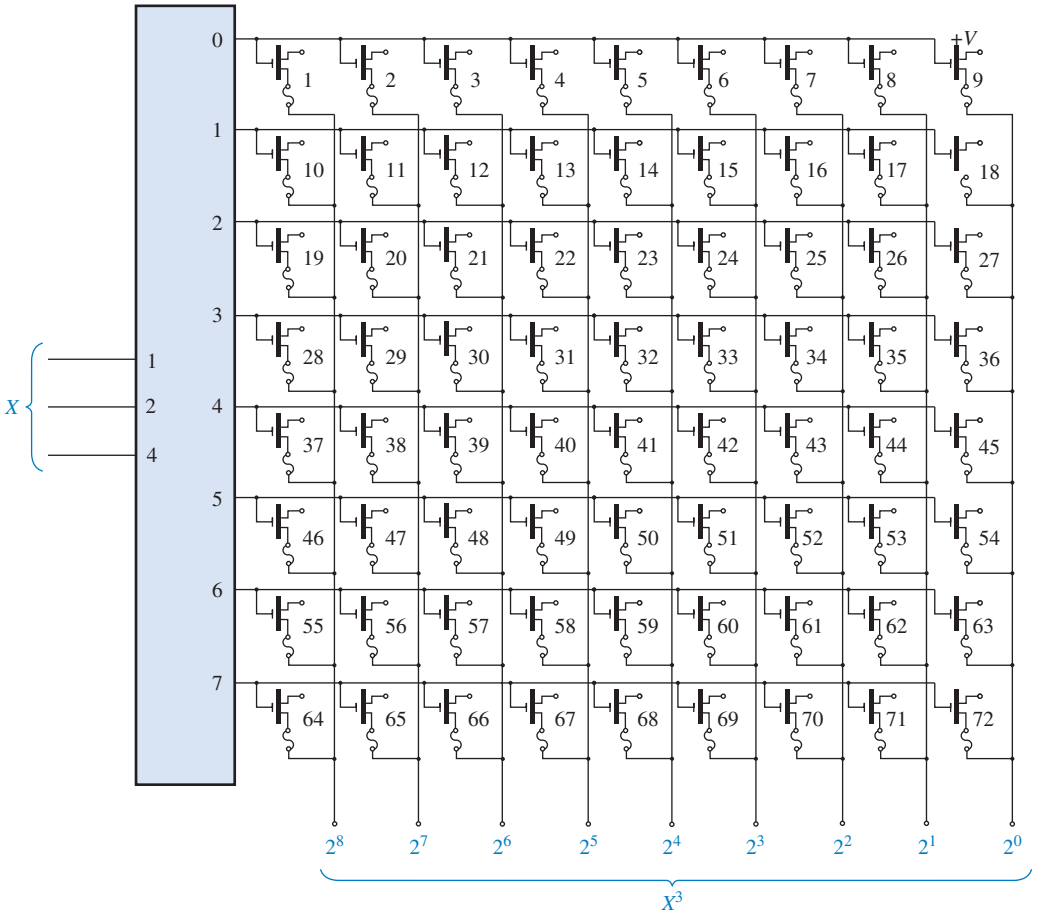
► FIGURA 10–82

13. Usando um procedimento similar ao do Exemplo 10–1, projete uma ROM para a conversão de um único dígito BCD para código de excesso 3.

14. Qual a capacidade total em *bits* de uma ROM que tem 14 linhas de endereço e 8 saídas de dados.

SEÇÃO 10–4 ROMs Programáveis (PROMs e EPROMs)

15. Considerando que a matriz de uma PROM mostrada na Figura 10–83 é programada "queimando" uma conexão fusível para criar um nível 0, indique as conexões a serem queimadas para programar uma tabela de busca com saída X^3, onde X é um número de 0 a 7.



▲ FIGURA 10–83

16. Determine os endereços que são programados e o conteúdo de cada endereço após a seqüência de programação dada na Figura 10–84 ser aplicada numa EPROM como a que é mostrada na Figura 10–31.

SEÇÃO 10–6 Expansão de Memória

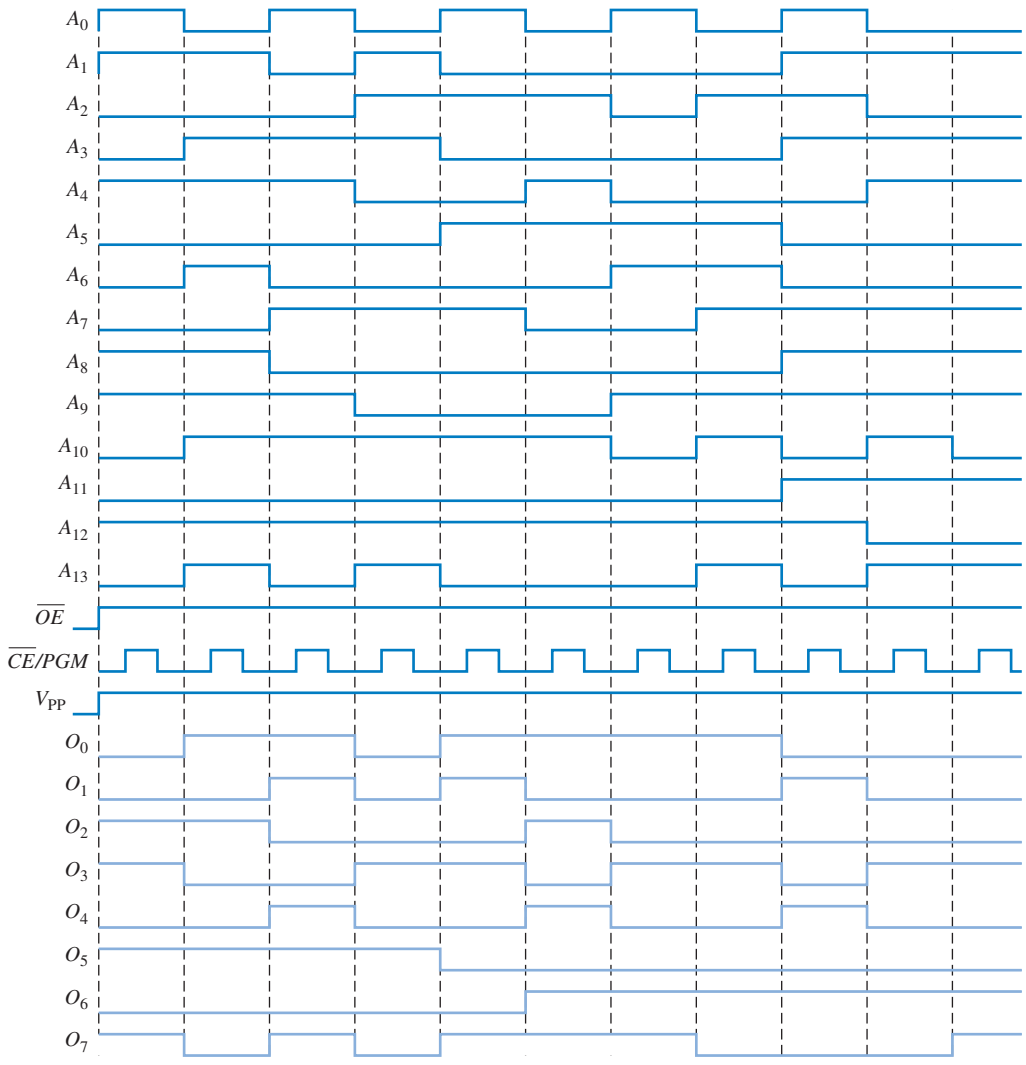
17. Use DRAMs de 16k × 4 para construir uma DRAM de 64k × 8. Mostre o diagrama lógico.

18. Usando um diagrama em bloco, mostre como uma RAM dinâmica de 64k × 1 pode ser expandida para construir uma RAM de 256k × 4.

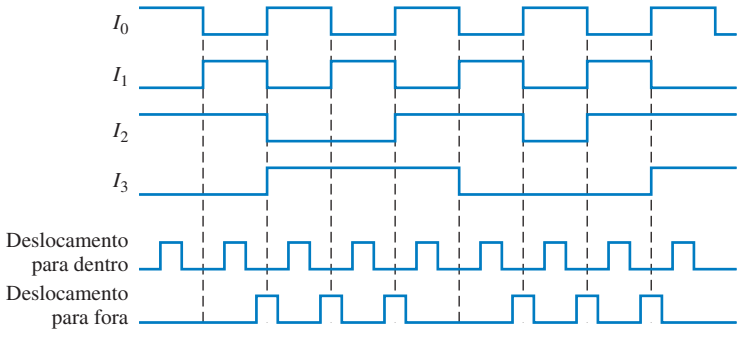
19. Qual é o tamanho da palavra e a capacidade de palavras da memória do Problema 17? E do Problema 18?

SEÇÃO 10–7 Tipos Especiais de Memórias

20. Complete o diagrama de temporização dado na Figura 10–85 mostrando as formas de onda de saída as quais estão inicialmente todas em nível BAIXO para uma memória serial FIFO como a que é mostrada na Figura 10–49.



▲ **FIGURA 10–84**



▶ **FIGURA 10–85**

21. Considere uma RAM de 4096 × 8 na qual os últimos 64 endereços são usados como uma pilha LIFO. Se o primeiro endereço na RAM for 000_{16}, determine os 64 endereços usados para a pilha.

22. Na memória do Problema 21, 16 bytes são armazenados na pilha. Em qual endereço está o primeiro byte? Em qual endereço está o último byte?

SEÇÃO 10–8 Armazenamento Magnético e Óptico

23. Descreva de forma geral a formatação de um disco rígido.

24. Explique o significado de tempo de busca e período de latência num drive de disco rígido.

25. Por que uma fita magnética requer um tempo de acesso muito maior que para um disco?

26. Explique as diferenças entre disco magneto-óptico, CD-ROM e WORM.

SEÇÃO 10–9 Análise de Defeito

27. Determine se os conteúdos da ROM vista na Figura 10–86 estão corretos.

ROM	
1 0 1 1 1	
1 1 1 1 0	
1 1 0 1 1	
1 0 1 1 0	
1 1 1 0 1	
1 1 1 0 0	
0 0 0 0 1	
Checksum	0 1 1 0 0

▶ **FIGURA 10–86**

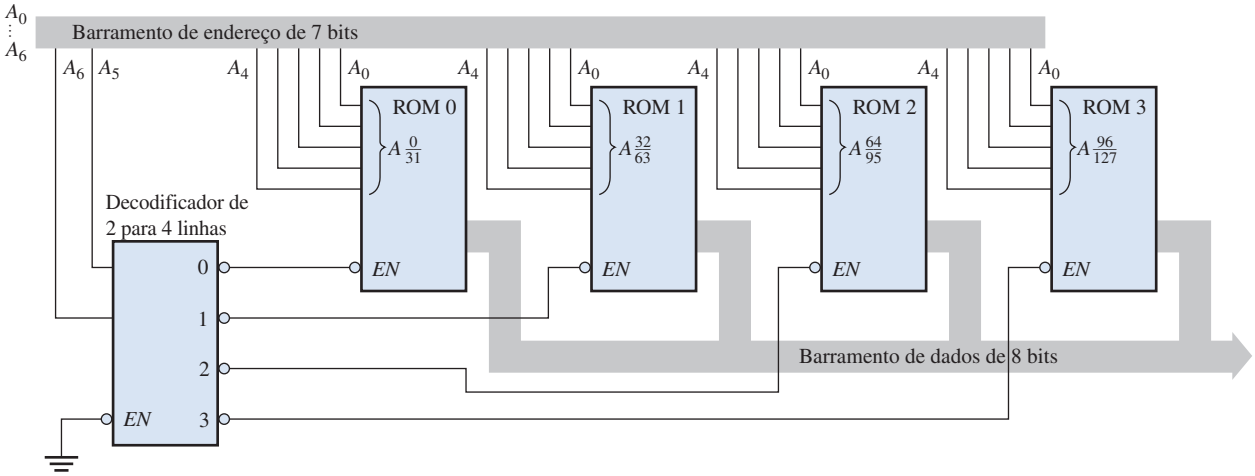
28. Uma ROM de 128 × 8 é implementada como mostra a Figura 10–87. O decodificador decodifica os dois bits de endereço mais significativos para habilitar as ROMs uma de cada vez, dependendo do endereço selecionado.

(a) Expresse em hexadecimal o menor e o maior endereço de cada ROM.

(b) Considere que um único *checksum* é usado para toda a memória e ele é armazenado no endereço mais alto. Desenvolva um fluxograma para o teste do sistema de memória completo.

(c) Considere que cada ROM tenha um *checksum* armazenado no seu endereço mais alto. Modifique o fluxograma desenvolvido na parte (b) da figura para acomodar essa alteração

(d) Qual é a desvantagem de usar um único *checksum* para toda a memória em vez de um *checksum* para cada ROM individual?



▲ **FIGURA 10–87**

29. Suponha que um teste *checksum* seja executado na memória mostrada na Figura 10–87 e cada ROM individual tenha um *checksum* no endereço mais alto. Qual é o CI ou os CIs que você substituiria para cada uma das seguintes mensagens de erro que aparece no monitor do sistema?

(a) ENDEREÇO 40-5F DEFEITO

(b) ENDEREÇO 20-3F DEFEITO

(c) ENDEREÇO 00-7F DEFEITO

Aplicações em Sistemas Digitais

30. Desenvolva um diagrama de temporização para o circuito lógico básico da memória visto na Figura 10–72 para ilustrar a inserção dos dígitos 4321 na SRAM. Inclua todas as entradas e saídas de cada dispositivo.

31. Na programação do sistema de segurança, qual o estado do contador mostrado na Figura 10–77 após os códigos de dois dígitos serem inseridos?

32. Qual a finalidade do circuito lógico da memória?

33. Discuta as vantagens e as desvantagens do uso de uma PROM externa ao CPLD em vez da memória no chip CPLD no circuito lógico da memória.

Problemas Especiais de Projeto

34. Modifique o projeto do circuito lógico da memória do sistema de segurança para acomodar um código de entrada de 5 dígitos.

35. Faça as modificações apropriadas para o circuito lógico do código de segurança do sistema de segurança para um código de entrada de 5 dígitos. Consulte o tópico de Aplicações em Sistemas Digitais do Capítulo 9.

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 10–1 Fundamentos de Memória Semicondutora

1. O bit é a menor unidade de dado.

2. 256 bytes é 2048 bits.

3. Uma operação de escrita armazena dados na memória.

4. Uma operação de leitura faz uma cópia dos dados da memória.

5. Uma unidade de dado é localizada pelo seu endereço.

6. Uma RAM é volátil e tem capacidade de leitura/escrita. Uma ROM e não-volátil e tem apenas a capacidade de leitura.

SEÇÃO 10–2 Memórias de Acesso Aleatório (RAMs)

1. Assíncronas e síncronas com característica de rajada.

2. Uma pequena memória rápida entre a CPU e a memória principal.

3. As SRAMs têm latches como células de armazenamento que podem reter dados indefinidamente enquanto a alimentação estiver presente. As DRAMs têm células com armazenamento capacitivo que têm que ser reavivadas periodicamente.

4. A operação de refresh evita que o dado seja perdido devido à descarga do capacitor. O bit armazenado é reforçado periodicamente recarregando o capacitor para o seu nível nominal.

5. FPM, EDO, BEDO, Síncrona

SEÇÃO 10–3 Memórias Apenas de Leitura (ROMs)

1. 512×8 é igual a 4096 bits.

2. ROM de máscara, PROM, EPROM, UV EPROM, EEPROM

3. Oito bits de endereço são necessários para 256 posições de bytes ($2^8 = 256$).

SEÇÃO 10–4 ROMs Programáveis (PROMs e EPROMs)

1. As PROMs são programáveis por campo; as ROMs não.

2. Após o apagamento de uma EPROM típica restam somente 1s.

3. A leitura é o modo normal de operação para uma PROM.

SEÇÃO 10–5 Memórias Flash

1. Flash, ROM, EPROM e EEPROM não são voláteis.

2. A flash é não-volátil; a SRAM e a DRAM são voláteis.

3. Programação, leitura e apagamento

SEÇÃO 10–6 Expansão de Memória

1. Oito RAMs
2. Oito RAMs
3. SIMM; Módulo de memória em linha única
4. DIMM; Módulo de memória em linha dupla
5. RIMM; Módulo de memória em linha rambus

SEÇÃO 10–7 Tipos Especiais de Memórias

1. Em uma memória FIFO, o primeiro bit (ou palavra) a entrar é o primeiro bit (ou palavra) a sair.
2. Em uma memória LIFO, o último bit (ou palavra) a entrar é o primeiro bit (ou palavra) a sair. Uma pilha é uma LIFO.
3. A operação ou instrução que acrescenta dados na memória pilha.
4. A operação ou instrução que remove dados da memória pilha.
5. CCD é um dispositivo acoplado por carga

SEÇÃO 10–8 Armazenamento Magnético e Óptico

1. Armazenamento magnético: disquete, disco rígido, fita e disco magneto-óptico.
2. Capacidade de armazenamento do disquete: 1,44 MB
3. Um disco magnético é organizado em trilhas e setores
4. Um disco magneto-óptico usa um feixe laser e um eletroímã.
5. Armazenamento óptico: CD-ROM, CD-R, CD-RW, DVD-ROM, WORM

SEÇÃO 10–9 Análise de Defeito

1. Os conteúdos da ROM são somados e o valor é comparado com o *checksum* previamente armazenado.
2. O *checksum* não pode ser usado porque os conteúdos de uma RAM não são fixos.
3. (1) um curto-circuito entre células adjacentes; (2) uma inabilidade de algumas células de armazenar 1s e 0s; (3) alteração dinâmica dos conteúdos de um endereço quando os conteúdos de outro endereço muda.

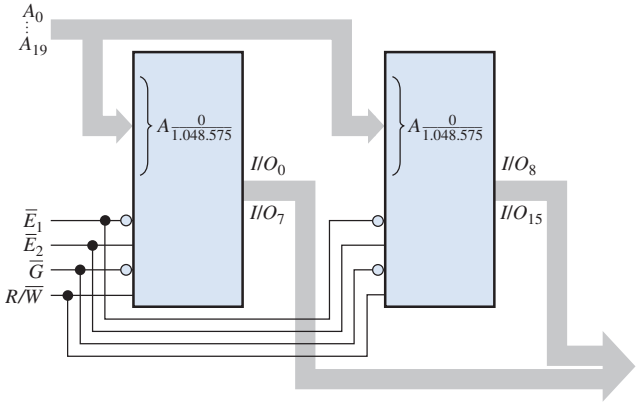
PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

10–1. $G_3G_2G_1G_0 = 1110$

10–2. Conecte oito ROMs de 64k × 1 em paralelo para formar uma ROM de 64k × 8

10–3. Dezesseis ROMs de 64k × 1

10–4. Veja a Figura 10–88.



▶ FIGURA 10–88

10–5. ROM 1: 0 a 524.287; ROM 2: 524.288 a 1.048.575

AUTOTESTE

1. (b) **2.** (c) **3.** (c) **4.** (d) **5.** (a) **6.** (d)
7. (c) **8.** (a) **9.** (b) **10.** (f) **11.** (c) **12.** (d)

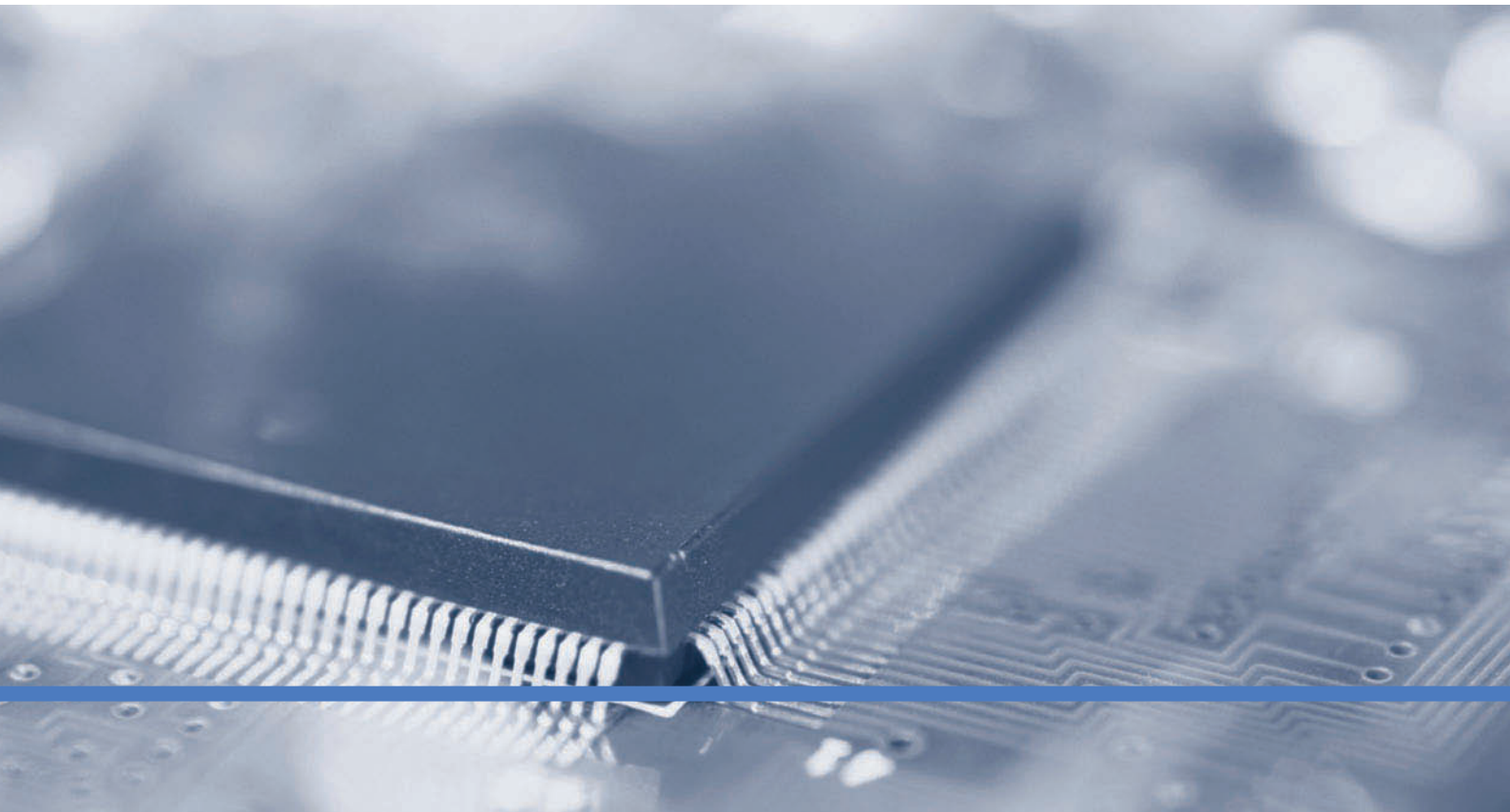
11

LÓGICA PROGRAMÁVEL E SOFTWARE

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Discutir os tipos de dispositivos de lógica programável (SPLDs e CPLDs) e explicar a estrutura básica deles
- Descrever a arquitetura básica dos dois tipos de SPLDs (PAL e GAL)
- Descrever a arquitetura da família Altera MAX 7000 de CPLDs
- Descrever a arquitetura do CPLD Altera MAX II
- Explicar a estrutura básica de um arranjo lógico programável (PLA)
- Descrever a arquitetura da família de CPLDs Xilinx CoolRunner II



- Discutir a operação de macrocélulas
- Fazer distinção entre CPLDs e FPGAs
- Explicar a operação básica de uma tabela de busca (LUT – *look-up table*)
- Definir *propriedade intelectual* e *plataforma FPGA*
- Descrever a arquitetura da família FPGA Altera Stratix
- Discutir funções embutidas
- Descrever a arquitetura da família FPGA Xilinx Virtex
- Mostrar o fluxo do projeto de um software básico para um dispositivo programável
- Explicar os elementos do fluxo de um projeto: inserção do projeto, simulação funcional, síntese, implementação, simulação de temporização e *download* (transferência)
- Discutir os diversos métodos de teste de um dispositivo lógico programável, introduzindo a lógica *boundary scan*

TERMOS IMPORTANTES

- PAL
- GAL
- Macrocélula
- Registrado
- CPLD
- LAB
- LUT
- FPGA
- CLB
- Propriedade intelectual
- Fluxo de projeto
- Dispositivo destino
- Inserção via diagrama
- Inserção via texto
- Compilador
- *Download*
- *Bed-of-nails*
- *Flying probe*
- *Boundary scan*
- Primitivo
- Ferramenta ajustador
- Simulação funcional
- Simulação de temporização

INTRODUÇÃO

Nesse capítulo, discutiremos a arquitetura básica (estrutura e organização internas) de SPLDs, CPLDs e FPGAs. Diversas especificações de CPLDs são apresentadas, incluindo as famílias Altera MAX 7000, MAX II e a Xilinx CoolRunner II. As FPGAs apresentadas são Altera Stratix e Xlinx Virtex.

Uma discussão das ferramentas de desenvolvimento de software aborda o fluxo de um projeto genérico para programação de um dispositivo incluindo inserção do projeto, simulação funcional, síntese, implementação, simulação de temporização e *download* (transferência). Além disso, existe uma seção de análise de defeito no circuito de uma placa uma vez que o dispositivo esteja operando. Dentre os métodos de testes se incluem *bed-of-nails*, *flying probe* e *boundary scan*.

DISCUSSÃO PRÉVIA DE APLICAÇÕES EM SISTEMAS DIGITAIS

O tópico de Aplicações em Sistemas Digitais ilustra a seqüência do desenvolvimento de um circuito lógico genérico para acionamento de um display de 7 segmentos. A lógica para cada segmento foi desenvolvida na seção de Aplicações em Sistemas Digitais no Capítulo 4 sendo que um programa VHDL foi escrito para cada lógica. Podemos inserir os programas VHDL usando uma ferramenta de software de inserção via texto. Entretanto, ilustraremos a seqüência do projeto com uma abordagem de inserção via esquemático.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

11-1 LÓGICA PROGRAMÁVEL: SPLDs E CPLDs

Os dois tipos principais de dispositivos lógicos programáveis simples (SPLDs) são o PAL e o GAL. *PAL* significa lógica de arranjo programável e *GAL* significa lógica de arranjo genérico. Geralmente, o dispositivo PAL é programável uma vez (OTP – *one-time programmable*) e um dispositivo GAL é um tipo de PAL reprogramável; entretanto, como alguns dispositivos programáveis SPLDs ainda são chamados de PALs, a linha entre PALs e GALs não é bem definida. O termo GAL é uma designação original usada pela Lattice Semiconductor e posteriormente licenciada para outros fabricantes. A estrutura básica de PALs e GALs é um arranjo AND programável e um arranjo OR fixo, que é uma arquitetura básica de soma-de-produtos. O dispositivo lógico programável complexo (CPLD) é basicamente um único dispositivo com múltiplos SPLDs que proporcionam maior capacidade para projeto de circuitos lógicos maiores.

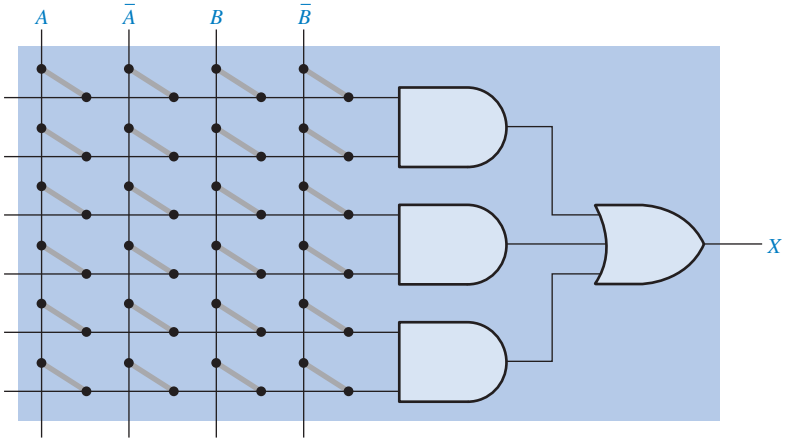
Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a operação de um SPLD ■ Mostrar como uma expressão de soma-de-produtos é implementada num dispositivo PAL ou GAL ■ Explicar diagramas lógicos simplificados de PAL/GAL ■ Descrever a macrocélula básica de um PAL/GAL ■ Discutir os dispositivos PAL16V8 e FAL22V10 ■ Descrever um CPLD básico

SPLD: O Dispositivo PAL

Um dispositivo **PAL** (*programmable array logic*) consiste em um arranjo programável de portas AND que se conecta a um arranjo fixo de portas OR. Geralmente, os dispositivos PALs são implementados com a tecnologia de conexão a fusível e, portanto, são programáveis uma vez (OTP).

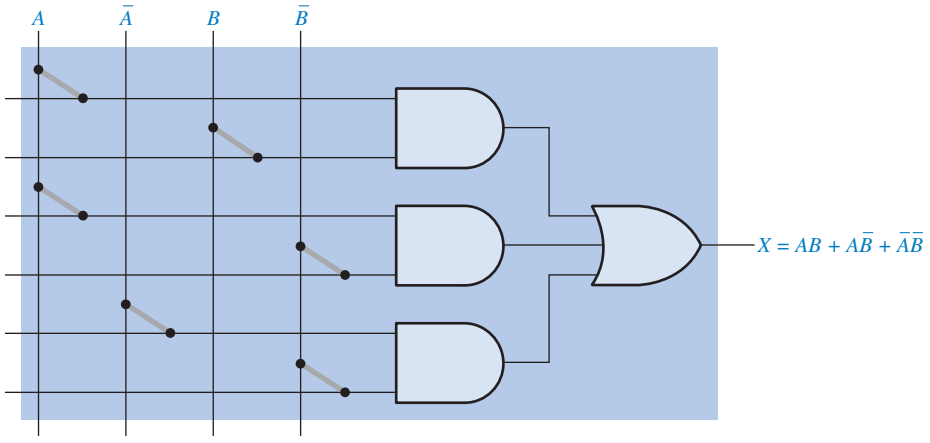
A estrutura de um dispositivo PAL permite que qualquer expressão lógica de soma-de-produtos com um número definido de variáveis seja implementada. Conforme já estudamos, qualquer função lógica combinacional pode ser expressa na forma de soma-de-produtos. Uma estrutura simples de um dispositivo PAL é mostrada na Figura 11–1 para duas variáveis de entrada e uma saída; a maioria dos dispositivos PALs tem muitas entradas e muitas saídas. Conforme estudamos no Capítulo 3, um arranjo programável e essencialmente uma grade ou matriz de condutores que formam linhas e colunas com uma conexão programável em cada ponto de cruzamento. Cada conexão programável, que no caso de um dispositivo PAL é um fusível, é denominada de *célula*. Cada linha é conectada na entrada de uma porta AND e cada coluna é conectada a uma variável de entrada ou o seu complemento. Programando a presença ou ausência de uma conexão a fusível, qualquer combinação das variáveis de entrada ou os complementos pode ser aplicada a uma porta AND para formar qualquer termo-produto desejado. As portas AND são conectadas a uma porta OR, criando uma saída de soma-de-produtos.



► **FIGURA 11–1**
Estrutura AND/OR básica de um dispositivo PAL.

Implementação de uma Expressão de Soma-de-Produtos Um exemplo de um dispositivo PAL simples é programado como mostra a Figura 11–2 de forma que o termo-produto AB é gerado pela porta AND superior, $A\overline{B}$ é gerado pela porta AND do meio e $\overline{A}\,\overline{B}$ é gerado pela porta AND inferior. Como podemos ver, os fusíveis são deixados intactos para conectar as variáveis desejadas ou seus complementos nas entradas das portas AND apropriadas. Os fusíveis são abertos onde uma variável ou o seu complemento não é usada num determinado termo-produto. A saída final da porta OR é uma expressão na forma de soma-de-produtos.

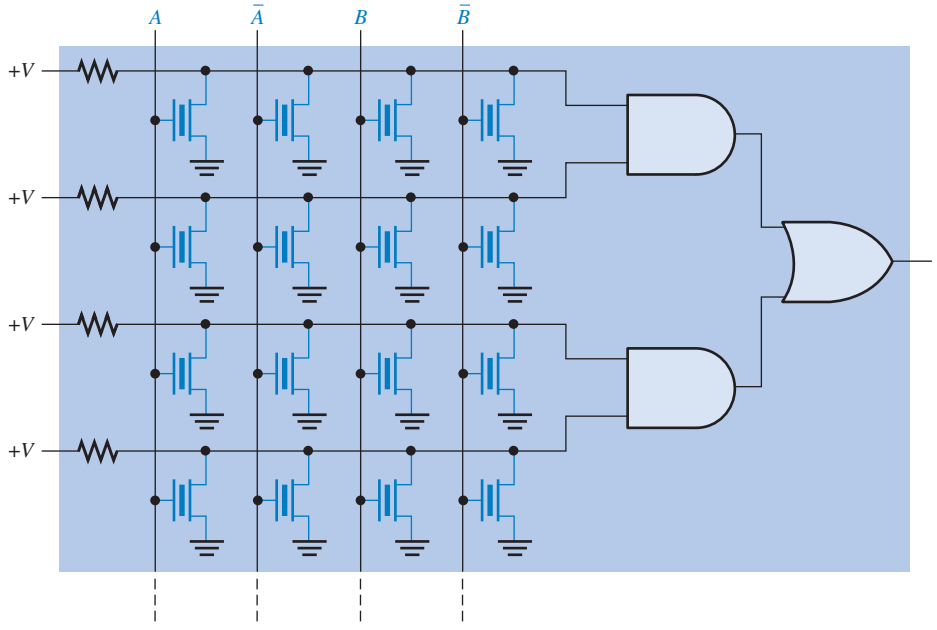
$$X = AB + A\overline{B} + \overline{A}\,\overline{B}$$



◀ **FIGURA 11–2**
Implementação de uma expressão na forma de soma-de-produtos usando um dispositivo PAL.

SPLD: O Dispositivo GAL

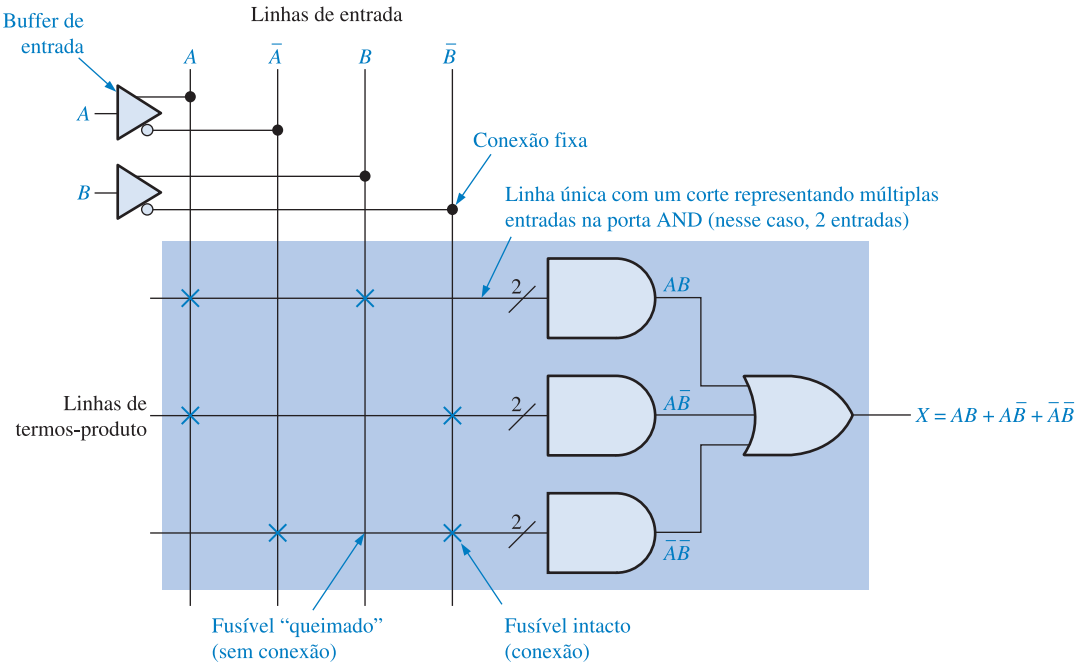
O dispositivo **GAL** é essencialmente um dispositivo PAL que pode ser reprogramado. Ele tem o mesmo tipo de organização AND/OR que o dispositivo PAL. A diferença básica é que um dispositivo GAL usa uma tecnologia de processo reprogramável, como uma EEPROM (E^2CMOS), em vez de fusíveis, como mostra a Figura 11–3.



◀ **FIGURA 11–3**
Arranjo simplificado de um dispositivo GAL.

Notação Simplificada para Diagramas PAL/GAL

Os atuais dispositivos PAL e GAL têm muitas portas AND e OR além de outros elementos e são capazes de manipular muitas variáveis e seus complementos. A maioria dos diagramas de PAL e GAL que podemos ver nas folhas de dados usa uma notação simplificada, como a ilustrada na Figura 11–4, para fazer com que o diagrama esquemático fique menos complicado.



▲ **FIGURA 11–4**

Parte de um dispositivo PAL/GAL programado.

As variáveis de entrada para um dispositivo PAL ou GAL têm geralmente buffers para evitar carregamento por um grande número de entradas de portas AND nas quais elas são conectadas. O símbolo do triângulo no diagrama representa um buffer que gera a variável e o seu complemento. As conexões fixas das variáveis de entrada e buffers são mostrados usando notação de ponto padrão.

Os dispositivos PALs e GALs têm um grande número de linhas de interconexões programáveis, sendo que cada porta AND tem múltiplas entradas. Os diagramas lógicos típicos de dispositivos PAL e GAL representam uma porta AND de múltiplas entradas com o símbolo da porta AND tendo uma única linha de entrada com um corte e um dígito representando o número real de entradas. A Figura 11–4 ilustra isso para o caso de portas AND de 2 entradas.

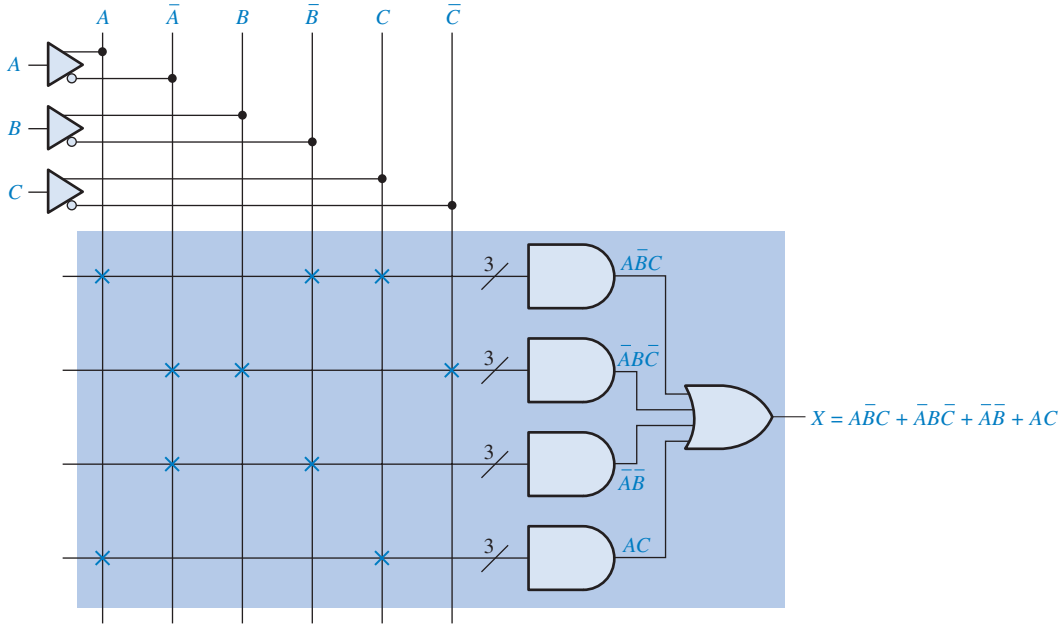
As conexões programáveis num arranjo são indicadas no diagrama por um X na cor laranja no ponto de cruzamento para um fusível intacto ou outro tipo de conexão e a ausência de um X para um fusível aberto ou outro tipo de conexão. A Figura 11–4 mostra a função lógica $AB + A\overline{B} + \overline{A}\,\overline{B}$ programada.

EXEMPLO 11–1

Mostre como um dispositivo PAL é programado para a seguinte função lógica de 3 variáveis:

$$X = A\overline{B}C + \overline{A}B\overline{C} + \overline{A}\,\overline{B} + AC$$

Solução O arranjo programado é mostrado na Figura 11–5. As conexões a fusível intactas são indicadas por pequenas marcas na forma de X em laranja. A ausência de um X significa que o fusível está aberto.



▲ **FIGURA 11–5**

Problema relacionado* Escreva a expressão para a saída se as conexões a fusível que conectam a entrada *A* na linha superior e na linha inferior na Figura 11–5 também estiverem abertas.

* As respostas estão no final do capítulo.

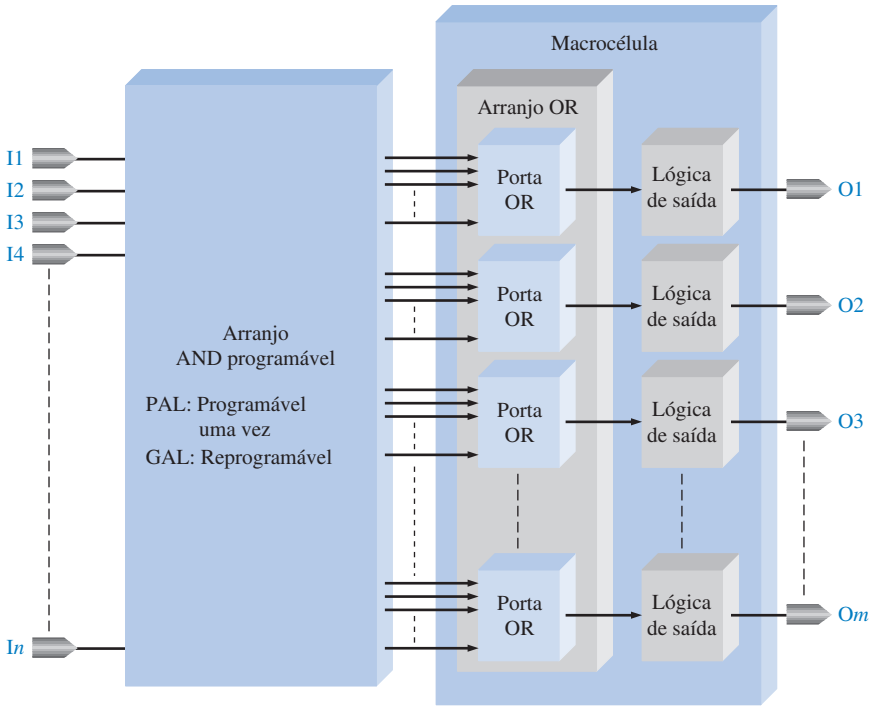
Diagrama em Bloco Geral de Dispositivos PAL/GAL

Um diagrama em bloco de um dispositivo PAL ou GAL é mostrado na Figura 11–6. Lembre-se, a diferença básica é que um dispositivo GAL tem um arranjo reprogramável e um dispositivo PAL é programável apenas uma vez. As saídas do arranjo AND programável vão para as portas OR fixas que são conectadas à lógica de saída adicional. Uma porta OR combinada com a sua lógica de saída associada é tipicamente denominada *macrocélula*. A complexidade da macrocélula depende do dispositivo em particular, sendo sempre reprogramável nos dispositivos GAL.

Macrocélula

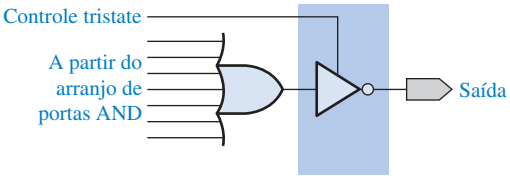
Uma **macrocélula** consiste geralmente em uma porta OR e alguma lógica de saída associada. As macrocélulas variam em complexidade dependendo do tipo particular de PAL ou GAL. Uma macrocélula pode ser configurada para lógica combinacional, lógica registrada ou uma combinação de ambos. A lógica **registrada** significa que existe um flip-flop na macrocélula para prover uma função lógica seqüencial. A operação de macrocélulas registradas é abordada na Seção 11–4.

A Figura 11–7 ilustra três tipos básicos de macrocélulas com lógica combinacional. A parte (a) da figura mostra uma macrocélula simples com porta OR e um inversor com um controle tristate que pode fazer com que o inversor se assemelhe a um circuito aberto para desconectar completamente a saída. A saída do inversor tristate pode ser nível ALTO, nível BAIXO ou desconectado (alta impedância). A parte (b) da figura é uma macrocélula que pode ser entrada ou saída. Quando a saída é usada como uma entrada, o inversor tristate é desconectado e a entrada

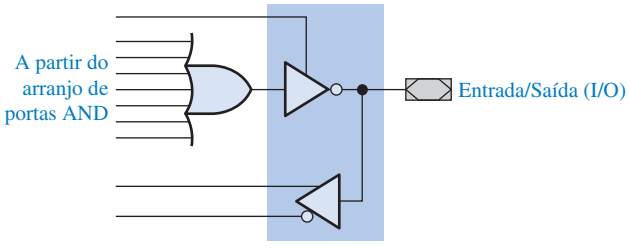


▶ **FIGURA 11–6**
Diagrama em bloco geral de um dispositivo PAL ou GAL.

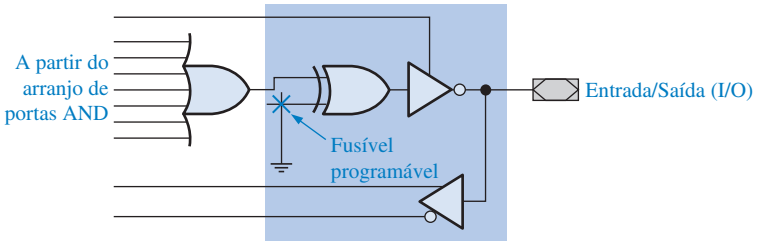
vai para o buffer que está conectado no arranjo AND. A parte (c) da figura é uma macrocélula que pode ser programada para ter uma saída de estado ativo ALTO ou estado ativo BAIXO, ou ela pode ser usada como uma entrada. Uma entrada para uma porta EX-OR pode ser programada para ser nível ALTO ou BAIXO. Quando a entrada da EX-OR programável for nível ALTO,



(a) Saída combinacional (ativa em nível BAIXO). Uma saída ativa em nível ALTO seria mostrada sem o pequeno círculo no símbolo da porta tristate.



(b) Entrada/Saída combinacional (ativa em nível BAIXO)



(c) Saída com polaridade programável

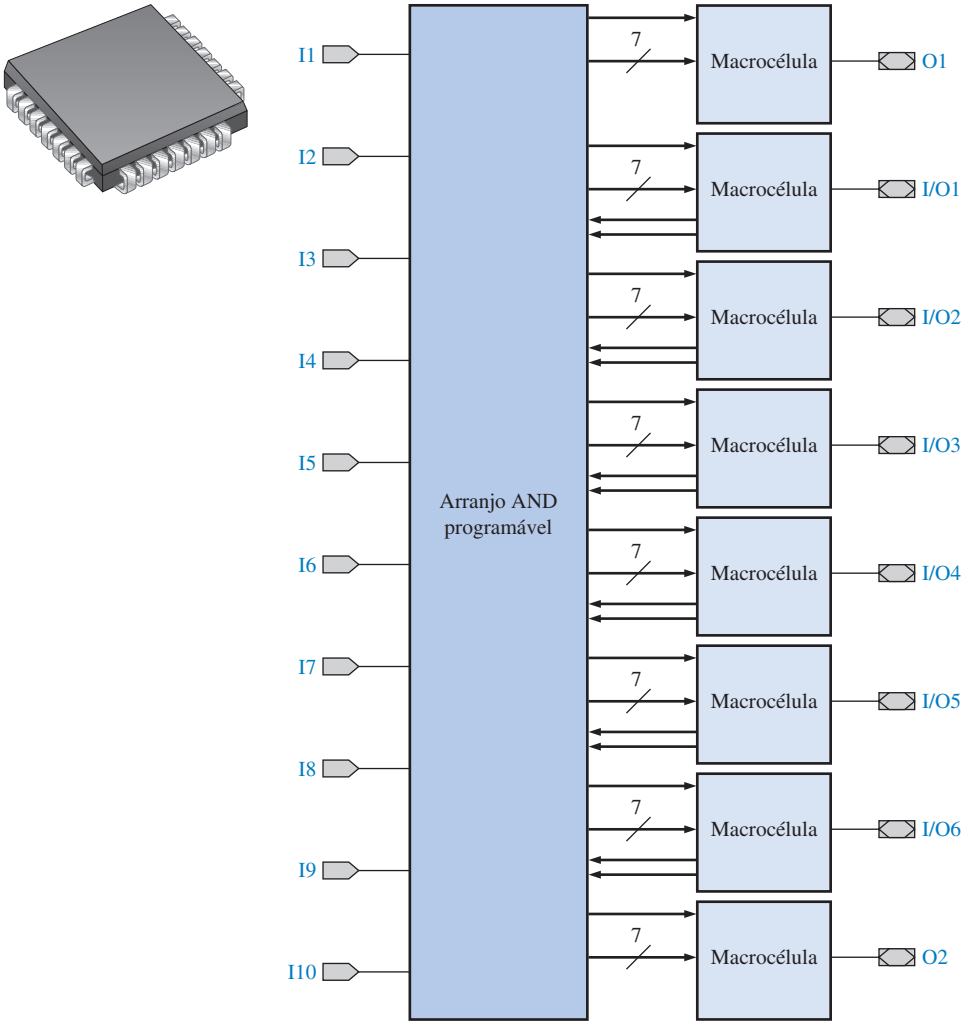
▲ **FIGURA 11–7**
Tipos básicos de macrocélulas PAL/GAL para lógica combinacional.

a saída da porta OR é invertida porque $0 \oplus 1 = 1$ e $1 \oplus 1 = 0$. De forma similar, quando a entrada da porta EX-OR programável for nível BAIXO, a saída da porta OR não é invertida porque $0 \oplus 0 = 0$ e $0 \oplus 0 = 1$.

SPLDs Específicos

Geralmente, as configurações de encapsulamento SPLD variam de 20 a 28 pinos. Dois fatores que podemos usar para ajudar a determinar se um determinado dispositivo PAL ou GAL é adequado para um determinado projeto lógico são o número de portas equivalentes ou a densidade. Outros parâmetros a considerar são a freqüência máxima de operação, os tempos de atraso e a tensão de alimentação. Lattice, Actel, Atmel e Cypress são algumas das diversas empresas que produzem SPLDs. Diversos fabricantes de SPLDs podem ter diferentes formas de definir densidade, assim temos que usar o número especificado de portas equivalentes com isso em mente.

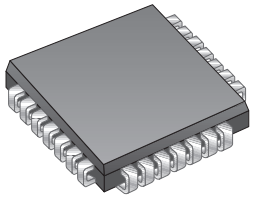
Os dispositivos 16V8 e 22V10 são tipos comuns de PALs e GALs. A designação dos dispositivos indicam o número de entradas, o número de saídas e o tipo de lógica de saída. Por exemplo, 16V8 significa que o dispositivo tem dezesseis entradas, oito saídas e as saídas são variáveis (V). A letra *L* ou *H* significa que a saída é ativa em nível BAIXO ou ativa em nível ALTO, respectivamente. O diagrama em bloco para um dispositivo PAL 16V8 e um tipo de encapsulamento SPLD são mostrados na Figura 11–8. Cada macrocélula tem oito entradas a partir do arranjo de portas

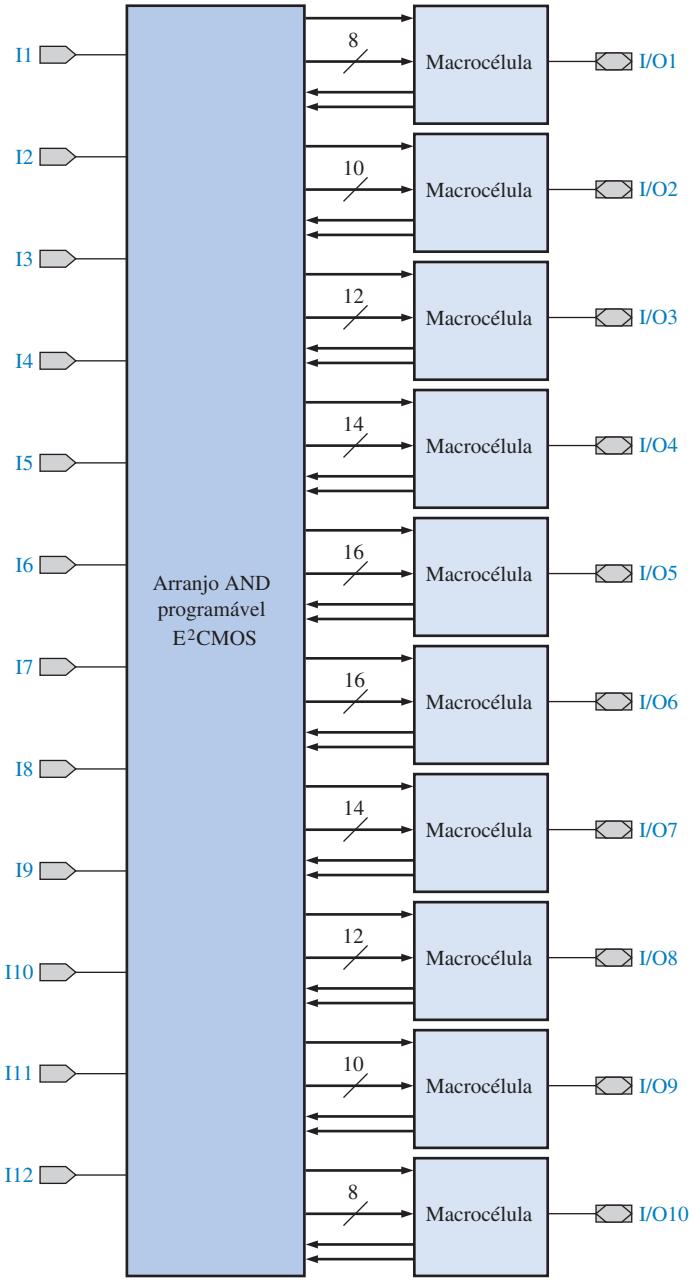


◀ **FIGURA 11–8**
Diagrama em bloco lógico de um dispositivo PAL16V8 e um encapsulamento SPLD típico.

AND, assim podemos ter até oito termos-produto para cada saída. Existem dez entradas dedicadas (I), duas saídas dedicadas (O) e seis pinos que podem ser usados como entradas ou saídas (I/O). Cada saída é ativa em nível BAIXO. O dispositivo PAL16V8 tem uma densidade de aproximadamente 300 portas equivalentes.

A Figura 11–9 mostra um diagrama em bloco para um dispositivo GAL22V10 e um encapsulamento SPLD típico. Esse dispositivo tem doze entradas dedicadas e dez pinos que podem ser entradas ou saídas. As macrocélulas têm entradas a partir do arranjo AND que varia de oito a dezesseis, conforme indicado pela notação simplificada. O dispositivo GAL22V10 tem uma densidade de aproximadamente 500 portas equivalentes.



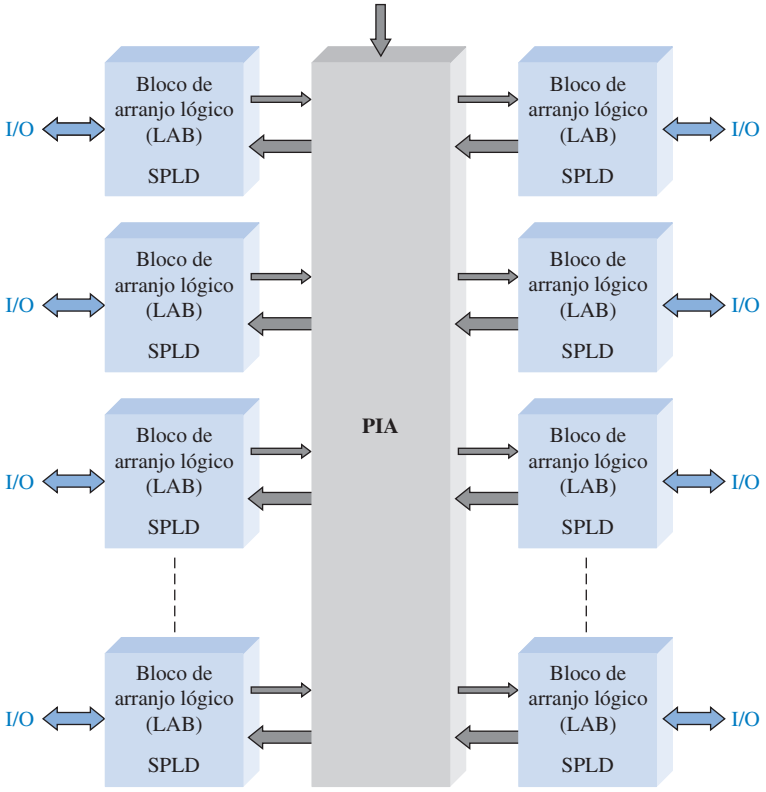


▲ **FIGURA 11–9**

Diagrama em bloco do dispositivo GAL22V10 e um encapsulamento SPLD típico.

CPLD

Um dispositivo lógico programável complexo – **CPLD** (*complex programmable logic device*) – consiste basicamente em múltiplos arranjos SPLDs com interconexões programáveis, conforme ilustrado na Figura 11–10. Embora a forma com que as CPLDs são organizadas internamente varie com o fabricante, a Figura 11–10 representa um CPLD genérico. Referimos-nos a cada arranjo SPLD num CPLD como um **LAB** (*logic array block* – bloco de arranjo lógico). Algumas vezes são usadas outras designações, como *bloco funcional*, *bloco lógico* ou *bloco genérico*. As interconexões programáveis são geralmente denominadas PIA (*programmable interconnect array* – arranjo de interconexões programáveis) embora alguns fabricantes, tal como Xilinx, usem o termo AIM (*advanced interconnect matrix* – matriz de interconexões avançadas) ou uma designação similar. Os LABs e as interconexões entre LABs são programados por software. Um CPLD pode ser programado para funções lógicas complexas baseadas em estruturas de soma-de-produtos de LABs individuais (na realidade SPLDs). Entradas podem ser conectadas a qualquer um dos LABs, sendo que suas saídas podem ser interconectadas a quaisquer outros LABs via PIA.



◀ **FIGURA 11–10**
Diagrama em bloco básico de um CPLD genérico.

A maioria dos fabricantes de dispositivos lógicos programáveis produz uma série de CPLDs que variam em densidade, tecnologia de processo, consumo de potência, tensão de alimentação e velocidade. Os fabricantes geralmente especificam a densidade de um CPLD em termos de macrocélulas ou blocos de arranjo lógico. As densidades podem variar de dezenas de macrocélulas a valores além de 2000 macrocélulas em encapsulamentos com até algumas centenas de pinos. À medida que as PLDs se tornam mais complexas, as densidades máximas aumentam. A maioria das CPLDs são reprogramáveis e usam tecnologia de processo de EEPROM ou SRAM para as conexões programáveis. O consumo de potência pode variar de alguns miliwatts a algumas centenas de miliwatts. As tensões de alimentação CC são tipicamente de 2,5 V a 5 V, dependendo do dispositivo.

Alguns fabricantes, (por exemplo, Altera, Xilinx, Lattice e Cypress) produzem CPLDs. Nesse capítulo, abordaremos os produtos da Altera e da Xilinx porque elas são as duas principais empresas do mercado. Outras empresas oferecem dispositivos e softwares similares, podemos facil-

mente fazer uma transição para outros produtos assim que estivermos familiarizados com um desses dois. Conforme estudaremos, os CPLDs e outros dispositivos lógicos programáveis são na realidade uma combinação de hardware e software.

SEÇÃO 11–1 REVISÃO

As respostas estão no final do capítulo.

1. O que quer dizer PAL?
2. O que quer dizer GAL?
3. Qual é a diferença entre PAL e GAL?
4. Basicamente, o que contém uma macrocélula?
5. O que é um CPLD?

11-2 CPLDs ALTERA

A empresa Altera produz algumas famílias de CPLDs, incluindo a família MAX II, MAX 3000 e MAX 7000. Nesta seção, o foco é principalmente na família MAX 7000 para ilustrar os conceitos da arquitetura CPLD tradicional, tendo em mente que as outras séries podem variar um pouco na arquitetura e/ou em parâmetros, como densidade, tecnologia de processo, consumo de potência, tensão de alimentação e velocidade.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever um típico CPLD da família MAX ■ Discutir a arquitetura básica de CPLDs MAX 7000 e MAX II ■ Explicar como termos-produto são gerados em CPLDs.

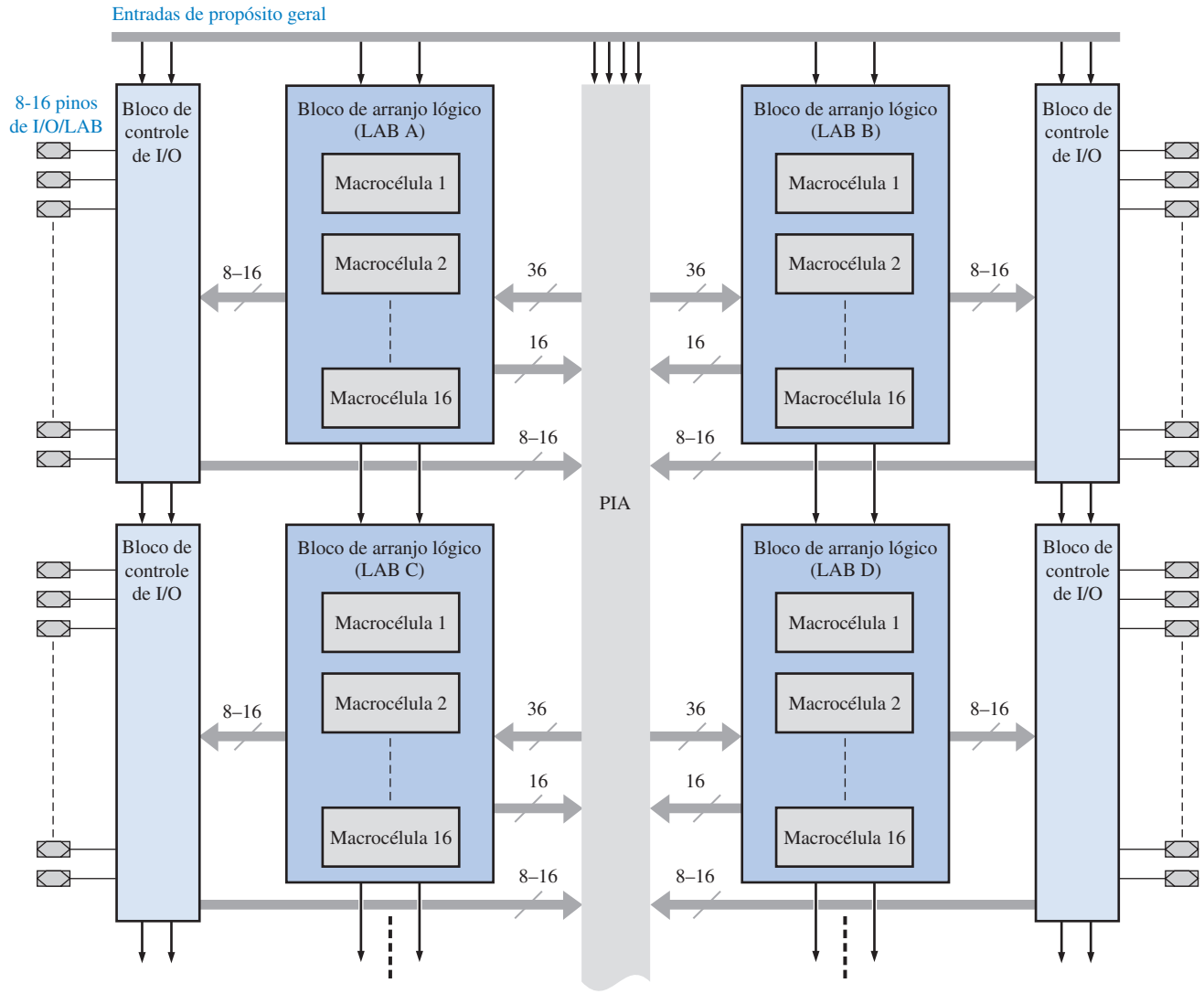
CPLD MAX 7000

A **arquitetura** de um CPLD é a forma na qual os elementos internos são organizados e arranjados. A arquitetura da família MAX 7000 é similar ao diagrama em bloco de um CPLD genérico (mostrado na Figura 11–10). Ele tem a estrutura de um PAL/GAL clássico que produz funções de soma-de-produtos. A densidade varia de 2 LABs a 16 LABs, dependendo do dispositivo da série. Lembre-se, um LAB é aproximadamente equivalente a um SPLD, sendo que os tamanhos de encapsulamentos variam de 44 pinos a 208 pinos. A série MAX 7000 de CPLDs usa a tecnologia de processo baseada em EEPROM. As versões ISP (*in-system programmable*) usam a interface padrão JTAG.

A Figura 11–11 mostra um diagrama em bloco geral do CPLD série MAX 7000 da Altera. Quatro LABs são mostrados, mas podem ser até seis, dependendo do dispositivo da série. Cada um dos quatro LABs consiste de dezesseis macrocélulas, sendo que múltiplos LABs são interconectados via PIA, que é uma estrutura de barramento (vai para todos os LABs) global programável no qual as entradas de propósito geral, os I/Os e as macrocélulas são conectados.

A Macrocélula Um diagrama simplificado de uma macrocélula da série MAX 7000 é mostrado na Figura 11–12. A macrocélula contém um pequeno arranjo AND programável com cinco portas AND, uma porta OR, uma matriz de seleção de termos-produto para conectar as saídas das portas AND à porta OR e á lógica associada que pode ser programada para entrada, saída lógica combinacional ou saída registrada. Essa macrocélula é abordada com mais detalhes na Seção 11–4.

Embora baseada no mesmo conceito, essa macrocélula difere um pouco da macrocélula discutida na Seção 11–1 em relação aos SPLDs porque ela contém uma parte de arranjo AND programável e uma matriz de seleção de termos-produto. Conforme mostra a Figura 11–12, cinco portas AND alimentam os termos-produto a partir do arranjo de interconexões programáveis

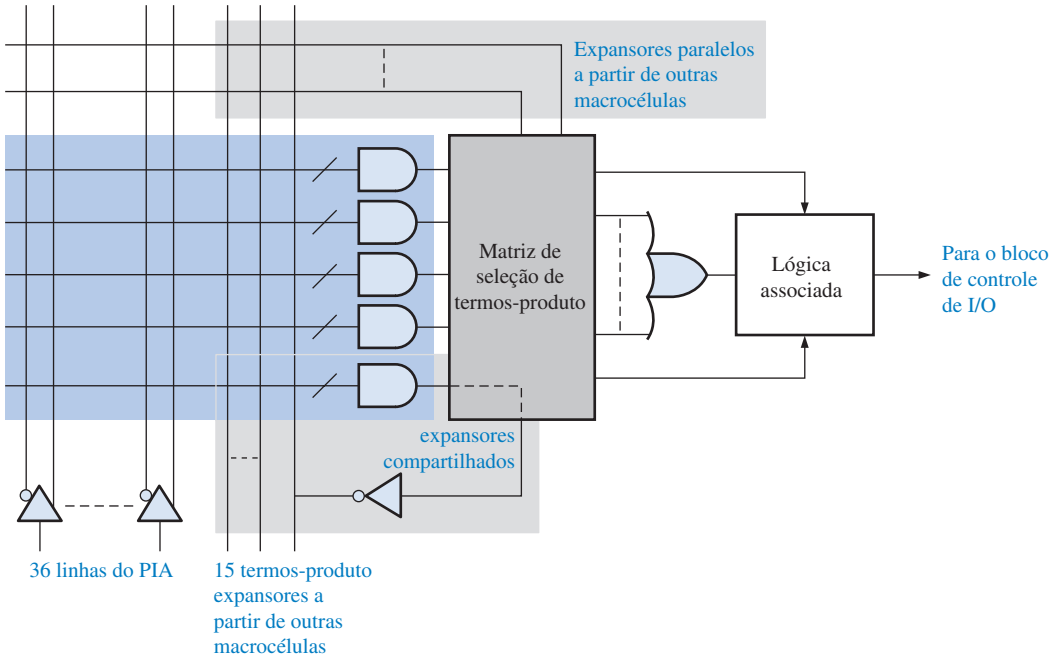


▲ **FIGURA 11–11**

Diagrama em bloco básico da CPLD série MAX 7000 do fabricante Altera.

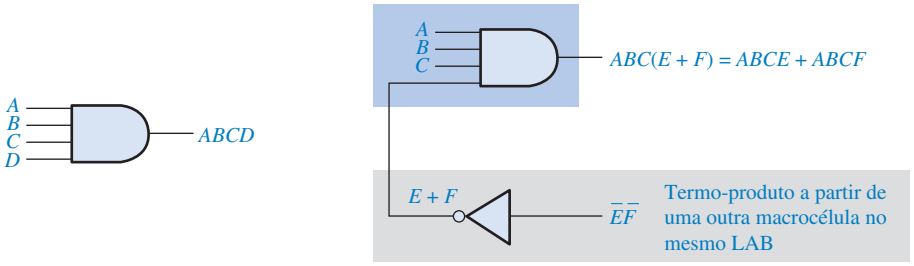
(PIA) para a matriz de seleção de termos-produto. O termo-produto da porta AND inferior pode ser realimentado de forma invertida no arranjo programável como um expansor compartilhado para uso por outras macrocélulas. As entradas do expansor paralelo permitem o empréstimo de termos-produto não usados de outras macrocélulas para expandir uma expressão de soma-de-produtos. A matriz de seleção de termos-produto é um arranjo de conexões programáveis que é usado para conectar as saídas selecionadas do arranjo AND e das entradas expansoras para a porta OR.

Expansores Compartilhados Um termo-produto complementado que pode ser usado para aumentar o número de termos-produto numa expressão de soma-de-produtos é disponibilizado a partir de cada macrocélula num LAB. A Figura 11–13 ilustra como um termo expansor compartilhado a partir de uma outra macrocélula pode ser usado para criar termos-produto adicionais. Nesse caso, cada uma das cinco portas AND num arranjo de macrocélulas está limitada a quatro entradas e, portanto, pode produzir termos-produto de até 4 variáveis, conforme ilustrado na parte (a) da figura. A Figura 11–13(b) mostra a expansão de dois termos-produto.



▲ **FIGURA 11–12**

Diagrama simplificado de uma macrocélula em um CPLD da série MAX 7000.



(a) Um arranjo de portas AND de 4 entradas podem produzir cada uma um termo-produto de 4 variáveis.

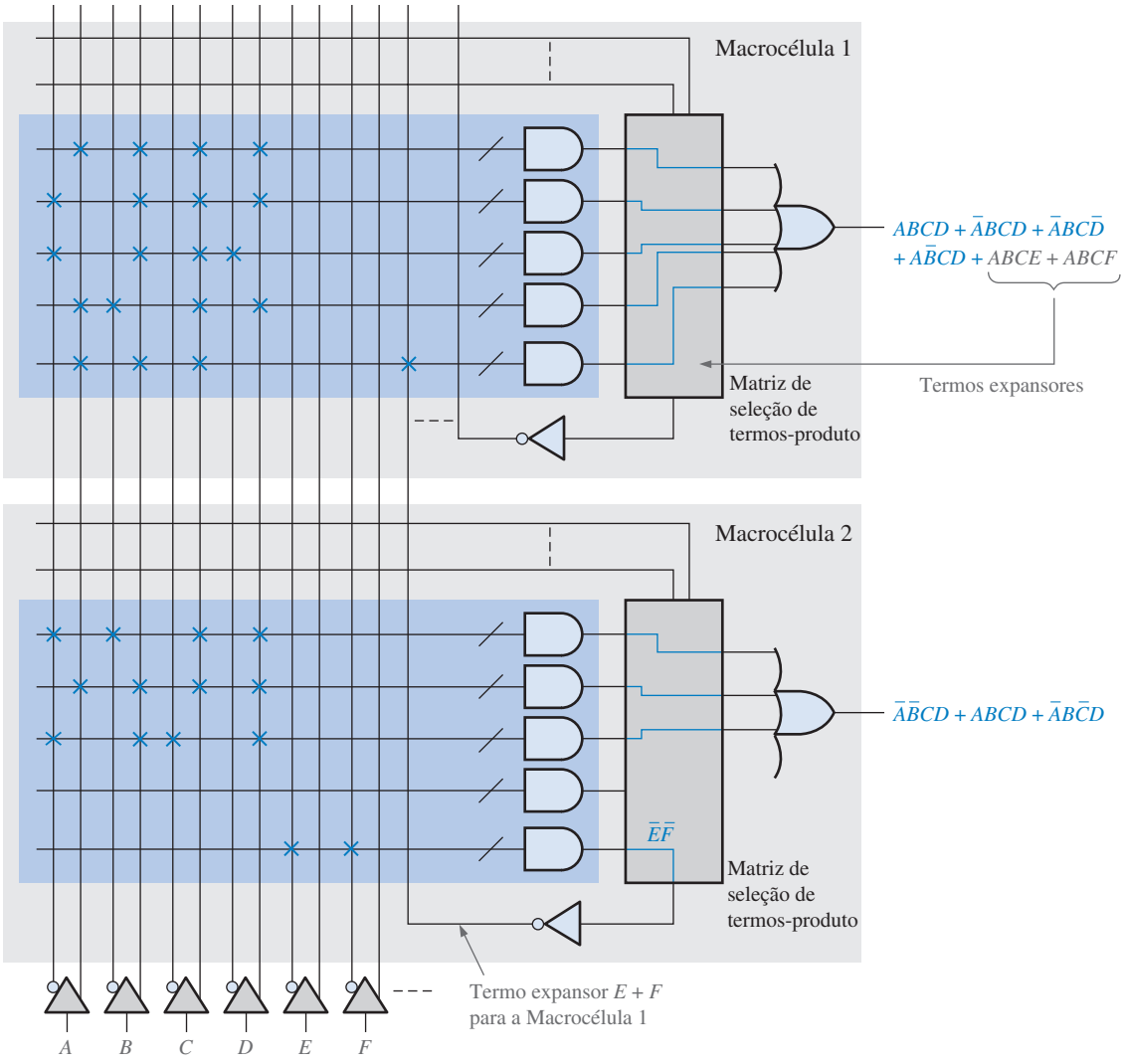
(b) Uma porta AND expandida para produzir dois termos-produto.

▶ **FIGURA 11–13**

Exemplo de como um expansor compartilhado pode ser usado numa macrocélula para aumentar o número de termos-produto.

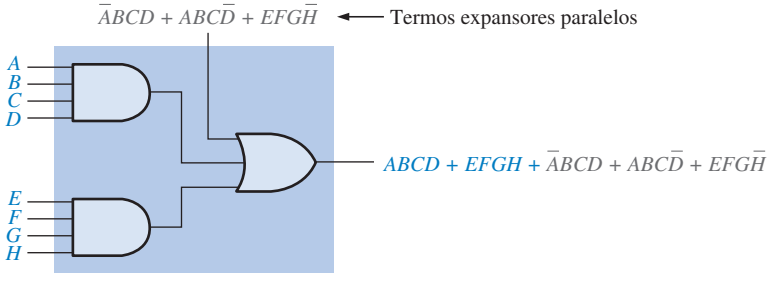
Cada macrocélula MAX 7000 pode produzir até cinco termos-produto gerados a partir do seu arranjo AND. Se uma macrocélula precisar mais de cinco termos-produto para a sua saída de soma-de-produtos, pode-se usar um termo expansor de uma outra macrocélula. Suponha que um projeto necessita de uma expressão de soma-de-produtos que contenha seis termos-produto. A Figura 11–14 mostra como um termo-produto de uma outra macrocélula pode ser usado para aumentar a saída de soma-de-produtos. A macrocélula 2, que está subutilizada, gera os termos expansores compartilhados $(E + F)$ que conecta a quinta porta AND na macrocélula 1 para produzir uma expressão de soma-de-produtos com seis termos-produto. As marcas na forma de X e as linhas na cor laranja representam as conexões produzidas no hardware pelo software compilador que executa o projeto programado.

Expansores Paralelos Uma outra forma de aumentar o número de termos-produto para uma macrocélula é usando expansores paralelos nos quais os termos-produto adicionais são submetidos a uma operação OR com os termos gerados por uma macrocélula em vez de combiná-los no arranjo AND, como no expansor compartilhado. Uma determinada macrocélula pode emprestar termos-produto não usados para as macrocélulas vizinhas (até cinco termos-produto a partir de outras três macrocélulas para o MAX 7000). O conceito básico está ilustrado na Figura 11–15 onde um circuito simplificado que pode produzir dois termos-produto empresta os três termos-produto restantes.



▲ **FIGURA 11–14**

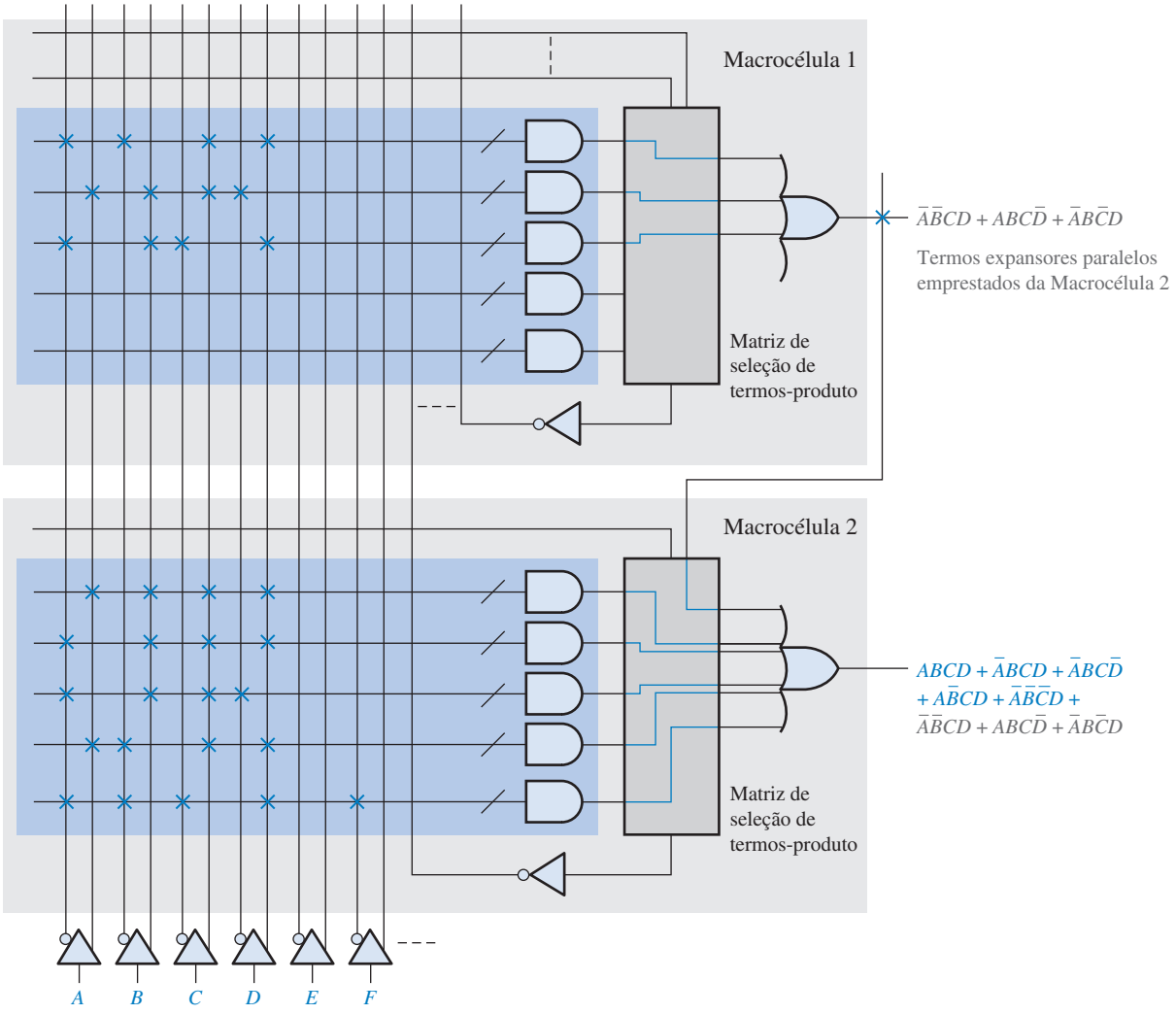
Ilustração simplificada do uso de um termo expansor compartilhado de uma outra macrocélula para aumentar a expressão de soma-de-produtos.



◀ **FIGURA 11–15**

Conceito básico do expansor paralelo.

A Figura 11–16 mostra como uma macrocélula pode tomar emprestado termos expansores paralelos a partir de outra macrocélula para aumentar a saída de soma-de-produtos. A macrocélula 2 usa três termos-produto da macrocélula 1 para produzir uma expressão de soma-de-produtos de oito termos. As marcas na forma de X e linhas na cor laranja representam as conexões produzidas no hardware pelo software compilador que executa o projeto programado.



▲ **FIGURA 11–16**

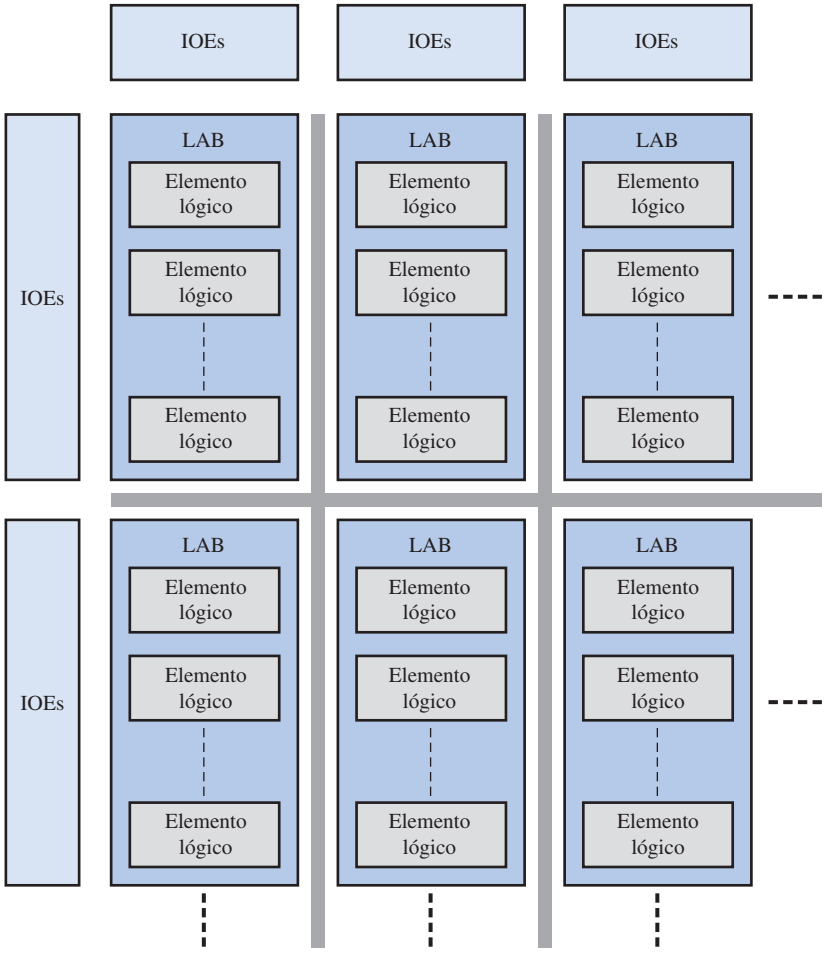
Ilustração simplificada do usos de termos expansores paralelos de outra macrocélula para aumentar a expressão de soma-de-produtos.

O CPLD MAX II

A arquitetura do CPLD MAX II difere bastante da família MAX 7000 e é o que o fabricante Altera chama de CPLD de "*post-macrocell*". Conforme mostra o diagrama em bloco na Figura 11–17, esse dispositivo contém blocos de arranjo lógico (LABs) cada um com elementos lógicos múltiplos (LEs). Um LE é a unidade de projeto lógico básica e é análoga à macrocélula. As interconexões programáveis são organizadas em linhas e colunas em torno dos LABs e os elementos de entrada/saída (IOEs) são situados ao longo do perímetro. A arquitetura dessa família de CPLDs é similar à de FPGAs, a qual discutimos na Seção 11–5. Na realidade, podemos imaginar a arquitetura MAX II como uma arquitetura FPGA de baixa densidade.

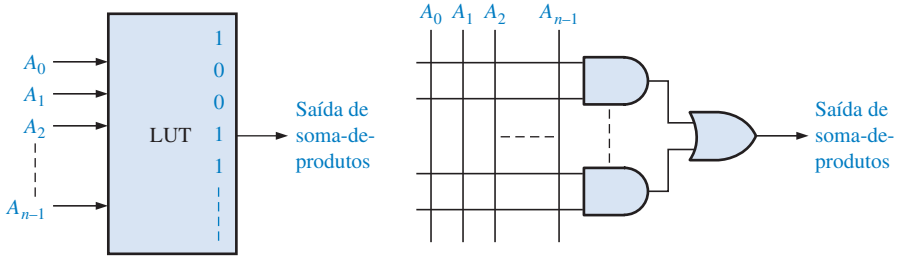
A principal diferença entre o CPLD MAX II e o SPLD clássico baseado no CPLD é a forma na qual uma função lógica é desenvolvida. O MAX II usa tabelas de busca (LUT) em vez de arranjos AND/OR. Uma **LUT** é basicamente um tipo de memória que pode ser programada para produzir funções de soma-de-produtos (discutida em mais detalhes na Seção 11–5). Essas duas abordagens são comparadas na Figura 11–18.

Conforme mencionado, o CPLD MAX II tem um arranjo de interconexões em linhas/colunas em vez de interconexões do tipo canal encontrado na maioria dos CPLDs clássicos. Essas duas abordagens são comparadas na Figura 11–19 e podem ser entendidas comparando as Figuras 11–11 e 11–17.



◀ **FIGURA 11–17**

Diagrama em bloco simplificado do CPLD MAX II.

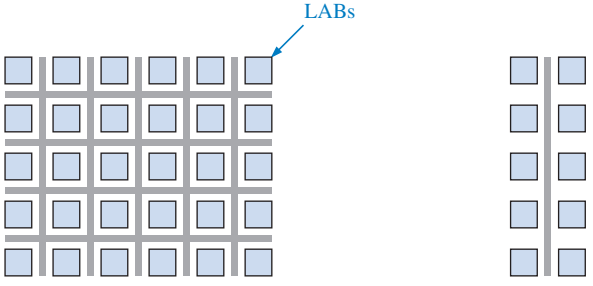


(a) Circuito lógico da tabela de busca. Um nível 1 é armazenado em cada endereço de termo-produto.

(b) Lógica de arranjo AND/OR

◀ **FIGURA 11–18**

CPLDs MAX II têm lógica LUT. CPLDs clássicos têm arranjos AND/OR.



(a) Interconexões de linhas/colunas

(b) Interconexões do tipo canal

◀ **FIGURA 11–19**

Os CPLDs MAX II têm interconexões de linhas/colunas. CPLDs clássicos têm interconexões do tipo canal.

A maioria dos CPLDs usa uma tecnologia de processo não-volátil para as conexões programáveis. Entretanto, a arquitetura MAX II usa uma tecnologia de processo baseada em SRAM que é **volátil** – toda a lógica programada é perdida quando a alimentação é desligada. Uma memória embutida no chip, a qual armazena as informações de programação do CPLD, usa tecnologia de memória não-volátil e reconfigura o dispositivo CPLD ao ser energizado.

SEÇÃO 11–2 REVISÃO

1. O que significa LAB?
2. Descreva um LAB em um CPLD MAX7000.
3. Qual é a finalidade de um expansor compartilhado?
4. Qual é a finalidade de um expansor paralelo?
5. Em que a arquitetura MAX II difere da MAX 7000?

11-3 CPLDs XILINX

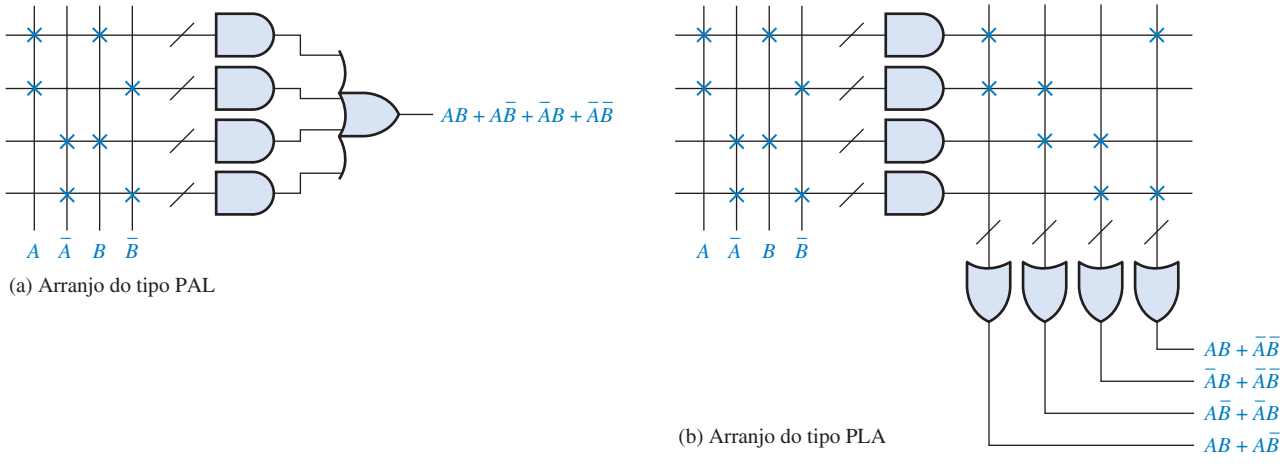
A Xilinx, assim como a Altera, produz uma série de CPLDs que variam em densidade, tecnologia de processo, consumo de potência e velocidade. Dentre as famílias produzidas por essa empresa estão a CoolRunner II, CoolRunnner XPLA3 e a XC9500. Essa última tem arquitetura similar à família CPLD MAX 7000 da Altera e apresenta a estrutura básica de dispositivos PAL/GAL. Nesta seção, abordaremos apenas a família CollRunner II para ilustrar os conceitos, tendo em mente que outras séries podem variar um pouco na arquitetura e/ou nos parâmetros mencionados anteriormente. Essa família de CPLDs é programável no próprio sistema (ISP) e está de acordo com o padrão JTAG.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever um dispositivo PLA e compará-lo com um PAL ■ Discutir a arquitetura de um CPLD CoolRanner II ■ Descrever um bloco funcional

PLA (Arranjo Lógico Programável)

Conforme estudamos, a arquitetura de um CPLD é a forma na qual os elementos internos são organizados e combinados. A arquitetura da família CoolRunner II da Xilinx é baseada na estrutura de um PLA (*programmable logic array* – arranjo lógico programável) em vez de um PAL (lógica de arranjo programável). A Figura 11–20 compara a estrutura de um dispositivo PAL simples com



(a) Arranjo do tipo PAL

(b) Arranjo do tipo PLA

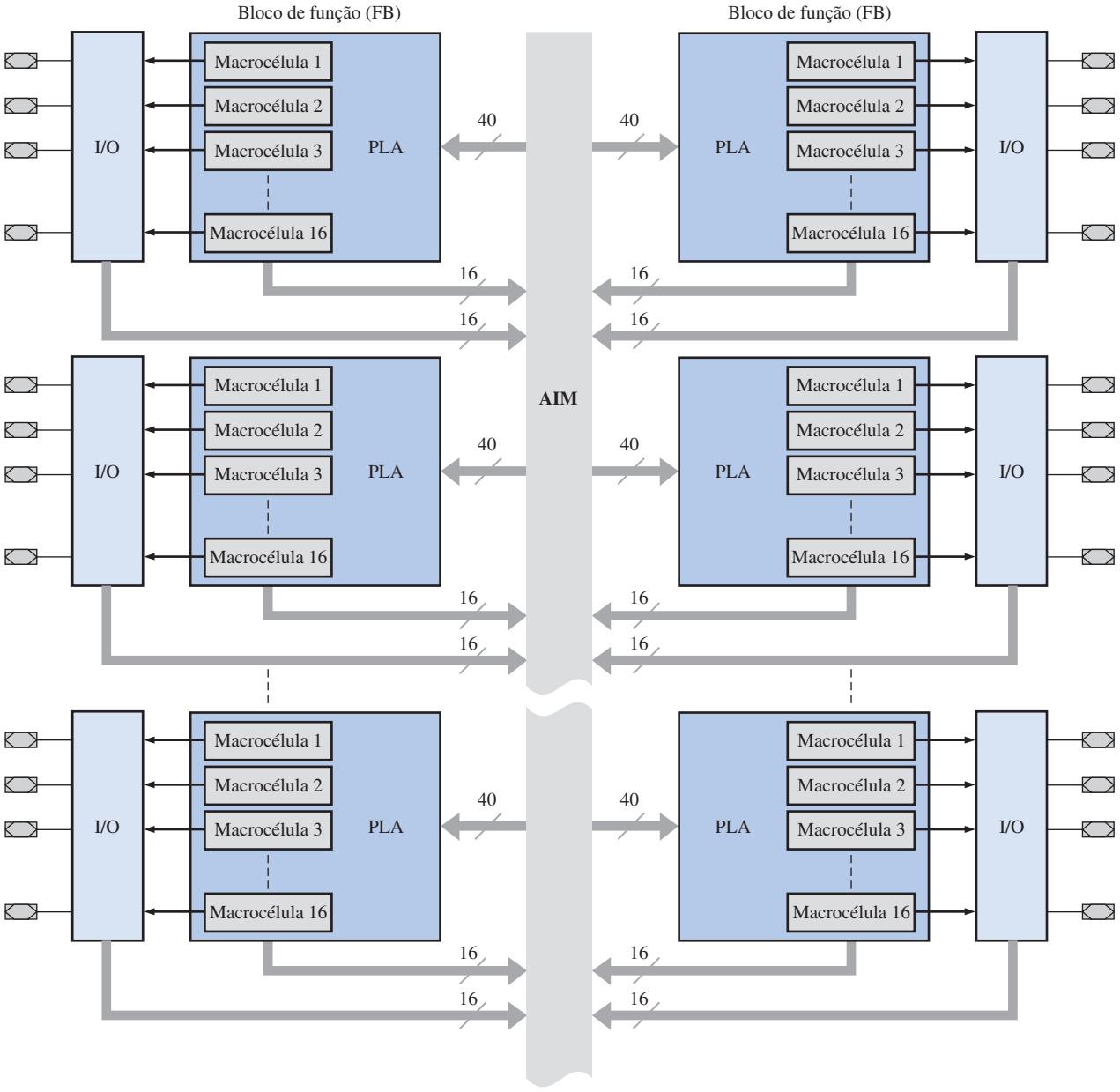
▲ **FIGURA 11–20**

Comparação entre dispositivos PLA e PAL básicos.

a de um PLA simples. Como sabemos, o dispositivo PAL tem um arranjo AND programável seguido por um arranjo OR fixo e produz uma expressão de soma-de-produtos, conforme mostra o exemplo dado na Figura 11–20(a). O dispositivo **PLA** tem um arranjo AND programável seguido por um arranjo OR também programável, conforme mostra o exemplo dado na Figura 11–20(b).

CoolRunner II

Um CPLD com arquitetura CoolRunner II usa uma estrutura do tipo PLA. Esse dispositivo tem múltiplos blocos de funções (FBs), os quais são análogos aos LABs na arquitetura MAX 7000 da Altera (Figura 11–11). Cada bloco de função contém dezesseis macrocélulas, exatamente como um LAB. Os blocos de funções são interconectados por uma matriz de interconexões avançadas (AIM), análoga ao PIA em um MAX 7000. Um diagrama em bloco básico da arquitetura CoolRunner II é mostrado na Figura 11–21. Conforme podemos ver, do ponto de vista do diagrama em

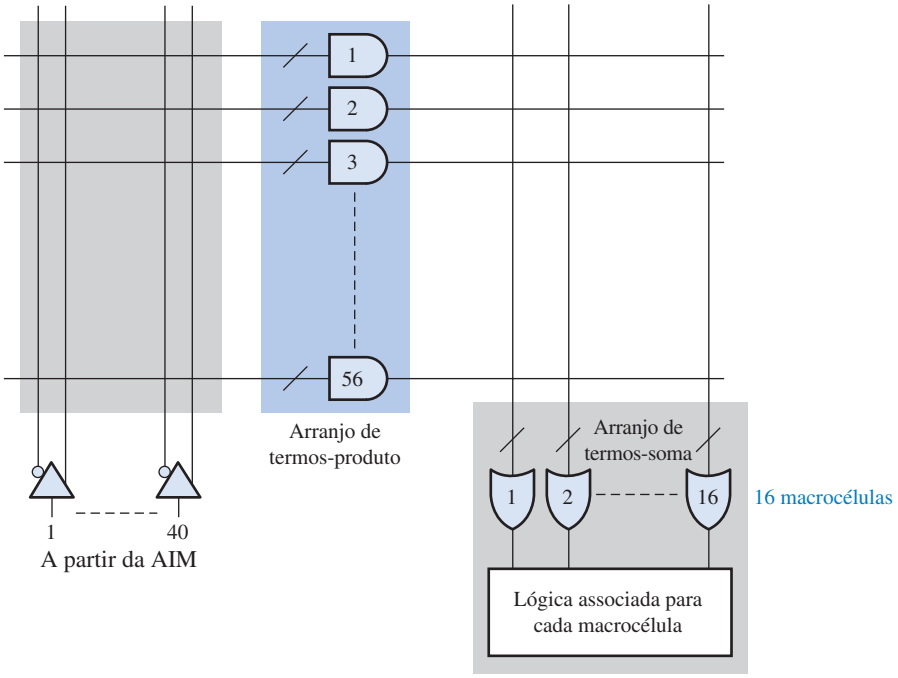


▲ **FIGURA 11–21**

Diagrama em bloco básico de um CPLD com arquitetura CoolRunner II.

bloco básico, não existem muitas diferenças entre um CPLD da Xilinx e um da Altera; entretanto, internamente existem diferenças.

A série CoolRunner II de CPLDs contém de 32 a 512 macrocélulas. Como existem 16 macrocélulas por bloco de função, o número de blocos de funções varia de 2 a 32. Um diagrama bastante simplificado de um bloco de função (FB) é mostrado na Figura 11–22. O arranjo AND programável tem 56 portas AND e o arranjo OR programável tem 16 portas OR. Com a estrutura PLA, qualquer termo-produto pode ser conectado a qualquer porta OR para criar uma saída de soma-de-produtos. Com utilização máxima, cada FB pode produzir 16 saídas de soma-de-produtos tendo cada uma 56 termos-produto. Essa macrocélula é abordada em detalhes na Seção 11–4.



▲ **FIGURA 11–22**

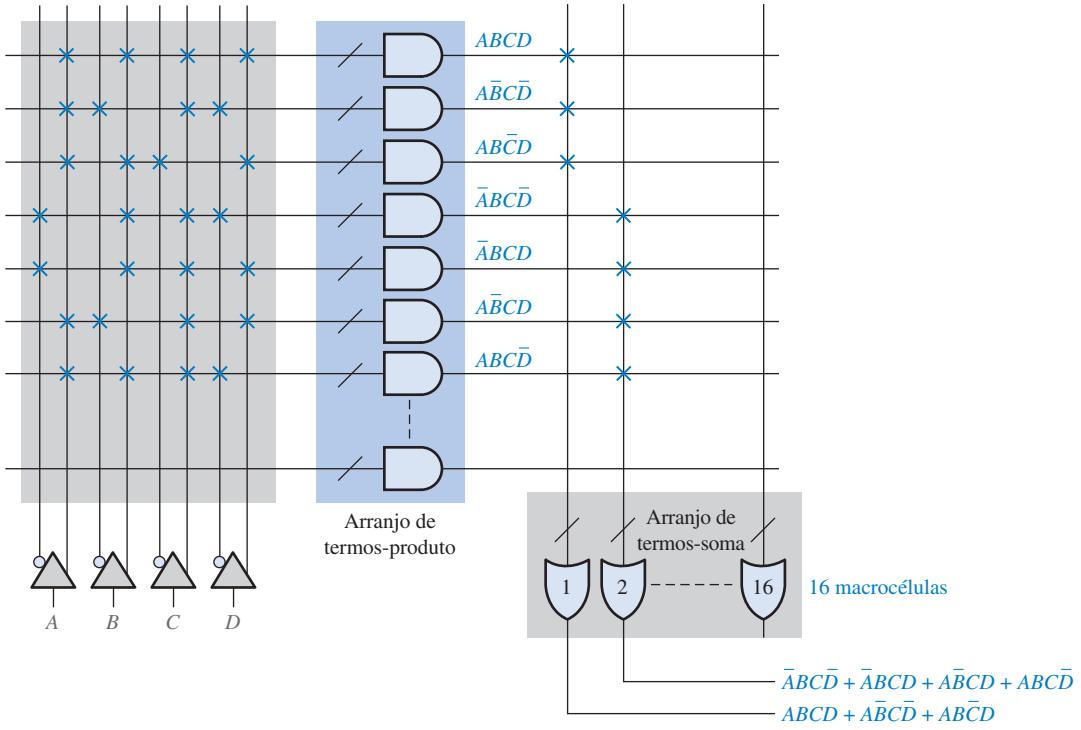
Diagrama simplificado de um bloco de função (FB) com uma estrutura PLA.

EXEMPLO 11–2

Mostre as conexões programadas no FB simplificado mostrado na Figura 11–22 para gerar a seguinte função de soma-de-produtos a partir da macrocélula 1: $AB\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D}$ e a seguinte função de soma-de-produtos a partir da macrocélula 2: $\overline{A}B\overline{C}\overline{D} + \overline{A}BCD + A\overline{B}CD + AB\overline{C}\overline{D}$.

Solução As marcas na forma de X na cor laranja na Figura 11–23 indicam conexões programadas nos arranjos AND e OR.

Problema relacionado Quantas funções do tipo soma-de-produtos podem ser geradas pelo FB mostrado na Figura 11–23?



▲ **FIGURA 11–23**

SEÇÃO 11–3 REVISÃO

1. Qual é a principal diferença entre os dispositivos CPLDs Altera e Xilinx?
2. Descreva um PLA.
3. Em que um PLA difere de um PAL?
4. O que significa FB?

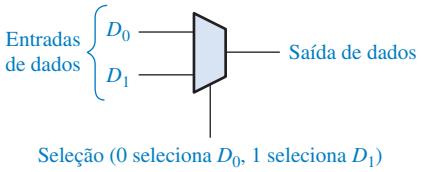
11-4 MACROCÉLULAS

As macrocélulas CPLD foram apresentadas em seções anteriores tanto para dispositivos Altera quanto Xilinx. Lembre que uma macrocélula pode ser configurada por programação como lógica combinacional ou entradas e saídas lógicas registradas (com memorização). O termo *registrador* se refere ao uso de flip-flops. Nesta seção, aprenderemos sobre a macrocélula, incluindo os modos de operação combinacional e registrado. Embora a arquitetura de uma macrocélula varie dentre CPLDs diferentes, dispositivos representativos são usados para ilustração.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever a operação de uma macrocélula MAX 7000 da Altera ■ Descrever a operação de uma macrocélula CPLD CoolRunner II da Xilinx

Os diagramas lógicos freqüentemente usam o símbolo mostrado na Figura 11–24 para representar um multiplexador. Nesse caso, o multiplexador tem duas entrada de dados e uma entrada de seleção programável; a entrada de seleção normalmente não é mostrada no diagrama lógico.

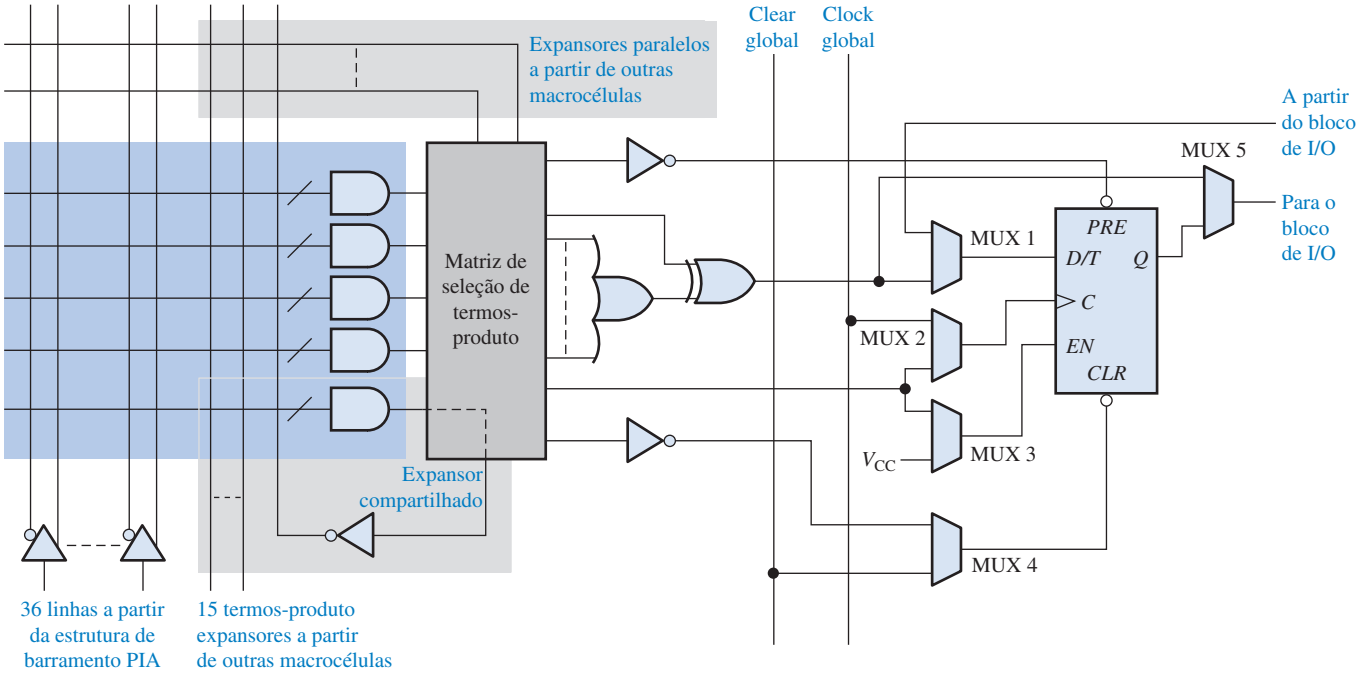


◀ **FIGURA 11–24**

Símbolo usado normalmente para representar um multiplexador. Ele pode ter qualquer número de entradas.

A Macrocélula MAX 7000 da Altera

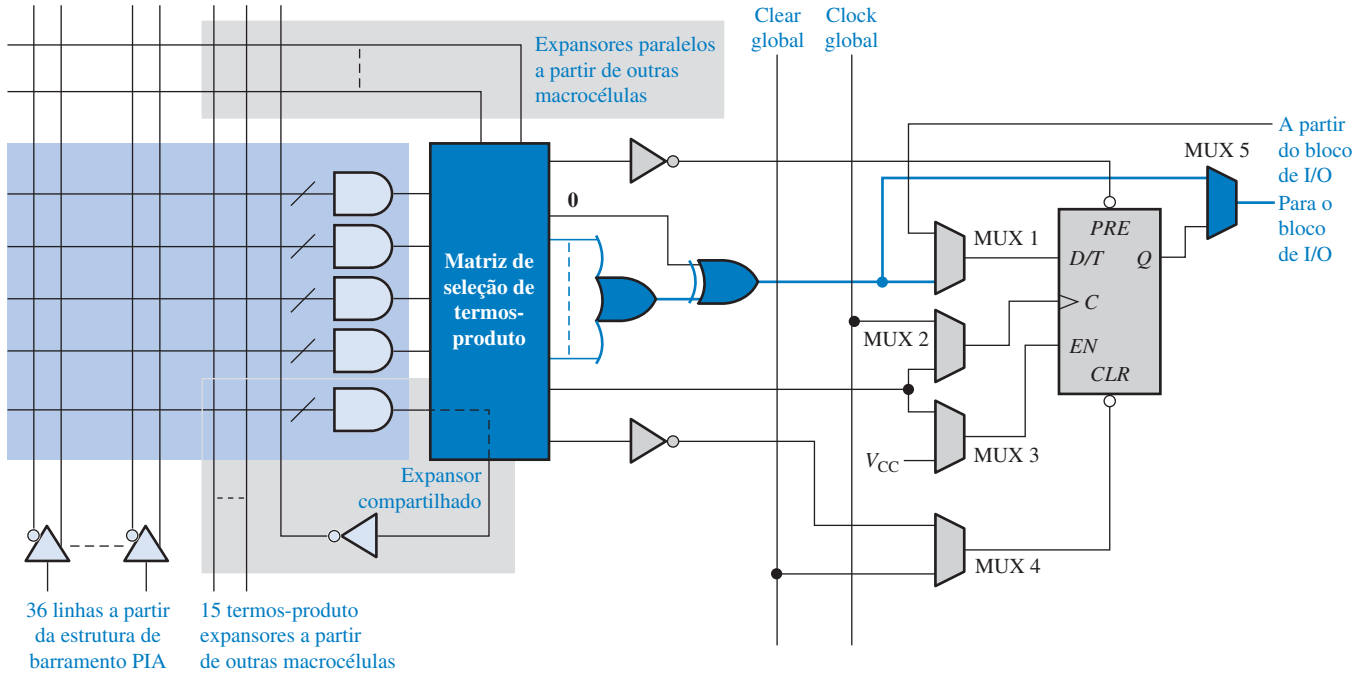
A Figura 11–25 mostra o diagrama completo da macrocélula incluindo o flip-flop (registrador). A porta EX-OR proporciona a complementação da função de soma-de-produtos a partir da porta OR para gerar uma função na forma de produto-de-somas. Um nível 1 na entrada superior da porta EX-OR complementa a saída da OR, enquanto um nível 0 permite que a saída da OR passe sem complementação (na forma de soma-de-produtos). O MUX 1 provê a seleção entre a saída da EX-OR e a entrada a partir do bloco I/O. O MUX 2 pode ser programado para selecionar entre o clock global e o sinal de clock baseado num termo-produto. O MUX 3 pode ser programado para selecionar entre um nível ALTO (V_{CC}) e um termo-produto para habilitação do flip-flop. O MUX 4 pode selecionar entre o *clear* global e um *clear* de termo-produto. O MUX 5 é usado para *bypass* (desvio) do flip-flop e conecta a saída da lógica combinacional ao bloco I/O. O flip-flop pode ser programado como sendo do tipo D, T (*toggle*), J-K ou S-R.



▲ **FIGURA 11–25**

Uma macrocélula na família MAX 7000 de CPLDs da Altera.

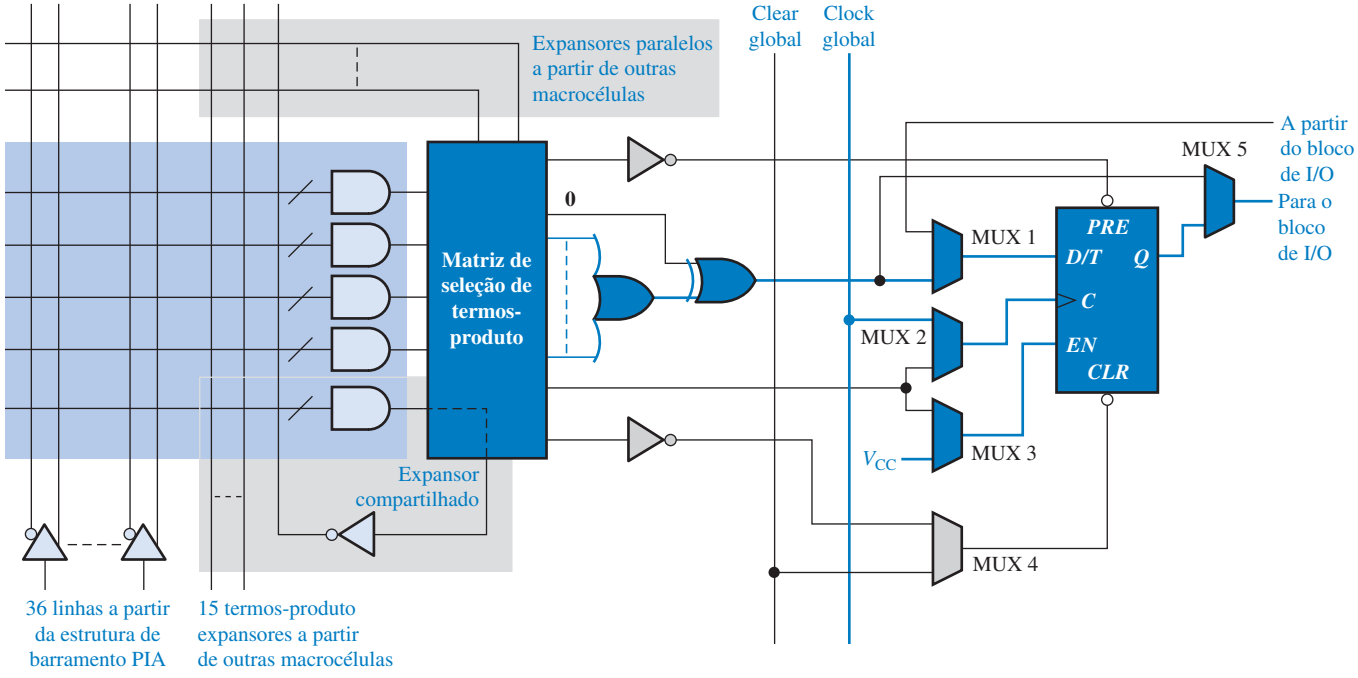
O Modo Combinacional Quando uma macrocélula é programada para produzir uma função lógica combinacional de produto-de-somas, os elementos lógicos envolvidos, por onde os dados trafegam, são mostrados na cor laranja na Figura 11–26. Como podemos ver, apenas um MUX é usado e o registrador (flip-flop) sofre um *bypass*.



▲ **FIGURA 11–26**

Uma macrocélula configurada para gerar uma função lógica de soma-de-produtos. O percurso do tráfego de dados é indicado com a cor laranja.

O Modo Registrado Quando uma macrocélula é programada no modo registrado, com a saída lógica combinacional gerando a entrada de dados para o registrador disparado pelo clock global, os elementos por onde os dados trafegam são mostrados na cor laranja na Figura 11–27. Conforme podemos ver, quatro muxes são usados e o registrador (flip-flop) é ativado.



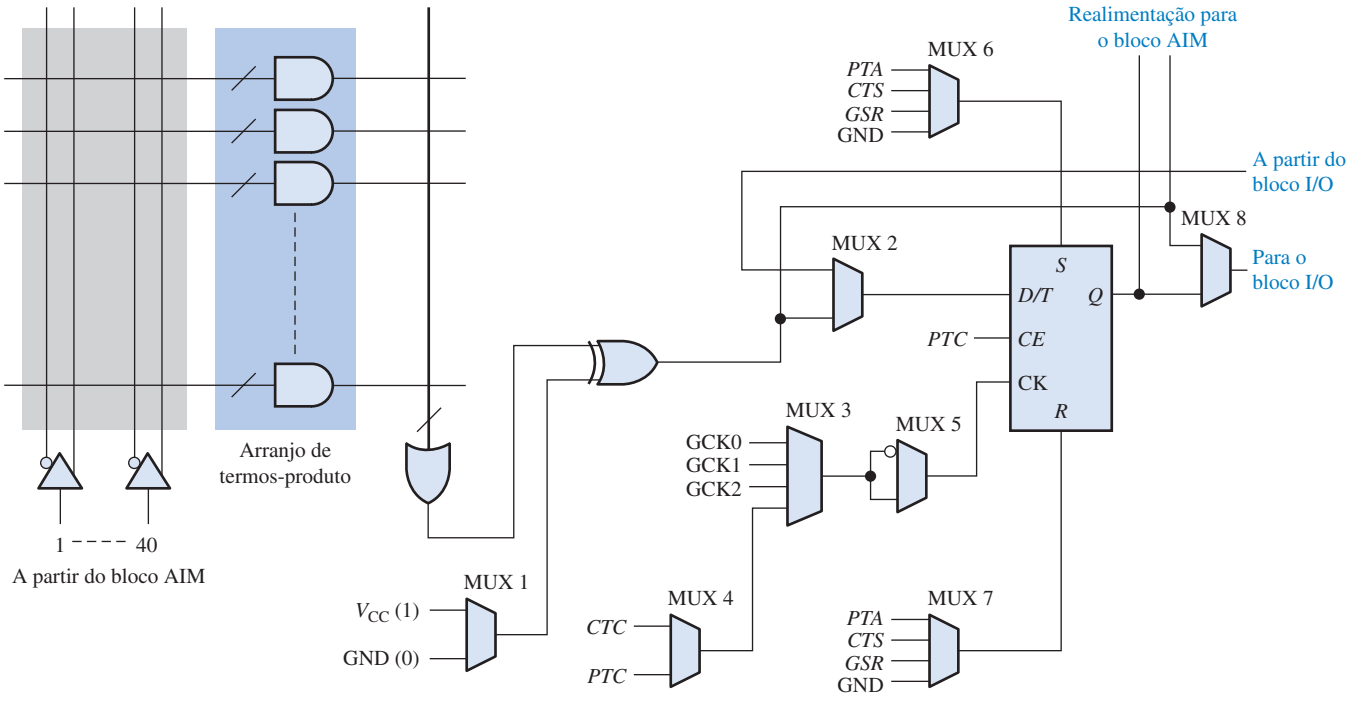
▲ **FIGURA 11–27**

Uma macrocélula configurada para gerar uma função lógica com registrador (memória). O percurso do tráfego de dados é indicado com a cor laranja.

A Macrocélula CoolRunner II da Xilinx

A macrocélula de um CPLD CoolRunner II foi apresentada de forma concisa na Seção 11–3. Lembre que esse dispositivo possui uma arquitetura PLA, onde o arranjo AND e o arranjo OR são programáveis. A Figura 11–28 mostra a lógica completa para essa macrocélula, incluindo o flip-flop (registrador). A porta OR tem múltiplas entradas a partir do arranjo AND conforme indicado pela linha de corte transversal à linha de entrada da porta OR.

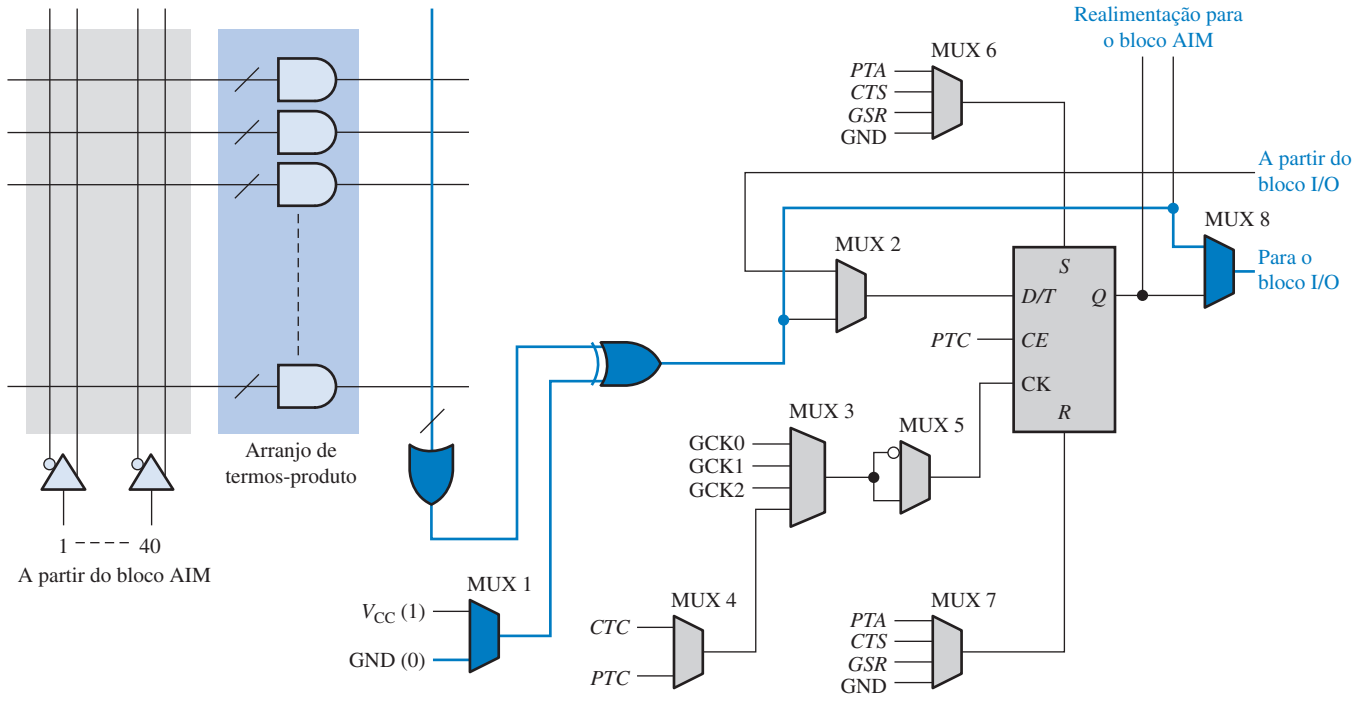
A porta EX-OR provê a complementação da função de soma-de-produtos a partir da porta OR gerando uma função na forma de produto-de-somas. Um nível 1 na entrada na entrada inferior da porta EX-OR complementa a saída da OR e, por sua vez, um nível 0 permite que a saída da OR passe sem complementação (na forma de soma-de-produtos). O MUX 1 provê a seleção entre as saídas lógicas de soma-de-produtos e de produto-de-somas. O MUX 2 provê a seleção entre a saída da porta EX-OR e a entrada do bloco de I/O. o MUX 3 e o MUX 4 podem ser programados para selecionar entre um dos clocks globais (GCK0, GCK1 ou GCK2) e um sinal de clock baseado em um termo-produto (*CTC* ou *PTC*). *CTC* é um termo compartilhado e *PTC* é um termo gerado localmente. O MUX 5 pode ser programado para proporcionar qualquer uma das polaridades do sinal de clock. O termo-produto *PTC* é usado para prover uma habilitação de clock para o flip-flop. O MUX 6 pode selecionar um dos quatro sinais para setar o flip-flop. Esses sinais são o *PTA* (termo-produto gerado localmente), o *CTS* (termo-produto compartilhado), o *GSR* (set/reset global) e o GND, que normalmente é selecionado quando um SET ativo é solicitado. O MUX 7 provê as mesmas funções para o *clear* ou resete do flip-flop assim como o MUX 6 o faz para a operação SET. O MUX 8 é usado para operação de *bypass* do flip-flop conectando a saída da lógica combinacional ao bloco de I/O ou conectando a saída do registrador ao bloco de I/O. O flip-flop pode ser programado como um do tipo D, T (*toggle*) ou como um *latch*.



▲ **FIGURA 11–28**

Uma macrocélula de um CPLD CoolRunner II da Xilinx.

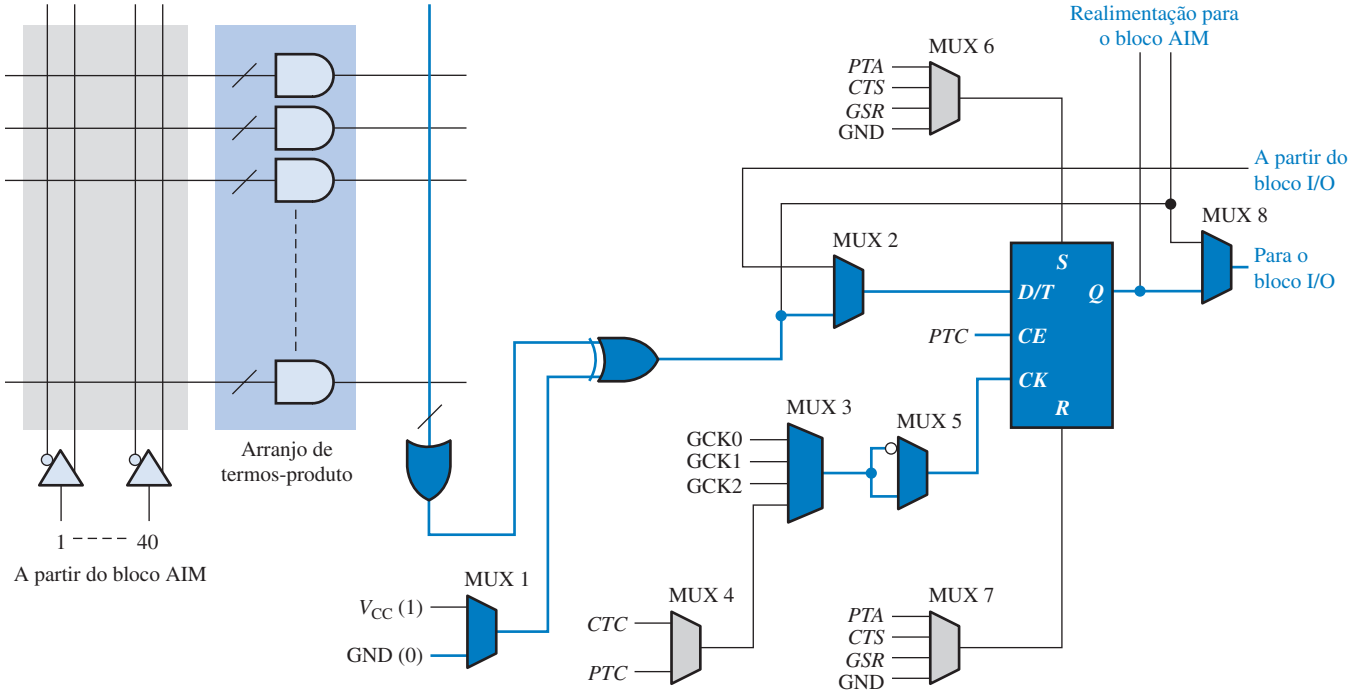
Modo Combinacional Quando uma macrocélula é programada para gerar uma função lógica combinacional na forma de soma-de-produtos, os elementos lógicos, pelos quais os dados trafegam, são aqueles mostrados em laranja na Figura 11–29. Conforme podemos ver, apenas dois muxes são usados e o registrador (flip-flop) não está incluído no caminho do tráfego de dados (*bypass*).



▲ **FIGURA 11–29**

Uma macrocélula configurada para gerar uma função lógica de soma-de-produtos. O percurso do tráfego de dados é indicado em laranja.

Modo Registrado Quando uma macrocélula é programada para o modo registrado com a saída da lógica combinacional sendo a entrada de dados do registrador disparado por um dos sinais de clock global, o tráfego dos dados ocorre através daqueles elementos mostrados em laranja na Figura 11–30. Conforme podemos ver, cinco muxes são usados e o registrador (flip-flop) é ativado.



▲ **FIGURA 11–30**

Uma macrocélula configurada para gerar uma função lógica registrada. O percurso do tráfego de dados é indicado em laranja.

SEÇÃO 11–4 REVISÃO

1. Explique a finalidade da porta EX-OR na macrocélula.
2. Quais são os dois principais modos de uma macrocélula?
3. O termo *registrado* se refere a que?
4. Além da porta OR, da porta EX-OR e do flip-flop, qual outro elemento lógico é usado normalmente em uma macrocélula?

11-5 LÓGICA PROGRAMÁVEL: FPGAs

Conforme estudamos, a arquitetura de uma CPLD clássica consiste em blocos lógicos do tipo PAL/GAL ou PLA com interconexões programáveis. Basicamente, o dispositivo FPGA (*field-programmable gate array* – arranjo de portas programáveis por ação de campo) possui uma arquitetura diferente (não faz uso de arranjos do tipo PAL/PLA) e possui densidades muito maiores que os dispositivos CPLDs. Um FPGA típico tem, muitas vezes, mais portas equivalentes que um CPLD típico.

Ao final do estudo desta seção você deverá ser capaz de:

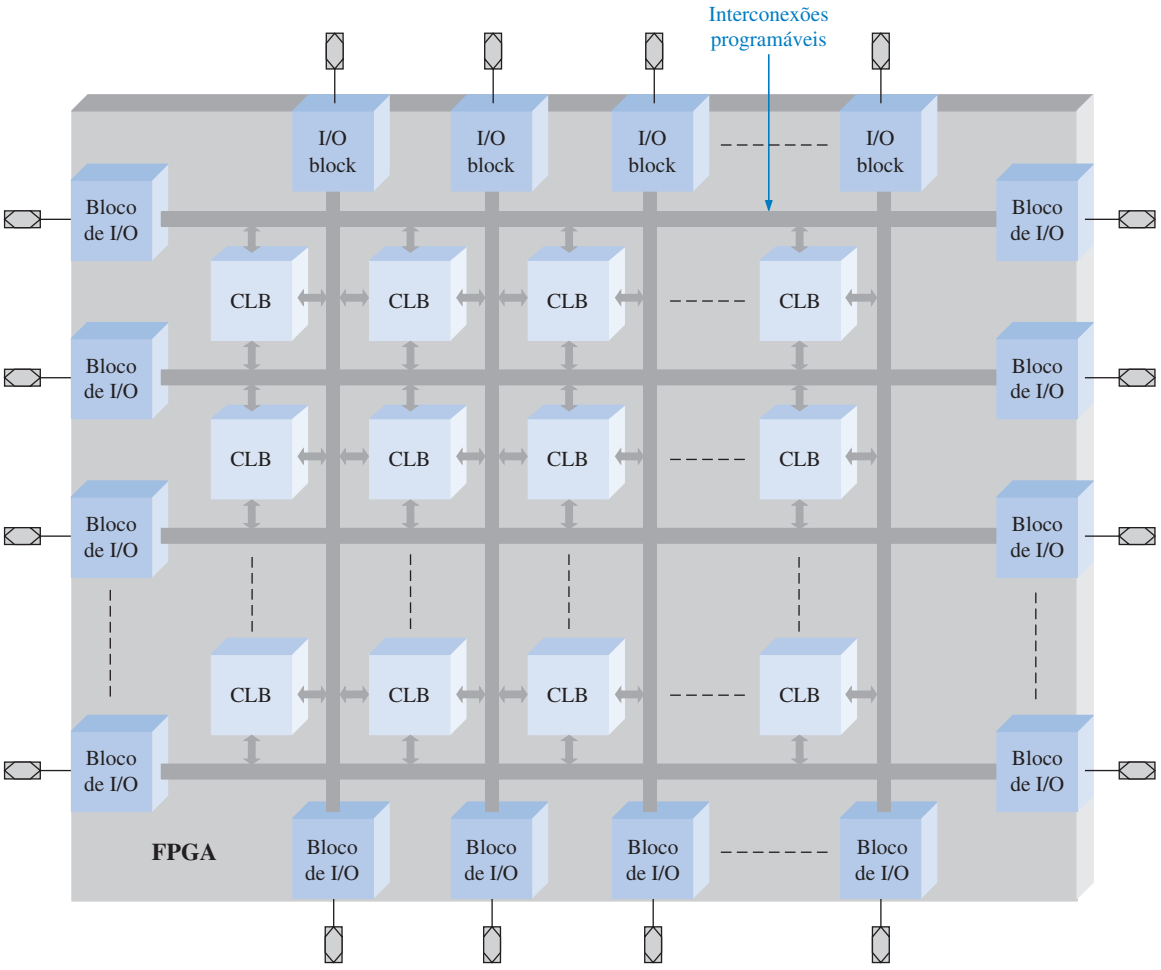
■ Descrever a estrutura básica de um FPGA ■ Comparar um FPGA com um CPLD ■ Discutir os blocos de memória LUT ■ Discutir o FPGA baseado em SRAM ■ Definir o núcleo de um FPGA

O conceito básico de um FPGA foi apresentado no Capítulo 1. Os três elementos básicos de um **FPGA** são o bloco lógico configurável (CLB), as interconexões e os blocos de entrada/saída (I/O), conforme ilustrado na Figura 11–31. Os blocos lógicos configuráveis (CLBs) em um FPGA não são tão complexos quanto os LABs ou FBs em um CPLD, porém geralmente a quantidade de CLBs é bem maior. Nos casos em que os CLBs são relativamente simples, a arquitetura do FPGA é denominada *fine graned* (granulação fina). Nos casos em que os CLBs são maiores e mais complexos, a arquitetura é denominada *coarse grained* (granulação grossa). Os blocos de I/O em torno do perímetro da estrutura provê entradas, saídas ou acesso bidirecionais ao mundo externo, selecionáveis individualmente. A matriz distribuída de interconexões programáveis provê a interconexão entre CLBs e a conexão à entradas e saídas. FPGAs maiores podem ter dezenas de centenas de CLBs além de memória e outros recursos.

A maioria dos fabricantes de dispositivos lógicos programáveis produzem séries de FPGAs que variam em densidade, consumo de potência, tensão de alimentação, velocidade e, em determinado grau, variam na arquitetura. Os FPGAs são reprogramáveis e usam tecnologia SRAM ou de antifusível nas conexões programáveis. As densidades dos dispositivos podem variar desde centenas de módulos lógicos a aproximadamente 180.000 módulos lógicos em encapsulamentos com mais de 1000 pinos. As tensões de alimentação cc estão tipicamente na faixa de 1,2 V a 2,5 V, dependendo do dispositivo específico.

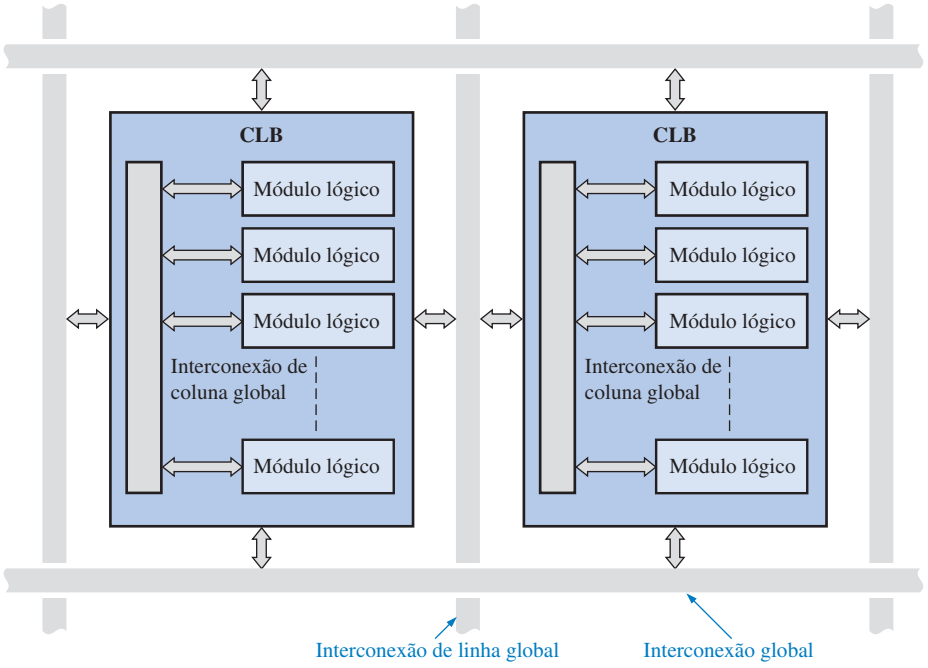
Blocos Lógicos Configuráveis

Tipicamente, um bloco lógico de um FPGA consiste em vários módulos lógicos menores, os quais são as unidades construtivas básicas, que por sua vez são análogas às macrocélulas de um dispositivo CPLD. A Figura 11–32 mostra os blocos lógicos configuráveis (CLBs) fundamentais dentro das interconexões programáveis globais em linha/coluna que são usadas para interconectar blocos lógicos. Cada **CLB** é formado de múltiplos módulos lógicos menores e uma interconexão programável local que é usada para interconectar módulos dentro de uma CLB.



▲ **FIGURA 11–31**

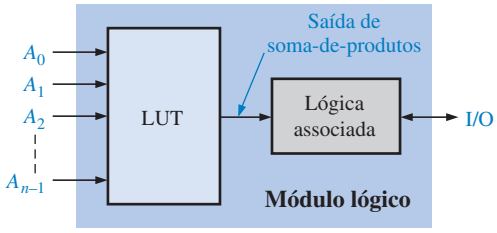
Estrutura básica de um FPGA. O bloco lógico configurável é o CLB.



◀ **FIGURA 11–32**

Blocos lógicos configuráveis básicos (CLBs) dentro de interconexões programáveis globais em linha/coluna.

Módulos Lógicos Um módulo lógico em um bloco lógico de FPGA pode ser configurado como lógica combinacional, lógica registrada ou uma combinação de ambas. Um flip-flop é parte da lógica associada e é usado na lógica registrada. (Os flip-flops foram abordados no Capítulo 7). A Figura 11–33 mostra o diagrama em bloco de um típico módulo lógico baseados em LUT. Conforme sabemos, uma LUT (tabela de busca) é um tipo de memória programável usada para gerar funções lógicas combinacionais de soma-de-produtos. A LUT faz essencialmente o mesmo trabalho que um PAL ou PLA.

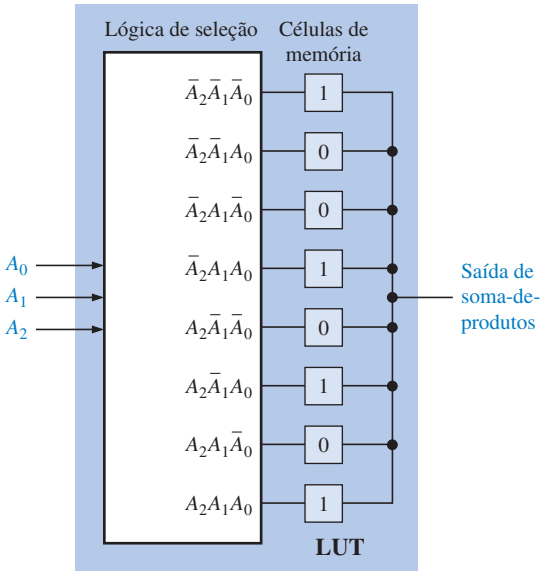


► **FIGURA 11–33**

Diagrama em bloco básico de um módulo lógico de um FPGA.

Geralmente, a organização de uma LUT consiste em um número de células de memória igual a 2^n, onde n é o número de variáveis de entrada. Por exemplo, três entradas podem selecionar até 8 células de memória. Assim, uma LUT com três variáveis de entrada pode produzir um expressão de soma-de-produtos com até oito termos-produto. Pode ser programado um padrão de 1s e 0s nas células de memória de uma LUT, conforme ilustra a Figura 11–34 para uma função de soma-de-produtos específica. Cada nível 1 significa o termo-produto associado que aparece na saída de soma-de-produtos, e cada 0 significa que o termo-produto associado não aparece na saída de soma-de-produtos. A expressão resultante da saída de soma-de-produtos é

$$\overline{A}_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1A_0 + A_2\overline{A}_1A_0 + A_2A_1A_0$$



► **FIGURA 11–34**

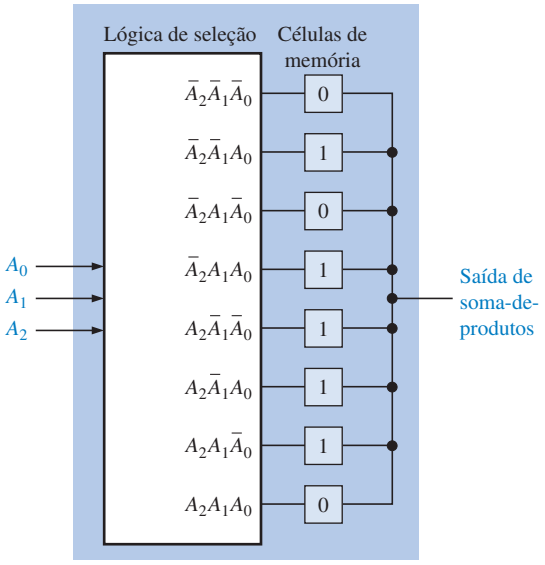
O conceito básico de uma LUT programada como uma determinada saída de soma-de-produtos.

EXEMPLO 11–3

Mostre uma LUT básica de três variáveis programada para produzir a seguinte função de soma-de-produtos:

$$A_2A_1\overline{A}_0 + A_2\overline{A}_1\overline{A}_0 + \overline{A}_2A_1A_0 + A_2\overline{A}_1A_0 + \overline{A}_2\overline{A}_1A_0$$

Solução Um nível 1 é armazenado para cada termo-produto na expressão de soma-de-produtos, conforme mostra a Figura 11–35.



▶ **FIGURA 11–35**

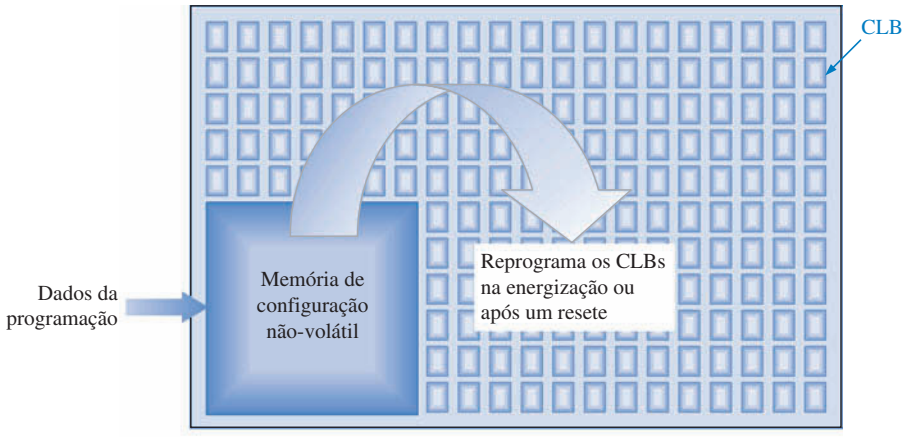
Problema relacionado Quantas células de memória teria uma LUT com quatro variáveis de entrada? Qual seria o número máximo possível de termos-produto na saída de soma-de-produtos?

FPGAs Baseadas em SRAM

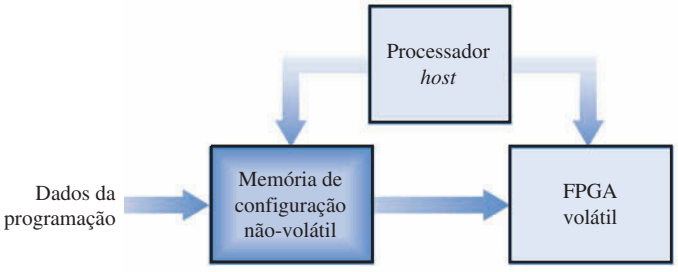
As FPGAs são dispositivos não-voláteis nos casos em que são baseados em tecnologia antifusível ou são voláteis quando são baseados em tecnologia SRAM. O termo *volátil* significa que todos os dados programados nos blocos lógicos configuráveis são perdidos quando a alimentação é desligada. Portanto, FPGAs baseadas em SRAM incluem uma memória não-volátil embutida no chip, para armazenar os dados e reconfigurar o dispositivo cada vez que ele for energizado, ou eles usam uma memória externa com transferência de dados controlada por um processador *host* (hospedeiro). O conceito de memória no chip é ilustrado na Figura 11–36(a). O conceito da configuração de processador *host* é mostrado na parte (b).

Núcleos de FPGAs

As FPGAs, conforme temos discutido, são essencialmente como "quadros em branco" em que o usuário final pode programar qualquer projeto lógico. Estes dispositivos comercializados também contêm um núcleo de lógica rígida. Um **núcleo rígido** corresponde a uma parte da lógica de uma FPGA que é inserida pelo fabricante para prover uma função específica a qual não pode ser reprogramada. Por exemplo, se um cliente necessita de um pequeno microprocessador como parte do projeto de um sistema, ele pode ser programado no FPGA pelo cliente ou pode ser fornecido pelo fabricante como um núcleo rígido. Se a função embutida tiver algumas características programáveis, ela é conhecida como função de **núcleo flexível**. Uma vantagem da abordagem de núcleo



(a) FPGA volátil com uma memória de configuração não-volátil no chip

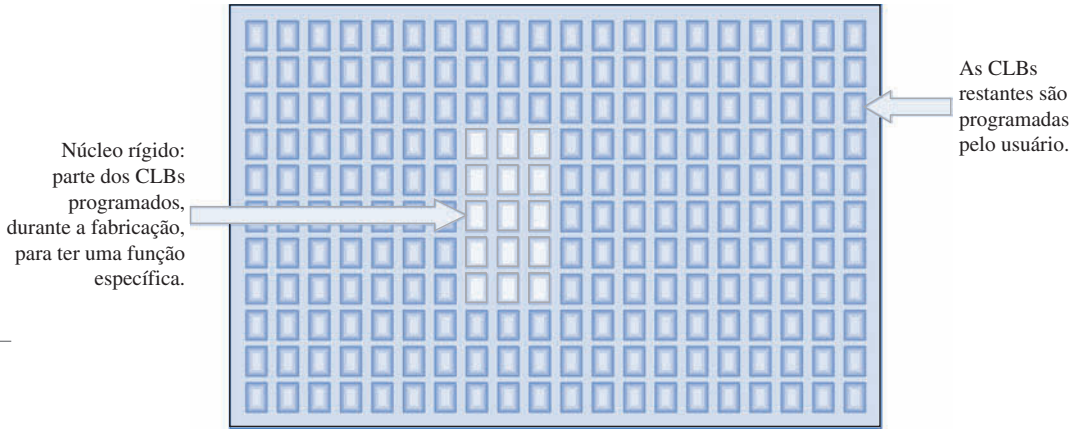


(b) FPGA volátil com memória no circuito e processador *host*

► **FIGURA 11–36**
Conceitos básicos de configurações de FPGA volátil.

rígido é que o mesmo projeto pode ser implementado usando bem menos da capacidade disponível do FPGA do que se o usuário programasse a função no chip, resultando em menos espaço ("bens imóveis") e menos tempo de desenvolvimento para o usuário. Além disso, as funções de núcleo rígido são completamente testadas. A desvantagem do núcleo rígido é que as especificações são estabelecidas durante a fabricação, sendo que o cliente tem que usar a lógica de núcleo rígido "como ela é". Ela não pode ser alterada posteriormente.

Os núcleos rígidos são geralmente comercializados com funções que normalmente são usadas em sistemas digitais, tal como um microprocessador, interfaces padrão de entrada/saída e processadores de sinais digitais. Mais do que uma função de núcleo rígido pode ser programada em um FPGA. A Figura 11–37 ilustra o conceito de um núcleo rígido envolto por uma lógica configurável programada pelo usuário. Trata-se de um sistema básico embutido porque a função de núcleo rígido é embutida na lógica programada pelo usuário.



► **FIGURA 11–37**
Idéia básica de uma função de núcleo rígido embutida em uma FPGA.

Os projetos de núcleos rígidos são geralmente desenvolvidos pelos fabricantes de FPGAs sendo propriedade destes. Os projetos desenvolvidos por um fabricante são denominados de **propriedade intelectual** (IP). Uma empresa geralmente apresenta uma lista com os tipos de propriedades intelectuais que estão disponíveis no seu website. Algumas propriedades intelectuais são um combinado de núcleo rígido com núcleo flexível. Um processador que tem alguma flexibilidade na seleção e ajuste de certos parâmetros pelo usuário é um exemplo.

Os FPGAs que contêm processadores embutidos implementados com núcleo rígido ou núcleo flexível, ou ainda ambos, são conhecidos como **FPGAs de plataforma** porque eles podem ser usados para implementar um sistema completo sem a necessidade de dispositivos de suporte externo.

SEÇÃO 11–5 REVISÃO

1. Em que um FPGA difere de um CPLD?
2. O que significa CLB?
3. Descreva uma LUT e discuta sua finalidade.
4. Qual é a diferença entre uma interconexão local e uma global em uma FPGA?
5. O que é um núcleo de FPGA?
6. Defina o termo *propriedade intelectual* em relação a um fabricante de FPGA.

11-6 FPGAs ALTERA

A empresa Altera produz diversas famílias de FPGAs incluindo as famílias Stratix II, Stratix, Cyclone e ACEX. Nesta seção, focalizaremos apenas a família Stratix II para ilustrar os conceitos, tendo em mente que outros dispositivos na família podem diferir basicamente em certos aspectos da arquitetura deles e/ou parâmetros tais como densidade, velocidade e potência.

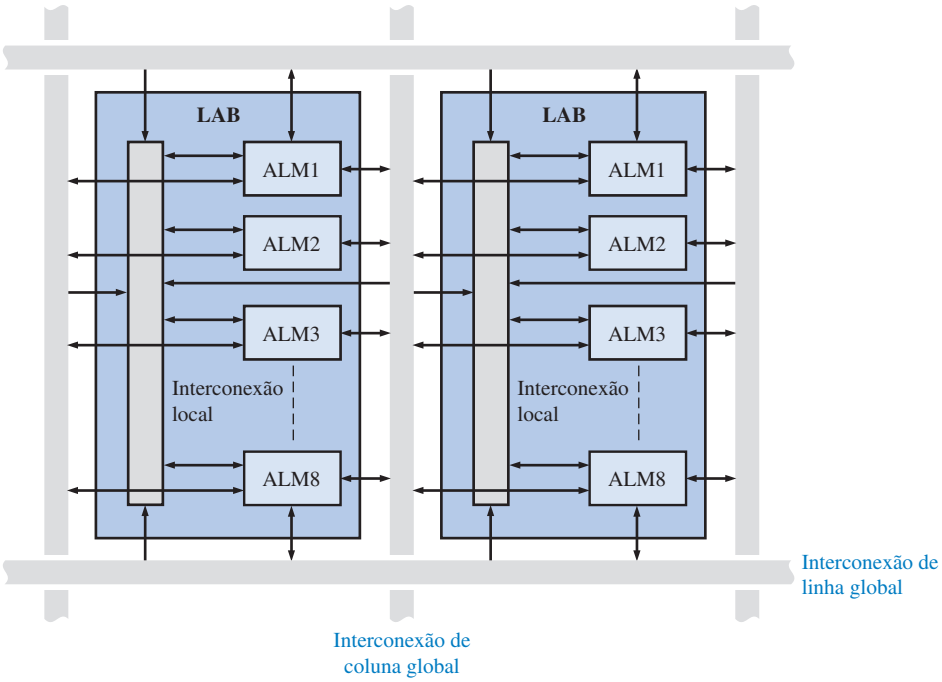
Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir a arquitetura básica de um típico FPGA da família Stratix II ■ Explicar como termos-produto são gerados em FPGAs ■ Discutir funções embutidas

O Bloco de Arranjo Lógico (LAB)

A Figura 11–31 mostra o diagrama em bloco de um FPGA genérico; a arquitetura da família Stratix II e outras famílias da Altera são similares. Elas têm a estrutura LUT clássica para os módulos lógicos, denominados de módulos lógicos adaptáveis (ALMs – *adaptive logic modules*) pela Altera, os quais produzem funções de soma-de-produtos. A Altera também denomina os blocos de arranjo configuráveis, mostrados no dispositivo genérico, de blocos de arranjo lógico (LABs). A densidade varia de quase 2000 LABs a mais de 22.000 LABs, dependendo do dispositivos em particular na família; sendo que cada LAB tem oito ALMs. O tamanho dos encapsulamentos varia de 341 pinos a 1173 pinos. Dispositivos que necessitam de tensões alimentação de 1,2 V, 1,5 V e 2,5 V são tipicamente comercializados. A família Stratix II de FPGAs usa tecnologia de processo baseada em SRAM.

A Figura 11–38 mostra um diagrama simplificado da estrutura de um LAB Stratix II. Cada LAB consiste de oito ALMs; múltiplos LABs são interconectados via interconexões de linhas e colunas globais. A interconexão local conecta os ALMs dentro de cada LAB.



▶ **FIGURA 11–38**

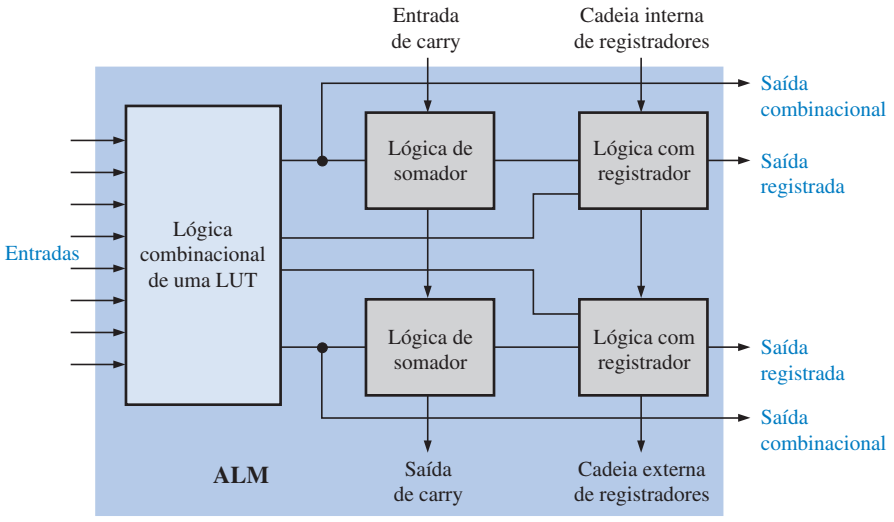
Diagrama simplificado da estrutura de um LAB (bloco de arranjo lógico) em uma FPGA da família Stratix II. Os ALMs são módulos lógicos adaptáveis.

O Módulo Lógico Adaptável

O bloco ALM é a unidade de projeto básica no FPGA da família Stratix II. Cada ALM contém uma seção de lógica combinacional baseada em LUT e uma lógica associada que pode ser programada por duas saídas de lógica combinacional ou registradas. Além disso, o ALM tem lógica de somador, flip-flops e outras lógicas que permitem a implementação de funções aritméticas, de contadores e de registradores de deslocamento. A Figura 11–39 mostra um diagrama simplificado de um ALM da família Stratix II.

Modos de Operação de um ALM Um ALM pode ser programado nos seguintes modos de operação:

- Modo normal
- Modo LUT estendida
- Modo aritmético
- Modo aritmético compartilhado

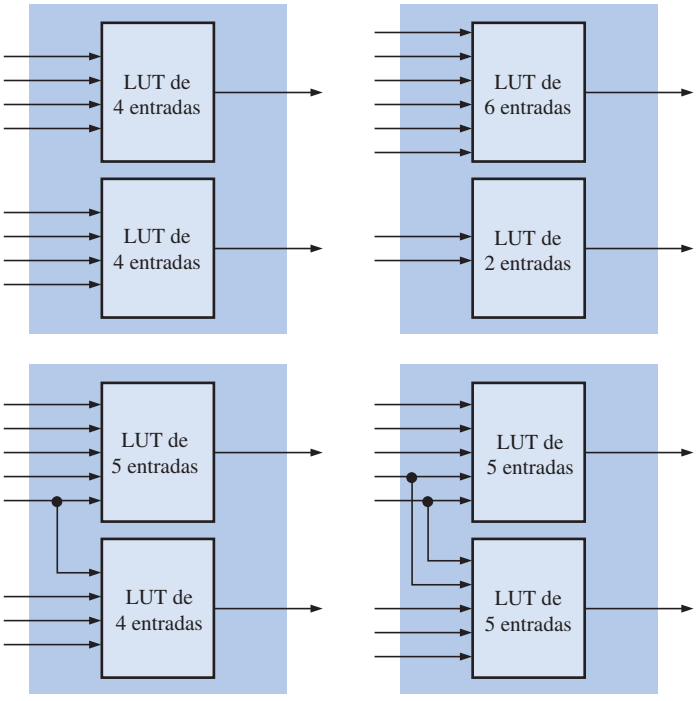


▶ **FIGURA 11–39**

Diagrama simplificado de um módulo lógico adaptável (ALM) da família Stratix II.

Além desses quatro modos, um ALM pode ser utilizado como uma cadeia de registradores para criar contadores e registradores de deslocamento. Nesta seção, discutiremos o modo normal e o modo estendido da LUT.

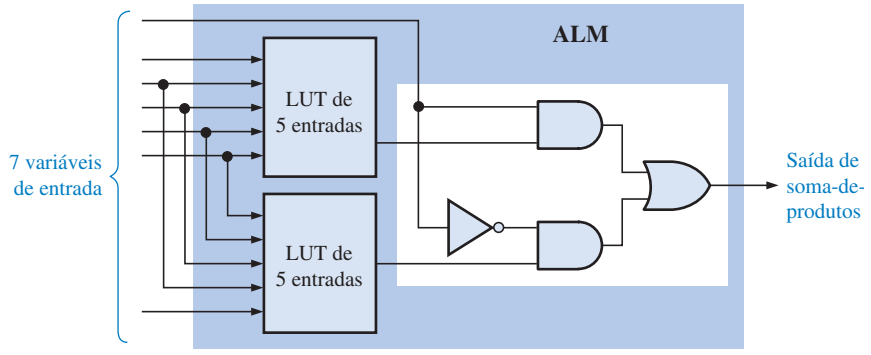
O *modo normal* é usado principalmente para gerar funções lógicas combinacionais. Um ALM pode implementar uma ou duas funções combinacionais de saída com as suas duas LUTs. A Figura 11–40 mostra exemplos de configurações de quatro LUTs. Com um ALM pode ser implementado duas funções de soma-de-produtos, cada uma com quatro variáveis ou menos, sem o compartilhamento de entradas. Por exemplo, podemos ter duas funções de quatro variáveis, uma função de quatro variáveis e outra de 3 variáveis ou ainda duas funções de três variáveis. Por meio do compartilhamento de entradas, podemos ter qualquer combinação de um total de oito entradas sendo no máximo seis entradas para cada LUT. No modo normal, estamos limitados a funções de soma-de-produtos de seis variáveis.



◀ **FIGURA 11–40**

Exemplos de configurações de LUT possíveis em um módulo lógico adaptável (ALM) no modo normal.

O *modo LUT estendida* permite a expansão para uma função de 7 variáveis, conforme ilustrado na Figura 11–41. O multiplexador formado pelo circuito AND-OR com uma entrada complementada é parte da lógica dedicada em um ALM.

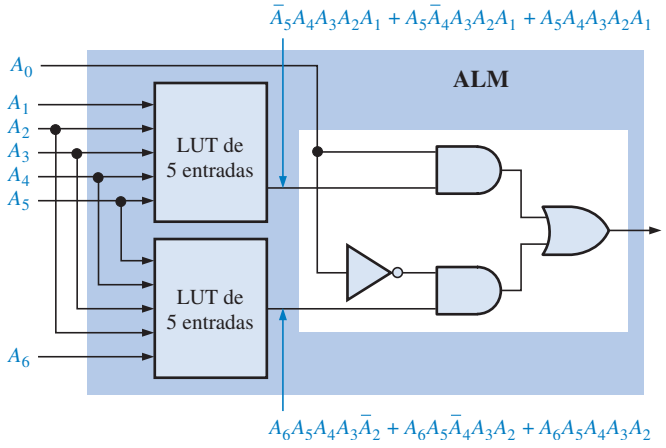


◀ **FIGURA 11–41**

Expansão de um ALM para produzir uma função de soma-de-produtos de 7 variáveis no modo LUT estendida.

EXEMPLO 11–4

Um ALM em um FPGA Stratix II é configurado no modo LUT estendida, como mostra a Figura 11–42. Para as saídas da LUT específica mostrada, determine a saída de soma-de-produtos final.



▶ FIGURA 11–42

Solução A expressão da saída de soma-de-produtos é:

$$\overline{A}_5A_4A_3A_2A_1A_0 + A_5\overline{A}_4A_3\,A_2A_1A_0 + A_5A_4A_3A_2A_1A_0 + A_6A_5A_4A_3\overline{A}_2\overline{A}_0 + A_6A_5\overline{A}_4A_3A_2\overline{A}_0 + A_6A_5A_4A_3A_2\overline{A}_0$$

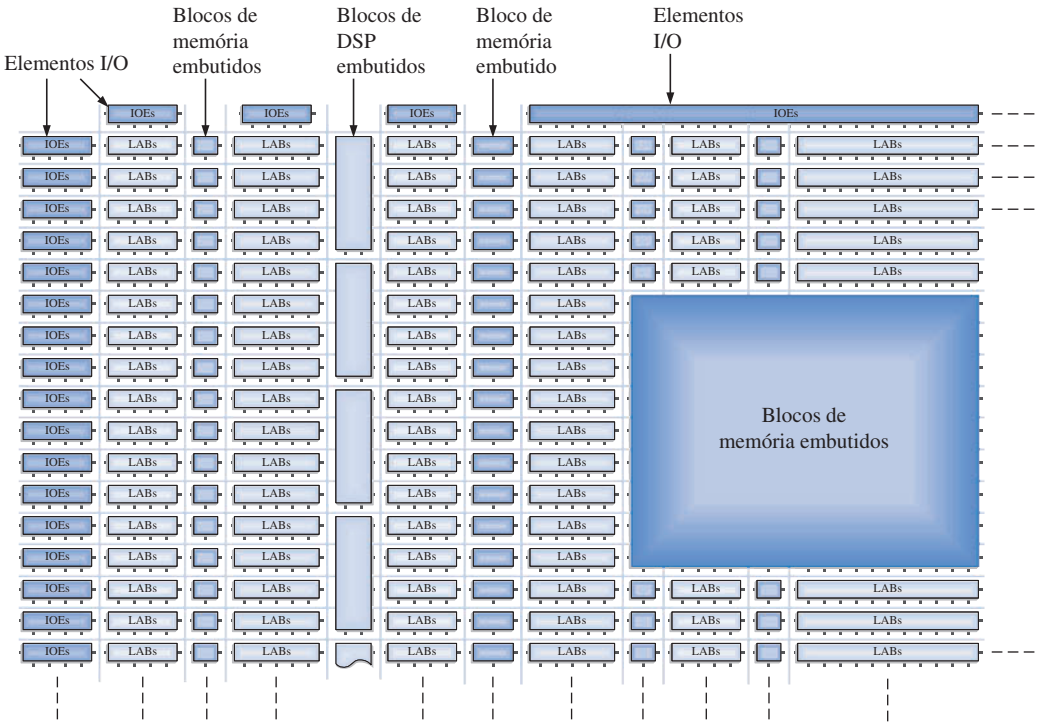
Problema relacionado Mostre um ALM configurado no modo normal para produzir uma função de soma-de-produtos com cinco termos-produto a partir de uma LUT e três termos-produto a partir de outra LUT.

Funções Embutidas

Um diagrama em bloco geral do FPGA Stratix II é mostrado na Figura 11–43. O FPGA contém funções de memória embutidas bem como funções de processamento de sinais digitais (DSP). As funções DSP, tal como os filtros digitais, são normalmente usadas em muitos sistemas. Conforme podemos ver no diagrama em bloco, os blocos embutidos são arranjados ao longo da matriz de interconexões e os elementos de entrada/saída (IOEs) são colocados em torno do perímetro da FPGA.

SEÇÃO 11–6 REVISÃO

1. Qual é a unidade de projeto lógico básica na FPGA da família Stratix II?
2. Quantos ALMs existem em um LAB?
3. O que é usado para produzir funções lógicas combinacionais em um ALM?
4. Quantas funções de soma-de-produtos um ALM pode produzir?
5. Cite os dois tipos de funções embutidas na família Stratix II.



▲ **FIGURA 11–43**

Diagrama em bloco Stratix II.

11-7 FPGAs XILINX

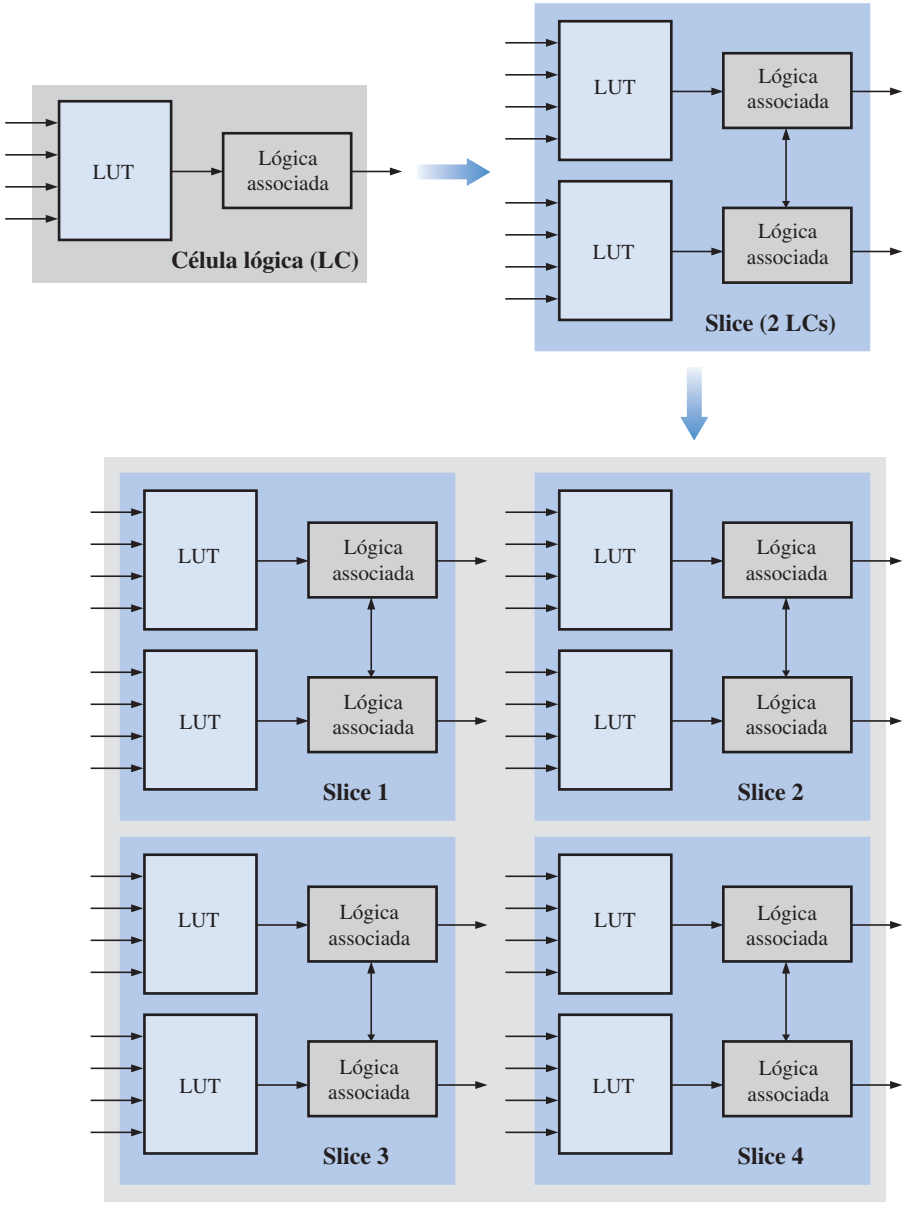
A Xilinx tem duas importantes linhas de FPGAs, a Spartan e a Virtex, sendo que existem diferentes famílias dentro de cada linha. Como exemplos citamos a Spartan 3 e Spartan IIE, Virtex 4, Virtex II, Virtex II Pro e Virtex II Pro X. A Xilinx classifica a Virtex II, Virtex II Pro e Virtex II Pro X como plataformas FPGAs porque elas têm funções embutidas, tal como memórias, processadores, transceptores entre outros núcleos IP rígidos e flexíveis. As famílias FPGAs diferem geralmente nos parâmetros de densidade e desempenho. A maioria dos dispositivos Xilinx tem uma arquitetura FPGA tradicional; entretanto, o Virtex II Pro X tem o que é denominado Bloco Modular Específico de Aplicação, ASMBL™ (pronunciado *assemble*), uma arquitetura com mais de um bilhão de transistores em um único dispositivo.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever uma FPGA típica da família Virtex ■ Discutir a arquitetura Virtex básica ■ Explicar como termos-produto são gerados por um FPGA ■ Descrever a arquitetura ASMBL

Blocos Lógicos Configuráveis

A área lógica configurável (denominada estrutura FPGA) da maioria das FPGAs Xilinx é dividida em blocos lógicos configuráveis (CLBs) sendo que cada CLB contém múltiplas unidades lógicas básicas denominadas de células lógicas (LCs). Cada célula lógica é baseada na lógica LUT de quatro entradas tradicional mais uma lógica adicional e um flip-flop. Uma LUT de 4 entradas pode produzir de um termo-produto a uma função de termos-produto com dezesseis termos. Duas células lógicas idênticas são denominadas de *slice* (*fatia*). A Figura 11–44 ilustra os níveis da lógica configurável da célula lógica do CLB. As densidades variam de cerca de 2000 a mais de 74.000 células lógicas num único dispositivo Virtex.

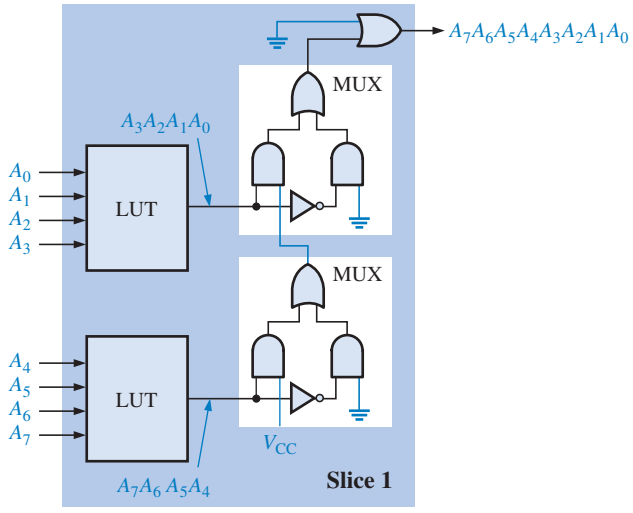


▲ **FIGURA 11–44**

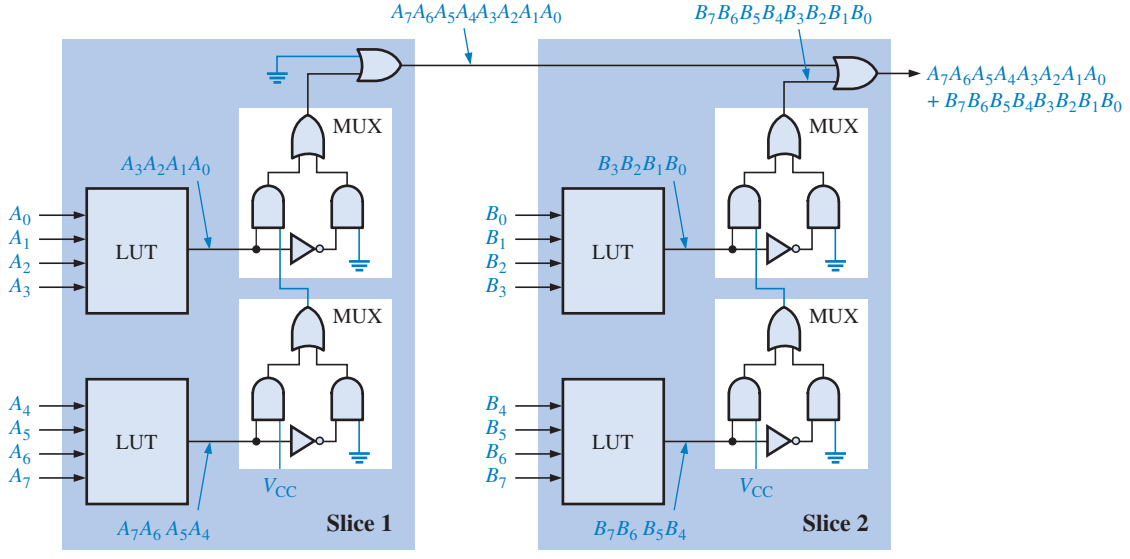
CLB simplificado em uma FPGA Virtex.

Cadeias em Cascata de Soma-de-Produtos

Uma *slice* simplificada (duas células lógicas) da lógica de uma cadeia em cascata é mostrada na Figura 11–45. Existe um multiplexador (MUX) dedicado dentro da lógica associada de cada LC, que pode ser usado na cadeia em cascata, bem como uma porta OR dedicada dentro de cada *slice*. A Figura 11–45(a) mostra um exemplo de como uma *slice* simplificada em um CLB pode ser configurada como uma porta AND para produzir um termo-produto de 8 variáveis. Duas *slices* podem ser configuradas para produzir uma função de soma-de-produtos com dois termos-produto de 8 variáveis, conforme mostra a parte (b) da figura. Um CLB completo de quatro *slices* pode ser configurado numa cadeia em cascata para produzir uma função de soma-de-produtos com quatro termos-produto de 8 variáveis, conforme mostra a parte (c) da mesma figura. Uma posterior expansão de soma-de-produtos pode ser feita usando CLBs adicionais.



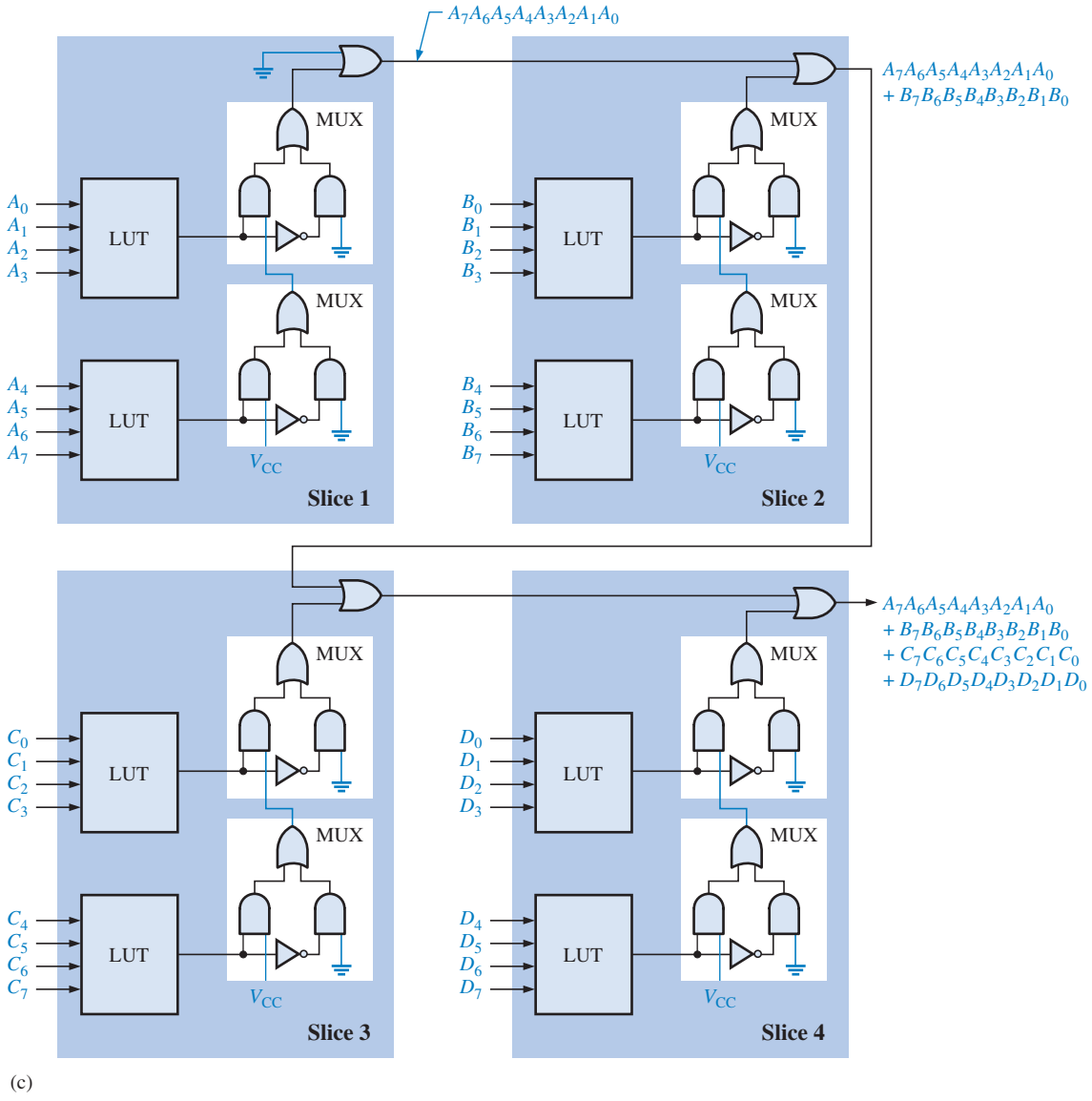
(a)



(b)

▲ **FIGURA 11–45**

Exemplo do uso de cadeias em cascata para a expansão de uma função de soma-de-produtos.



(c)

▲ **FIGURA 11–45**

Continuação.

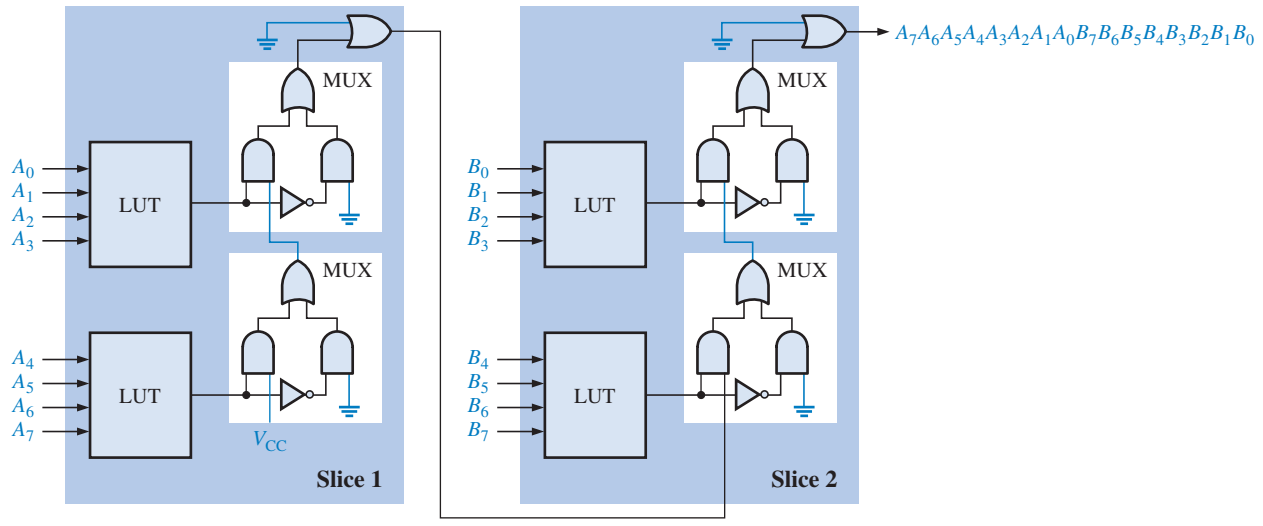
EXEMPLO 11–5

Mostre como uma porta AND de 16 entradas pode ser implementada em um CLB.

Solução Duas *slices* configuradas como mostra a Figura 11–46 resultam em uma porta AND de 16 entradas.

Problema relacionado Mostre como as duas *slices* vistas na Figura 11–46 poderiam ser configuradas para produzir a seguinte função de soma-de-produtos:

$A_7A_6A_5A_4 + A_3A_2A_1A_0 + B_7B_6B_5B_4 + B_3B_2B_1B_0$.



▲ **FIGURA 11–46**

Implementação de uma porta AND de 16 entradas para produzir um termo-produto com dezesseis variáveis.

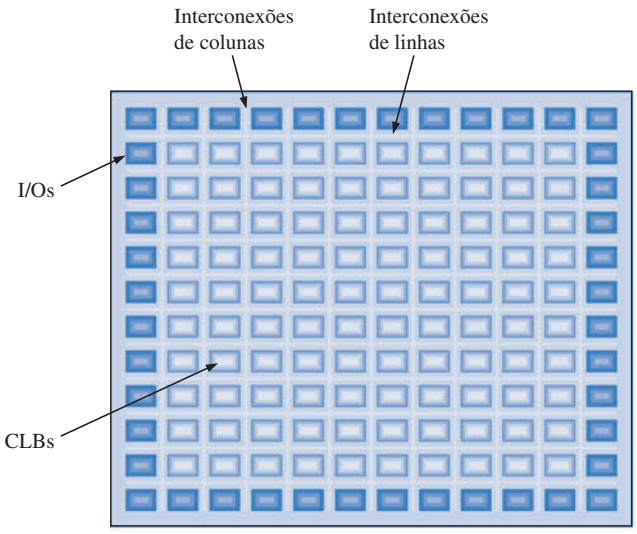
Arquitetura FPGA Tradicional *versus* a Arquitetura ASMBL

Conforme estudamos, a arquitetura FPGA tradicional aparece como um arranjo de blocos lógicos (CLBs ou LABs) envoltos por células de entrada/saída configuráveis. A quantidade de CLBs configuráveis em um FPGA depende do número de elementos de I/O que podem ser colocados fisicamente em torno do perímetro. Quando os núcleos IP são acrescentados, o tamanho físico de uma FPGA tem que ser aumentado para manter a lógica configurável necessária e aumentar o número de I/Os. Esse conceito geral é ilustrado na Figura 11–47.

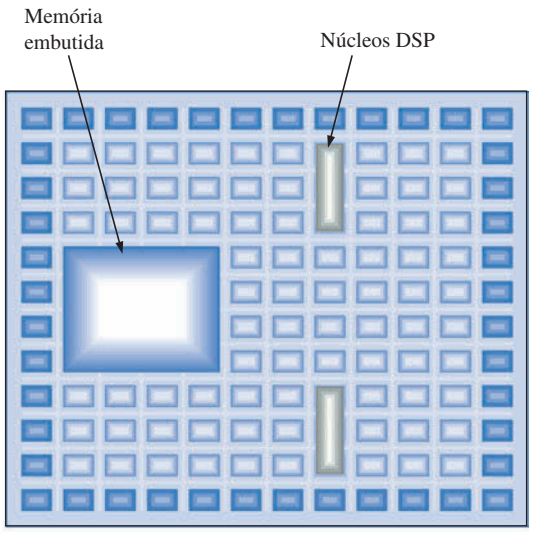
Quanto mais complexa a lógica em uma FPGA, mais I/Os são necessários. A relação de dependência entre lógica e I/Os resulta no aumento do tamanho e do custo do chip. Além disso, um outro problema com os FPGAs de plataforma é que quando funções de núcleo IP embutidas adicionais são necessárias, um reprojeto parcial ou maior no leiaute do chip pode ser necessário, o que é um processo de alto custo.

A Arquitetura ASMBL A Xilinx criou uma abordagem flexível para FPGAs de plataforma em dispositivos Virtex II Pro X para superar algumas das limitações da arquitetura tradicional. A arquitetura do bloco modular específico de aplicação (ASMBL) é uma estrutura baseada em colunas em vez de ser baseada em linhas/colunas. Os I/Os são posicionados em colunas intercaladas em vez de posicionados ao longo do perímetro, assim o número de I/Os pode ser aumentado sem ter que aumentar o tamanho do chip. Cada coluna é essencialmente uma tira de circuito lógico que pode ser substituída por um outro tipo de circuito sem a necessidade de reprojetar o leiaute do chip. Como exemplos de tiras de circuitos lógicos temos os blocos lógicos configuráveis (CLBs), os blocos de I/O (IOBs), memória e núcleos IP rígidos e flexíveis tal como um DSP e um processador.

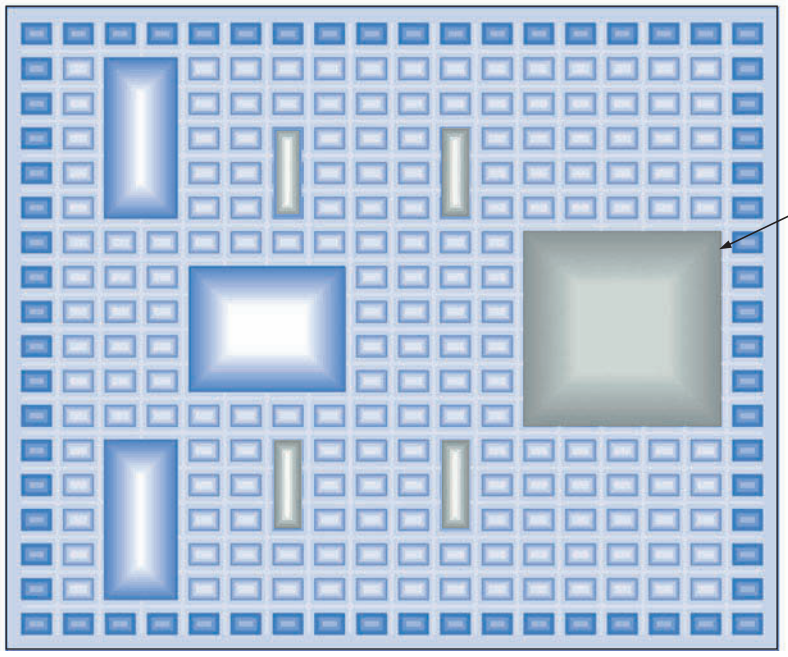
Diversas quantidades de cada tipo de tira de circuito lógico podem ser combinadas para atingir os requisitos de uma aplicação específica. Por exemplo, na configuração mais simples, poderíamos combinar tiras de circuito CLB e blocos de I/O, conforme ilustrado na Figura 11–48(a). Poderia se usar mais tiras de um circuito e menos de outro dependendo os requisitos da aplicação. Se necessitarmos de mais memória, uma ou mais tiras de circuito CLB poderiam ser substituídas, conforme indicado na parte (b) da figura. Caso a área de aplicação específica for processamento de sinais digitais, podemos acrescentar um núcleo IP DSP combinado com memória, como mostra a parte (c) da figura. A parte (d) da figura mostra a adição de núcleos no processador.



(a) FPGA com lógica completamente configurável.



(b) FPGA de mesmo tamanho com memória e núcleos IP (DSP) embutidos resultando em menos CLBs e com limitação de I/Os ao longo do perímetro.



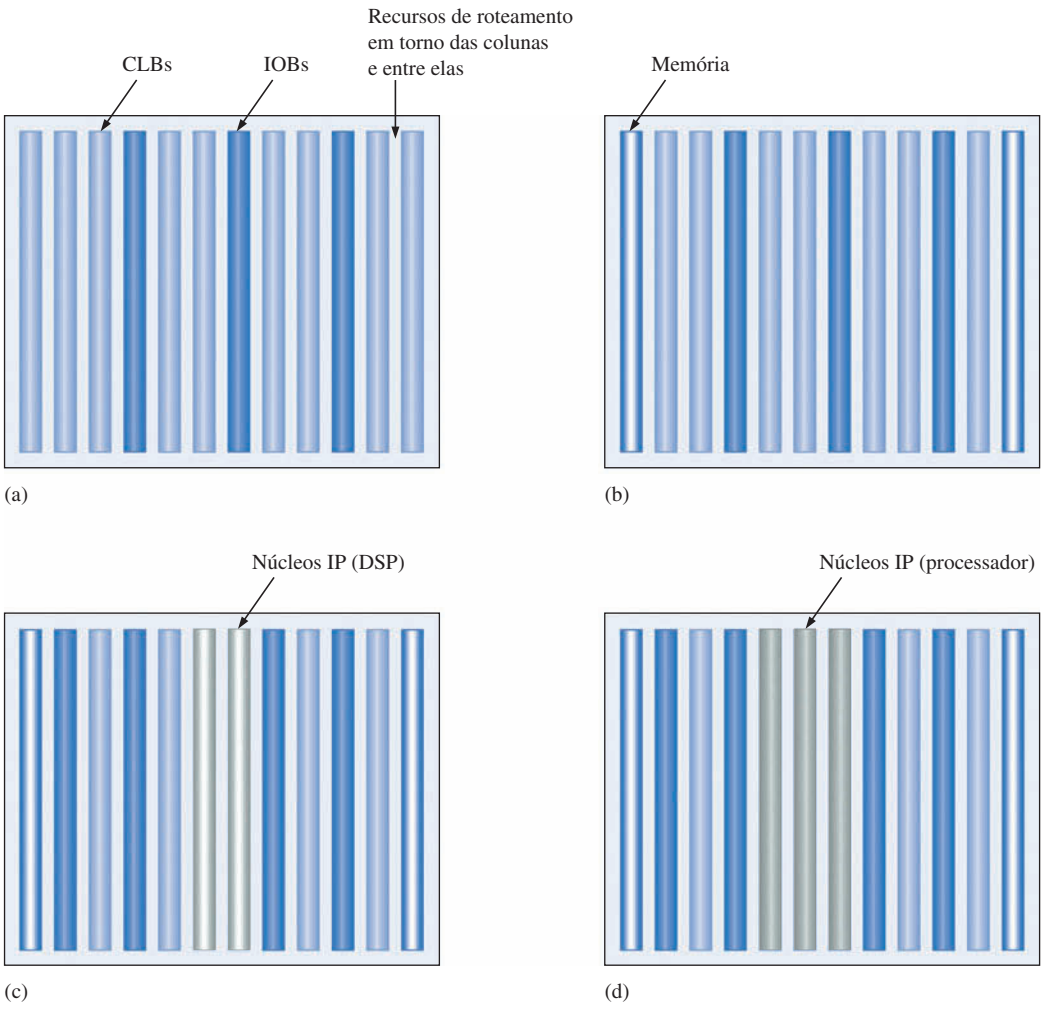
(c) Um tamanho físico maior torna-se necessário para um FPGA com mais memória embutida, núcleos DSP adicionais e um núcleo de processador.

▲ FIGURA 11–47

Com funções IP embutidas (memória, DSP e processador), temos menos lógica configurável e/ou chips fisicamente maiores devido ao aumento de I/Os.

SEÇÃO 11–7 REVISÃO

1. Em que consiste um CLB em um FPGA da Xilinx?
2. Em que consiste um LC?
3. Descreva o que é uma *slice* em um FPGA da Xilinx.
4. O que é uma cadeia em cascata de soma-de-produtos?
5. O que significa ASMBL?



▲ **FIGURA 11–48**

Conceito básico da arquitetura FPGA com plataforma ASMBL.

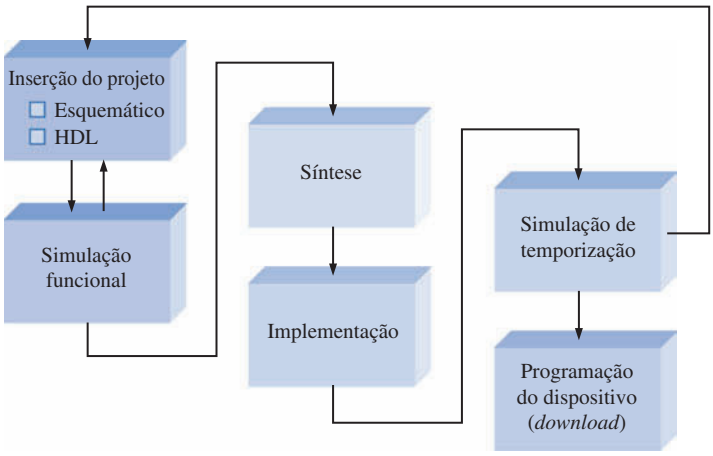
11-8 SOFTWARE PARA LÓGICA PROGRAMÁVEL

O resultado apresentado pela lógica programável depende da combinação dos componentes de hardware e software como uma unidade funcional. Todos os fabricantes de SPLDs, CPLDs e FPGAs fornecem um suporte de software para cada dispositivo. Esses pacotes de software são conhecidos como projeto auxiliado por computador (CAD – *computer aided design*). A programação de PLD foi apresentada inicialmente no Capítulo 1 e abordada posteriormente no Capítulo 3. Nesta seção, o software para a lógica programável é apresentado de forma geral.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar o processo de programação em termos do fluxo do projeto ■ Descrever as fases de inserção do projeto ■ Descrever a fase da simulação funcional ■ Descrever a fase de síntese ■ Descrever a fase de implementação ■ Descrever a fase da simulação temporal ■ Descrever a fase de download

O processo de programação é geralmente conhecido como **fluxo de projeto**. Um diagrama de fluxo de projeto básico para a implementação de um projeto lógico em um dispositivo programável é mostrado na Figura 11–49. Os pacotes de software mais específicos incorporam esses elementos de uma forma ou de outra e os processa automaticamente. O dispositivo a ser programado é geralmente conhecido como **dispositivo destino**.



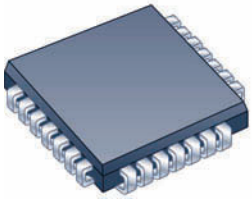
► **FIGURA 11–49**

Diagrama de fluxo de um projeto geral para a programação de SPLD, CPLD ou FPGA.

Para realizar a programação de um dispositivo são necessários quatro itens: um computador, um software de desenvolvimento, um dispositivo lógico programável (SPLD, CPLD ou FPGA) e um sistema para conexão do dispositivo ao computador. Esses itens essenciais são ilustrados na Figura 11–50. A parte (a) da figura mostra um computador que atende aos requisitos do sistema em

(a) Computador

(b) Software (CD ou download do website)



(c) Dispositivo

(d) Hardware de programação (equipamento de programação ou placa de desenvolvimento com cabo para conexão à porta de I/O do computador)

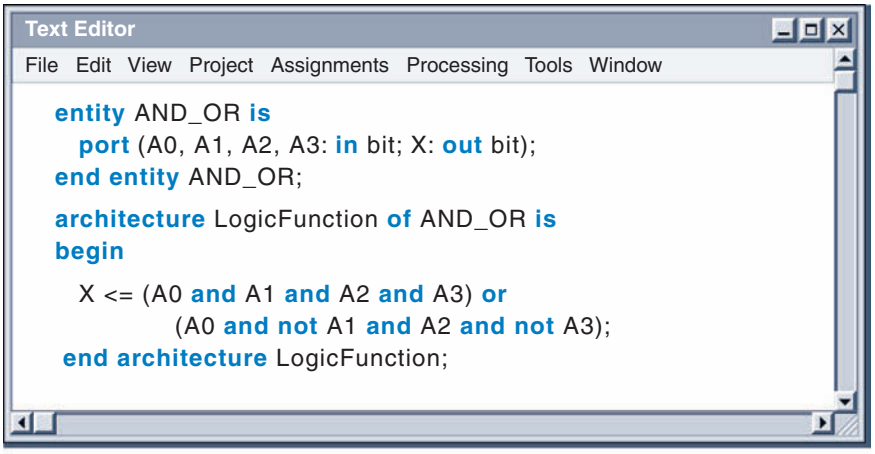
▲ **FIGURA 11–50**

Elementos essenciais para a programação de um SPLD, CPLD ou FPGA.

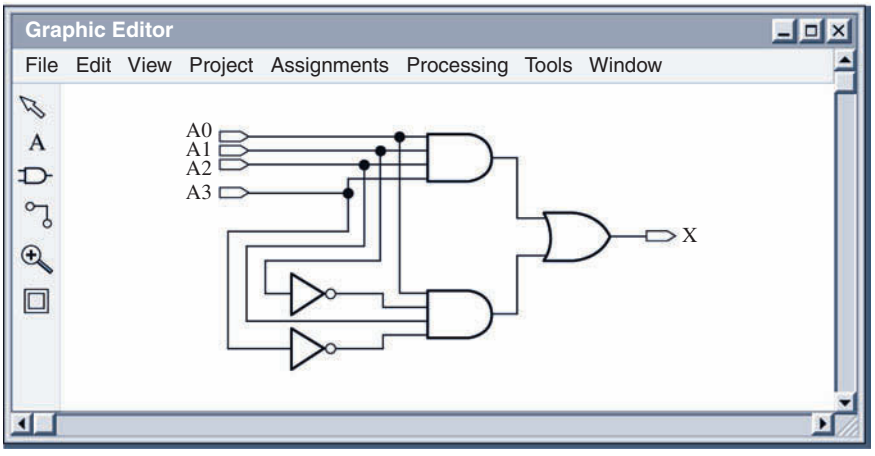
termos do software em particular a ser usado. A parte (b) da figura mostra o software adquirido do fabricante do dispositivo na forma de CD ou por meio de download feito a partir do website do mesmo. A maioria dos fabricantes fornece softwares livres que podem ser obtidos por download e usados durante um tempo limitado. A parte (c) da figura mostra um dispositivo lógico programável. Já a parte (d) ilustra dois meios para conectar fisicamente o dispositivo ao computador via cabo usando um equipamento de programação, no qual o dispositivo é inserido, ou então usando uma placa de desenvolvimento, na qual o dispositivo é montado. Após o software ser instalado no computador, temos que nos familiarizar com as ferramentas de software específicas antes de tentar conectar e programar um dispositivo. Esse processo de aprendizagem requer esforço e tempo.

Inserção do Projeto

Considere que temos um circuito lógico o qual desejamos implementar num dispositivo programável. Podemos inserir o projeto no computador em uma dentre duas formas: **inserção via esquemático** ou **inserção via texto**. Para usar esta última, temos que estar familiarizados com uma linguagem de descrição de hardware (HDL), tal como VHDL, Verilog, ABEL ou AHDL. A maioria dos fabricantes de lógica programável oferece pacotes de software que suportam VHDL e Verilog porque elas são HDLs padrão. Alguns suportam também linguagens como ABEL, AHDL ou outras HDLs proprietárias. A inserção via esquemático basicamente nos permite colocar símbolos de portas lógicas e outras funções lógicas a partir de uma biblioteca visualizada na tela conectando esses blocos conforme os requisitos do projeto. O conhecimento de uma HDL não é necessário para a inserção via esquemático. A Figura 11–51 ilustra esses dois tipos de inserção de forma genérica para um circuito lógico AND-OR simples.



(a) Inserção textual usando VHDL para descrever um circuito lógico AND-OR.



(b) Inserção esquemática do mesmo circuito lógico AND-OR inserido em (a).

◀ **FIGURA 11–51**
Exemplos de telas de inserção textual e esquemática.

Construindo um Esquema Lógico Quando inserimos um circuito lógico completo na tela, ele é denominado de esquema "plano". Circuitos lógicos mais complexos podem ser difíceis de encaixar na tela. Nesse caso, podemos inserir o circuito lógico em segmentos, salvando cada segmento como um bloco na forma de símbolo, e então conectando os blocos na forma de símbolos para formar um circuito completo. Essa abordagem é denominada de hierárquica.

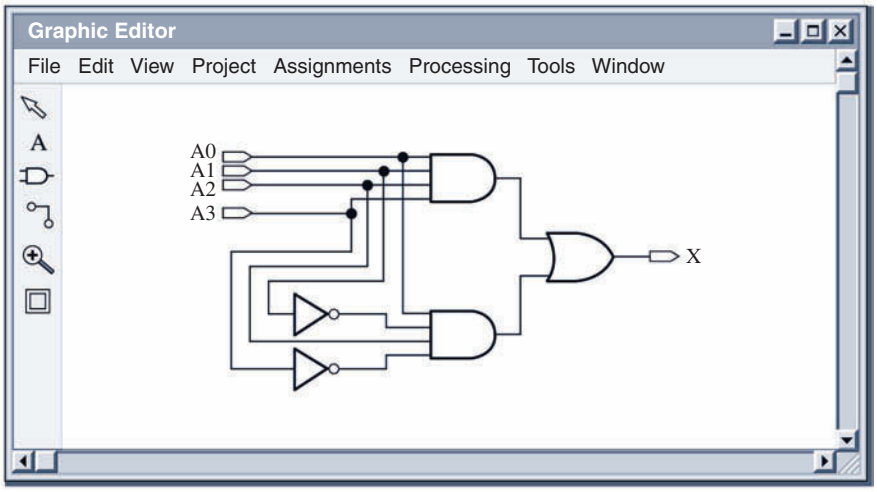
Como exemplo, considere que necessitamos de um circuito que produza a seguinte expressão de soma-de-produtos:

$$Z = (A_3A_2A_1A_0 + \overline{A}_3A_2\overline{A}_1A_0) + (A_3\overline{A}_2\overline{A}_1A_0 + A_3\overline{A}_2A_1\overline{A}_0 + \overline{A}_3A_2A_1\overline{A}_0)$$

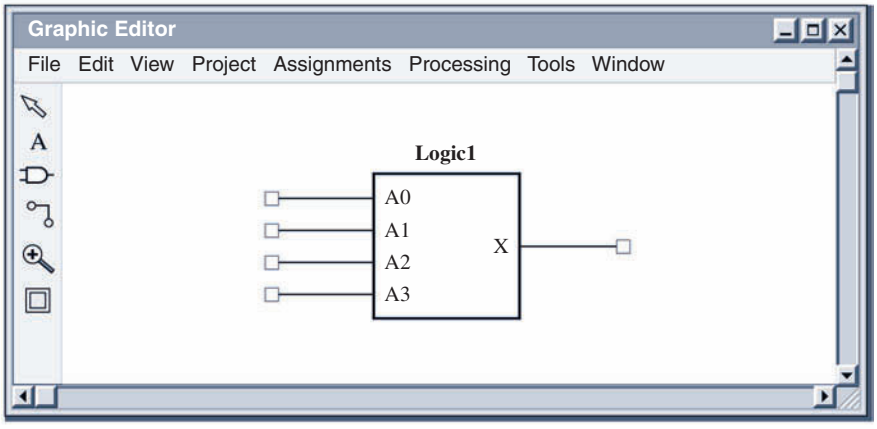
Vamos usar a abordagem hierárquica e criar o circuito lógico para cada um dos dois termos entre parênteses na equação acima; reduza cada circuito lógico a um único bloco na forma de um símbolo gráfico e, em seguida, quando os dois circuitos estiverem completos, coloque-os na tela e conecte suas saídas numa porta OR. Isso está ilustrado nas cinco partes da Figura 11–52. O circuito completo poderia ser inserido na tela de uma só vez, mas a abordagem hierárquica é útil quando o circuito lógico for grande e tiver que ser dividido em partes.

Na parte (e) da Figura 11–52, o circuito lógico poderia ser reduzido a um outro símbolo lógico e usado em um projeto lógico ainda maior; ou poderia ser salvo e reutilizado em outros projetos, conforme ilustrado na Figura 11–53.

Após o circuito lógico ser inserido como um esquema, um programa aplicativo denominado **compilador** controla as diversas ferramentas de CAD que processa o esquema e produz uma implementação para o dispositivo destino.



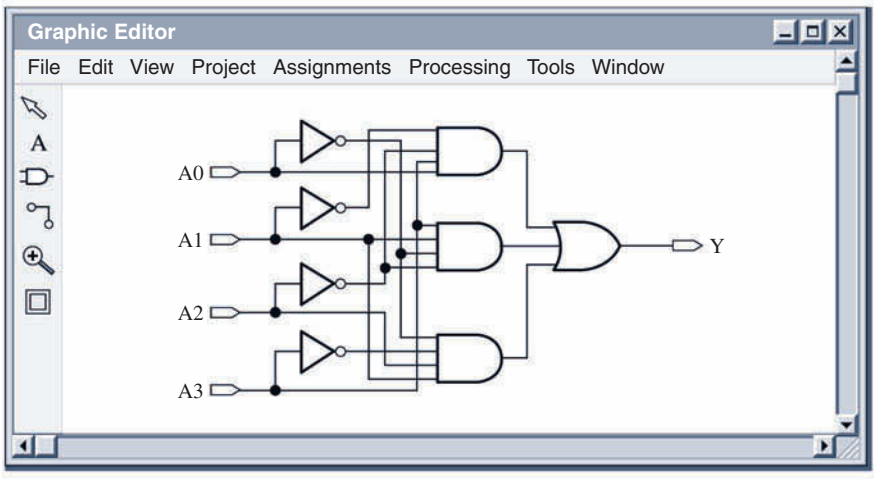
(a) Insira os dois termos-produto como um circuito lógico AND-OR.



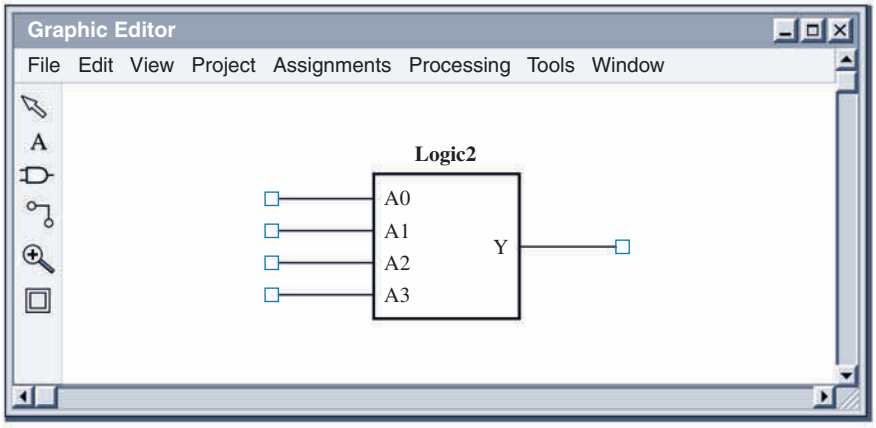
(b) Reduza o circuito lógico AND-OR a um símbolo de um bloco definido como Logic1.

► **FIGURA 11–52**

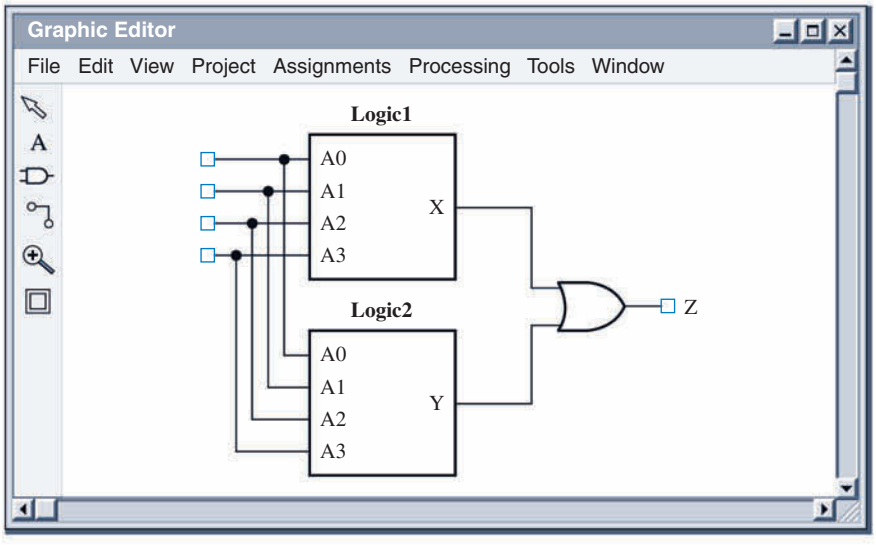
Exemplo da criação de um circuito lógico em segmentos, combinando em seguida os segmentos.



(c) Insira os três termos-produto como um circuito lógico AND-OR.



(d) Reduza o circuito a um símbolo de um bloco definido como Logic2.



(e) Combine Logic1 e Logic2 com uma porta OR e conecte as entradas comuns.

◀ **FIGURA 11–52**
Continuação.

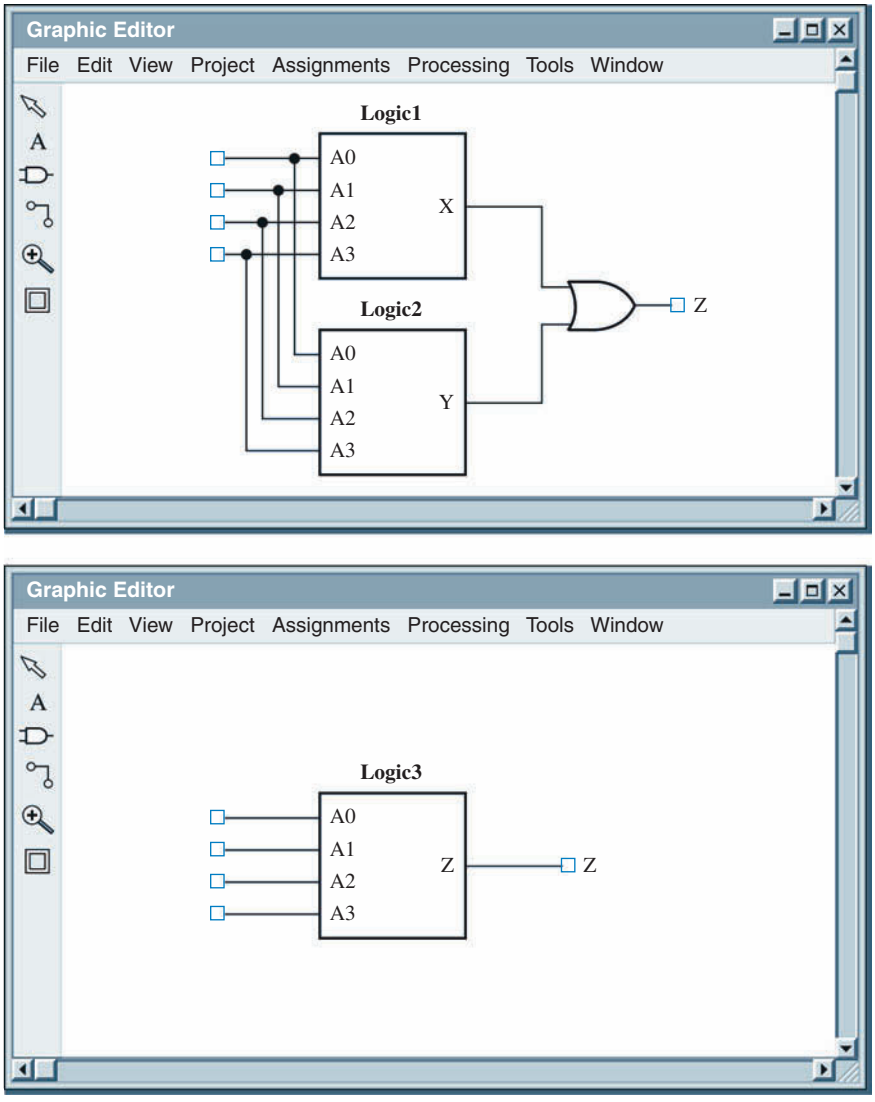


FIGURA 11–53

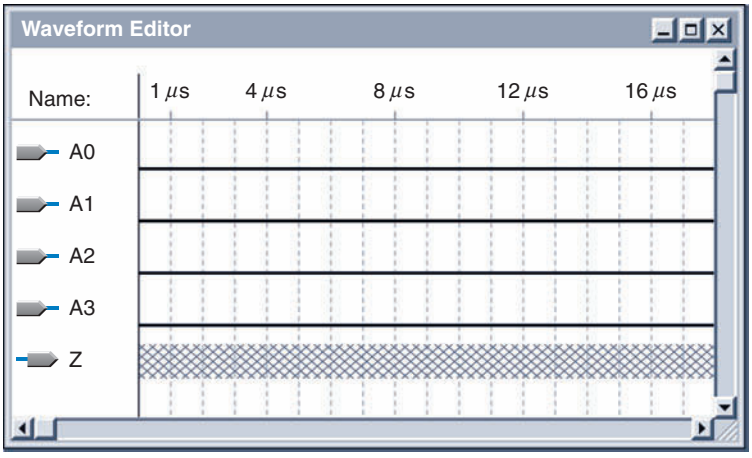
O circuito lógico com dois blocos lógicos e uma porta OR é reduzido a um outro bloco lógico (Logic3).

Simulação Funcional

A finalidade da simulação funcional no fluxo do projeto é garantir que o projeto inserido funciona como deveria funcionar, em termos de sua operação lógica, antes de sintetizá-lo no projeto de hardware. Basicamente, após o circuito lógico ser compilado, ele pode então ser simulado aplicando as formas de onda de entrada e verificando a saída para todas as combinações de entrada possíveis usando um editor de forma de onda.

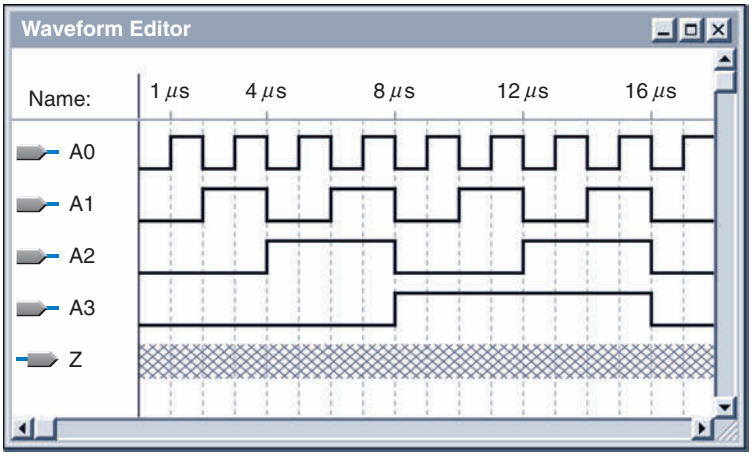
O editor de forma de onda nos permite selecionar os nós (entradas e saídas) que queremos testar. Os nomes das entradas e saídas selecionadas aparecem na tela do *Waveform Editor* (editor de forma de onda) juntamente com um símbolo ou outra designação que identifique cada uma das entradas ou saídas, conforme mostra a Figura 11–54. Inicialmente, todas as quatro entradas iniciam em 0 e a trama cruzada indica que a saída é desconhecida. Podemos selecionar os intervalos de tempo para visualização.

Em seguida, criamos cada forma de onda de entrada inserindo um nível 1 ou nível 0 para cada intervalo de tempo (intervalo entre as linhas pontilhadas na Figura 11–55). Isso geralmente é feito através de um ponto, click, e da seleção com o mouse, dependendo do software específico. Nesse caso em particular, criamos as formas de onda de forma que todas as 16 combinações possíveis de 4 entradas sejam representadas. A Figura 11–55 mostra as formas de onda de entrada (A0, A1, A2, A3) conforme sejam especificadas.



◄ **FIGURA 11–54**

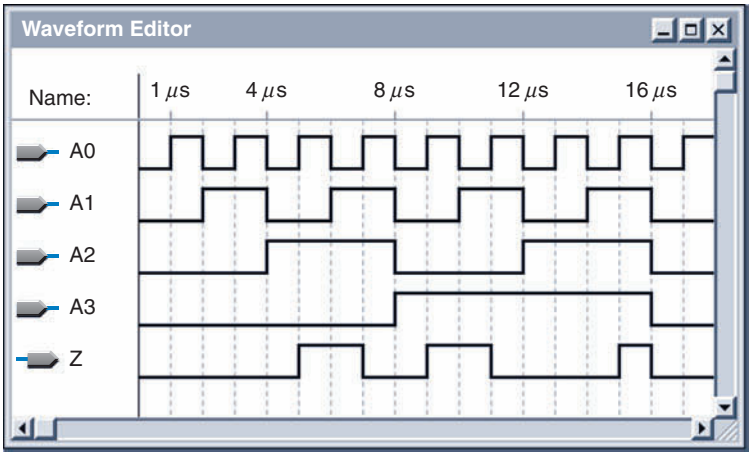
Tela de um Editor de Forma de Onda genérico com os nomes das entradas e saídas especificados para o circuito inserido na Figura 11–53.



◄ **FIGURA 11–55**

Formas de onda de entrada especificadas na tela de um Editor de Formas de Onda.

Após termos especificado as formas de onda de entrada, geralmente abre-se uma janela de controle da simulação, permitindo ajustar os tempos inicial e final para a simulação e especificar os intervalos de tempo a serem apresentados. Quando a simulação inicia, a forma de onda de saída Z é mostrada no Editor de Forma de Onda, conforme ilustra a Figura 11–56. Isso nos permite verificar se o projeto funciona corretamente ou não. Nesse caso, a forma de onda de saída é correta para as formas de onda de entrada selecionadas. Uma forma de onda de saída incorreta indicaria uma falha na funcionalidade do circuito lógico; assim teríamos que voltar atrás, verificando o projeto original e então reinserindo o projeto revisado.



◄ **FIGURA 11–56**

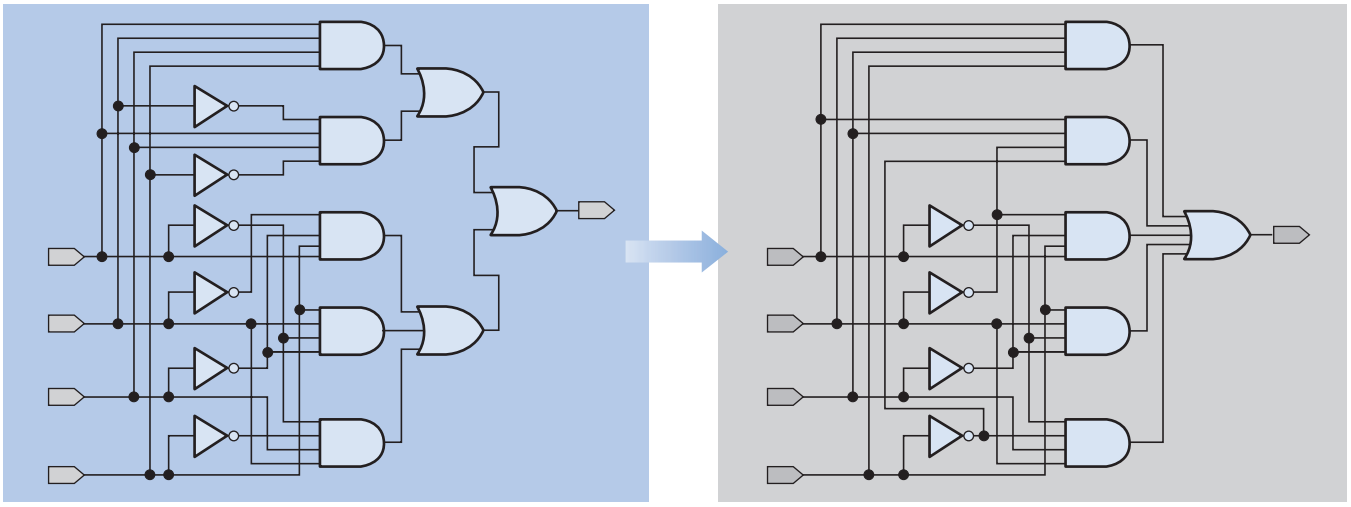
Após a simulação funcional ser executada, a forma de onda de saída deve indicar que o circuito lógico funciona corretamente.

Síntese

Uma vez inserido o projeto e simulado funcionalmente para verificar que a operação do circuito lógico está correta, o compilador passa automaticamente por várias fases preparando o projeto para ser transferido (operação de *download*) para o dispositivo destino. Durante a fase de síntese do fluxo do projeto, este é otimizado em termos da minimização do número e portas, substituindo uns elementos lógicos por outros que podem realizar a mesma função de forma mais eficiente, eliminando qualquer lógica redundante. O resultado final a partir da fase de síntese é uma *netlist* que descreve a versão otimizada do circuito lógico.

Netlist Uma ***netlist*** é basicamente uma lista de conectividade que descreve os componentes e como eles são interconectados. Geralmente, uma *netlist* contém referências para descrição dos componentes ou elementos usados. Cada vez que um componente, tal como uma porta lógica, é usado numa *netlist*, ele é denominado de *instância*. Cada instância tem uma definição que lista as conexões que podem ser feitas para um determinado tipo de componente bem como algumas propriedades básicas. Esses pontos de conexão são chamados de *portas* ou *pinos*. Geralmente, cada instância tem um nome único; por exemplo, se temos duas instâncias de portas AND, uma pode ser "and1" e a outra "and2". Exceto pelos nomes, elas podem ser idênticas. As conexões são os "fios" que interconectam pontos no circuito. *Netlists* baseadas em conexões geralmente descrevem todas as instâncias e seus atributos, em seguida descrevem cada conexão e especificam a qual porta, de entrada/saída, cada instância é conectada.

O circuito lógico AND-OR que inserimos na fase de projeto, mostrado na Figura 11–57(a), poderia resultar no circuito otimizado mostrado na Figura 11–57(b). Nessa ilustração, o compilador removeu as três portas OR substituindo-as por uma porta OR de 5 entradas. Além disso, dois inversores redundantes foram removidos.



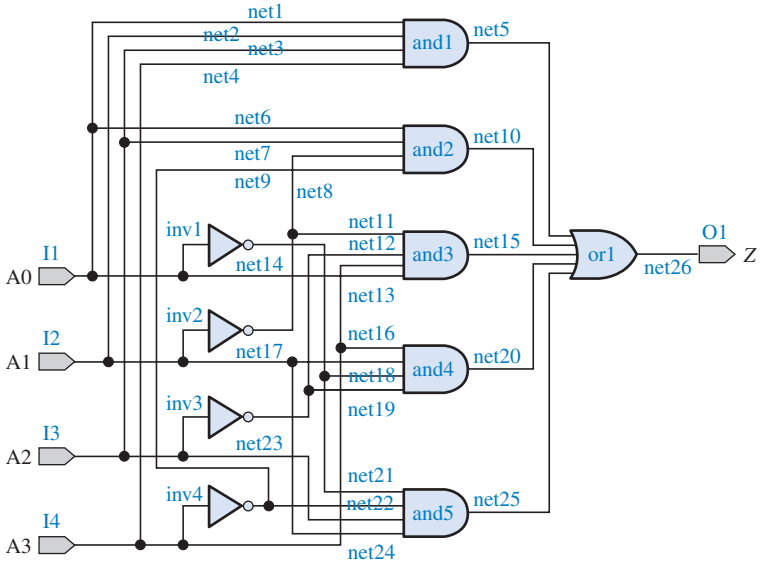
▲ **FIGURA 11–57**

Exemplo de otimização lógica durante a fase de síntese.

O software de síntese gera uma *netlist*. Para ilustrar o conceito de uma *netlist* genérica veja a Figura 11–58. A parte (a) da figura mostra em laranja as designações de conexões, de instâncias e de I/O. A *netlist* mostrada na Figura 11–58(b) não se assemelha a qualquer formato de *netlist* específica ou sintaxe. Essa *netlist* hipotética simplesmente indica o tipo de informação que seria necessária para descrever um circuito. Um formato usado para netlists é o **EDIF** (*Electronic Design Interchange Format*).

Implementação (Software)

Após o projeto ser sintetizado, o compilador implementa o projeto, que é basicamente um "mapeamento" do projeto de forma que ele seja "encaixado" no dispositivo destino com base na ar-



(a)

```
Netlist (Logic3)
net<name>: instance<name>, <from>; <to>;
instances: and1, and2, and3, and4, and5, or1, inv1, inv2,
inv3, inv4;
Input/outputs: I1, I2, I3, I4, O1;
net1: and1, inport1; I1;
net2: and1, inport2; I2;
net3: and1, inport3; I3;
net4: and1, inport4; I4;
net5: and1, outport1; or1, inport1;
net6: and2, inport1; I1;
net7: and2, inport2; I3;
net8: and2, inport3; inv2, outport1;
net9: and2, inport4; inv4, outport1;
net10: and2, outport1; or1, inport2;
net11: and3, inport1; inv2, outport1;
net12: and3, inport2; inv3, outport1;
net13: and3, inport3; I4;
net14: and3, inport4; I1;
net15: and3, outport1; or1, inport3;
net16: and4, inport1; I4;
net17: and4, inport2; I2;
net18: and4, inport3; inv1, outport1;
net19: and4, inport4; inv3, outport1;
net20: and4, outport1; or1, inport4;
net21: and5, inport1; inv1, outport1;
net22: and5, inport2; inv4, outport1;
net23: and5, inport3; I3;
net24: and5, inport4; I2;
net25: and5, outport1; or1, inport5;
net26: or1, outport1; O1;
end
```

(b)

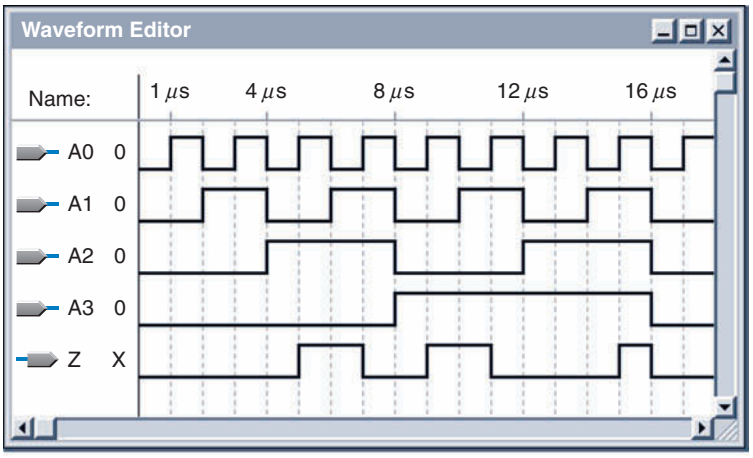
▲ **FIGURA 11–58**

A fase de síntese produz uma netlist para o circuito lógico otimizado.

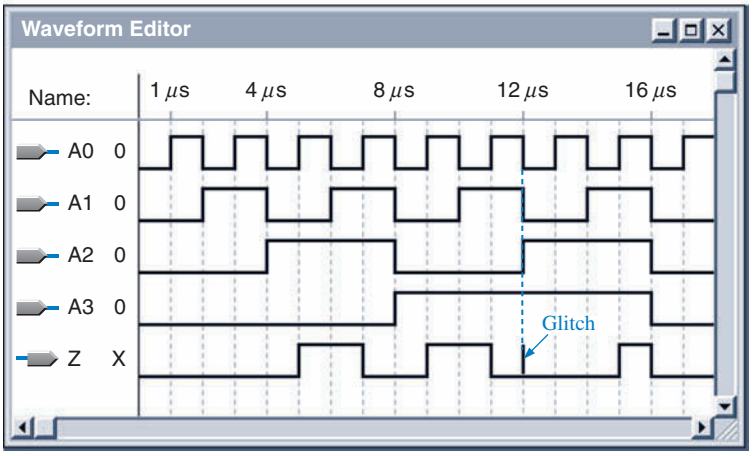
quitetura e configuração de pinos deste. Esse processo é denominado de *encaixe* ou *place and routing*. Para realizar a fase da implementação do fluxo do projeto, o software tem que "conhecer" os dispositivos específicos e ter informações detalhadas dos pinos. Os dados completos de todos os dispositivos destino potenciais são geralmente armazenados na biblioteca do software.

Simulação de Temporização

Esse parte do fluxo do projeto ocorre após a implementação e antes do download (transferência) para o dispositivo destino. A simulação de temporização verifica que o circuito trabalha na freqüência do projeto e que não existam atrasos de propagação ou outros problemas de temporização que afetem a operação global. Visto que a simulação funcional já foi realizada, o circuito deve funcionar adequadamente do ponto de vista da lógica. O software de desenvolvimento usa informações sobre o dispositivo destino específico, tal como atrasos de propagação das portas, para realizar a simulação de temporização do projeto. Para a simulação funcional, o dispositivo destino tem que ser escolhido. O Editor de Formas de Onda pode ser usado para visualizar o resultado da simulação de temporização assim como na simulação funcional, conforme ilustrado na Figura 11–59. Se não existirem problemas com a temporização, conforme mostra a parte (a) da figura, o projeto está pronto para a fase de download. Entretanto, suponha que a simulação de temporização revele um "*glitch*" devido ao atraso de propagação, conforme mostra a Figura 11–59(b). Um *glitch* é um *spike* de duração muito curta na forma de onda. Neste caso, devemos analisar cuidadosamente o projeto buscando a causa, para então reinserir o projeto modificado e repetir o processo do fluxo do projeto. Lembre-se, ainda não transferimos o projeto para o hardware.



(a) Resultado correto



(b) Problema de temporização

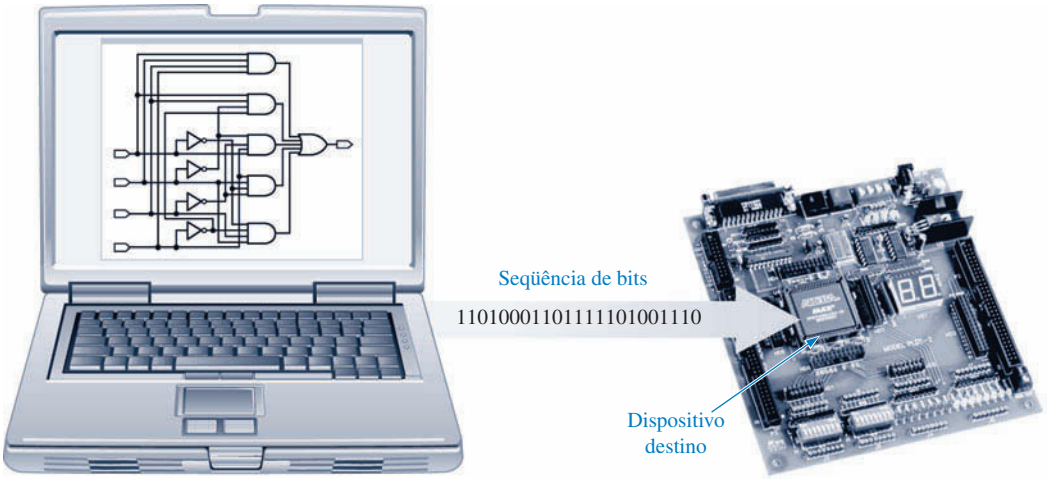
► **FIGURA 11–59**
Exemplos hipotéticos de resultados de simulação de temporização.

Programação do Dispositivo (download)

Uma vez que as simulações funcional e de temporização tenham verificado que o projeto está funcionando corretamente, podemos iniciar a seqüência de download. Uma **seqüência de bits** é gerada representando o projeto final, sendo enviada para o dispositivo destino configurando-o automaticamente. Feito isso, o projeto está realmente no hardware e pode ser testado no circuito. A Figura 11–60 mostra o conceito básico de **download**.

Uma Analogia com um Projeto de Assentamento

Uma forma de imaginar os processos de implementação e download é usar uma analogia com um projeto de assentamento real. O projetista começa com um levantamento do terreno e com a divisão do mesmo em lotes (análogo a um dispositivo não-programado). Um projeto de desenvolvimento para o local é idealizado com todos os lotes, prédios, estradas e utilidades (análogo à inserção do projeto). Em seguida, é necessário verificar se o número de prédios e infra-estruturas se encaixam no terreno identificando todos os códigos locais. O projeto do terreno também mostra a localização de cada prédio e o traçado de cada estrada e calçada (análogo à síntese e implementação). Enquanto esse mapeamento não estiver concluído, o projetista não pode começar a construção física dos prédios, rodovias e utilidades (análogo ao download). Essa analogia com um assentamento real é ilustrada na Figura 11–61(a). O processo de inserção de um projeto lógico em um dispositivo programável é ilustrado na parte (b) da figura, onde a fase de mapeamento é análoga ao projeto do terreno e a fase de download é análoga à edificação física das estruturas no terreno.



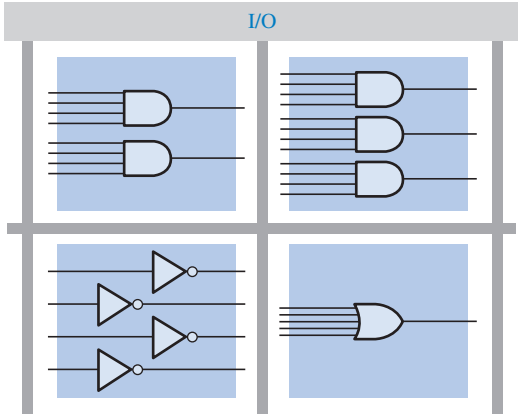
▲ **FIGURA 11–60**

Transferência de um projeto para o dispositivo destino.

O projeto do terreno precisa ser finalizado antes que o desenvolvimento físico seja implementado. Essa situação é análoga à fase de implementação do fluxo de projeto para a lógica programável.

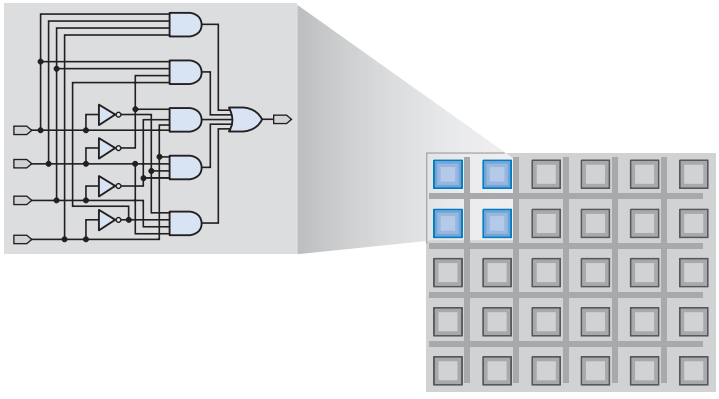
(a)

As estruturas físicas são colocadas apenas depois de verificar o projeto para saber como tudo se encaixará na propriedade. Essa situação é análoga ao download do projeto no dispositivo destino.



Encaixando o projeto virtual no "mapa" do dispositivo destino. (Análogo ao projeto do terreno.)

(b)



Download do projeto para o dispositivo destino. (Análogo à edificação física das estruturas.)

▲ **FIGURA 11–61**

As estruturas físicas são colocadas apenas depois de se verificar o projeto para saber como tudo se encaixará na propriedade. Essa situação é análoga ao download do projeto no dispositivo destino.

SEÇÃO 11–8 REVISÃO

1. Faça uma lista das fases do fluxo de projeto para uma lógica programável.
2. Faça uma lista dos elementos essenciais para a programação de um CPLD ou FPGA.
3. Qual é a finalidade de uma *netlist*?
4. O que deve acontecer primeiro no fluxo de um projeto, a simulação funcional ou a simulação de temporização?

11-9 LÓGICA BOUNDARY SCAN

A lógica *boundary scan* é usada no teste e na programação da lógica interna de um dispositivo lógico programável. O padrão JTAG para a lógica *boundary scan* é especificada pelo padrão 1149.1 da IEEE. A maioria dos dispositivos lógicos programáveis está de acordo com o JTAG. Nesta seção, a arquitetura básica de um dispositivo JTAG padrão 1149.1 da IEEE é apresentada e discutida em termos do seu registrador *boundary scan* e estrutura lógica de controle.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever os elementos necessários de um dispositivo que está de acordo com o padrão JTAG ■ Listar as entradas e a saída JTAG obrigatórias ■ Dizer a finalidade de um registrador *boundary scan* ■ Dizer a finalidade de um registrador de instrução ■ Explicar o que é um registrador de *bypass* (desvio)

Registradores no Padrão 1149.1 da IEEE

Todos os dispositivos programáveis que estão de acordo com o padrão 1149-1 da IEEE necessitam dos elementos mostrados no diagrama simplificado visto na Figura 11–62. Esses elementos são o registrador *boundary scan*, o registrador de *bypass*, o registrador de instrução e o circuito lógico TAP (*test access port* – porta de acesso para teste). Um quinto registro, o registro de identificação, é opcional e não é mostrado na figura.

Registrador Boundary Scan (BS) As BSCs (células *boundary scan*) interconectadas formam o registrador *boundary scan*. A entrada serial para o registrador é o TDI (*test data in* – entrada de dados para teste) e a saída serial é o TDO (*test data out*). Os dados a partir da lógica interna e dos pinos de entrada e saída do dispositivo também podem ser transferidos de forma paralela para o registrador BS. Esse registrador é usado para testar as conexões entre PLDs e a lógica interna que foi programada no dispositivo.

Registrador de Bypass (BP) Esse registrador de dados necessário (tipicamente apenas um flip-flop) otimiza o processo de transferência reduzindo o percurso entre o TDI e o TDO no caso do registrador BS, ou outro registrador, não ser usado.

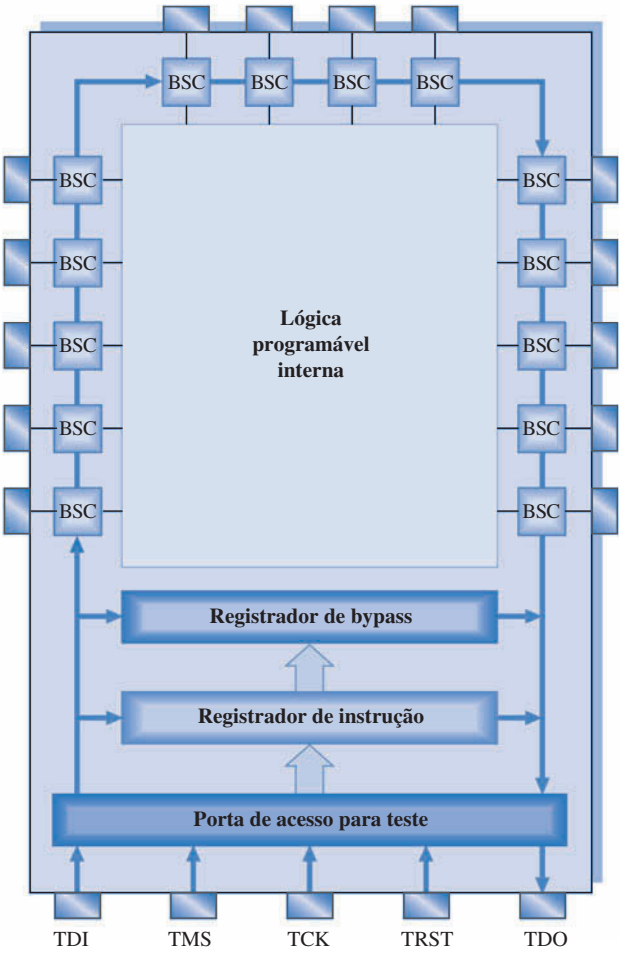
Registrador de Instrução Esse registrador necessário armazena instruções para a execução de diversas operações *boundary scan*.

Registrador de Identificação (ID) Um registrador de identificação é um registrador que não é necessário segundo o padrão 1149.1 da IEEE. Entretanto, ele é usado em algumas arquiteturas *boundary scan* para armazenar um código que identifica um dispositivo programável em particular.

Instruções Boundary Scan no Padrão 1149.1 da IEEE

Várias instruções padrão são usadas para controlar a lógica *boundary scan*. Além dessas, outras instruções opcionais estão disponíveis.

- *BYPASS* Essa instrução comuta o registrador BP para o percurso TDI/TDO.
- *EXTEST* Essa instrução comuta o registrador BS para o percurso TDI/TDO e permite que os testes dos pinos externos e os testes das interconexões entre a saída de um dispositivo lógico programável e a entrada de um outro.



◀ **FIGURA 11–62**

Diagrama significativamente simplificado de um dispositivo de lógica programável (CPLD ou FPGA) de acordo com o JTAG (padrão 1149.1 da IEEE). Os BSCs (células *boundary scan*) formam o registrador *boundary scan*. Apenas um pequeno número de BSCs são mostradas para ilustração.

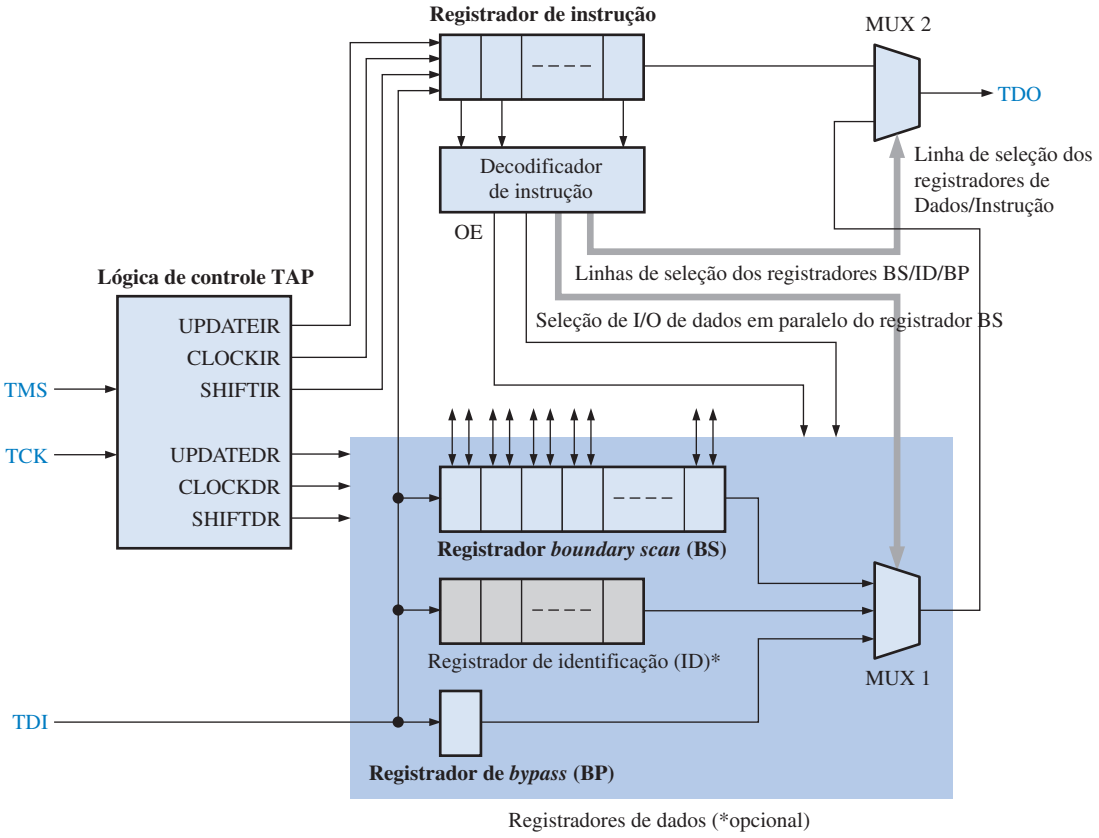
- *INTEST* Essa instrução comuta o registrador BS para o percurso TDI/TDO e permite o teste da lógica programável interna.
- *SAMPLE/PRELOAD* Essa instrução é usada para amostra de dados nos pinos de entrada do dispositivo e para aplicar os dados na lógica interna. Além disso, ela é usada para aplicar dados (pré-carga) a partir da lógica interna para os pinos de entrada do dispositivo.
- *IDCODE* Essa instrução comuta o registrador de identificação opcional para o percurso TDI/TDO, assim o código ID pode ser transferido para o TDO.

Porta de Acesso para Teste (TAP) no Padrão 1149.1 da IEEE

A porta de acesso para teste (TAP) consiste de lógica de controle, quatro entradas e saída obrigatórias e uma entrada opcional definida, Teste de RESET (TRST).

- *Test Data In (TDI)* O TDI provê o teste de deslocamento serial e a programação de dados bem como instruções para dentro da lógica *boundary scan*.
- *Test Data Out (TDO)* O TDO provê o teste de deslocamento serial e a programação de dados bem como instruções para fora da lógica *boundary scan*.
- *Test Mode Select (TMS)* O TMS comuta entre os estados do controlador TAP.
- *Test Clock (TCK)* O TCK provê temporização para o controlador TAP que gera sinais de controle para o registrador de dados e o registrador de instruções.

Um diagrama em bloco da lógica *boundary scan* é mostrado na Figura 11–63. As instruções e dados são deslocados para dentro através da linha TDI. O controlador TAP direciona as instruções para o registrador de instrução ou os dados para o registrador de dados apropriado. Uma instrução decodificada, a partir do decodificador de instrução, seleciona qual registrador será acessado via MUX 1 e também se uma instrução ou dado será deslocada para fora na linha TDO via MUX 2. Além disso, uma instrução decodificada provê o estabelecimento do registrador *boundary scan* em um dos cinco modos básicos. A célula *boundary scan* e os seus modos de operação são descritos a seguir.



▲ **FIGURA 11–63**

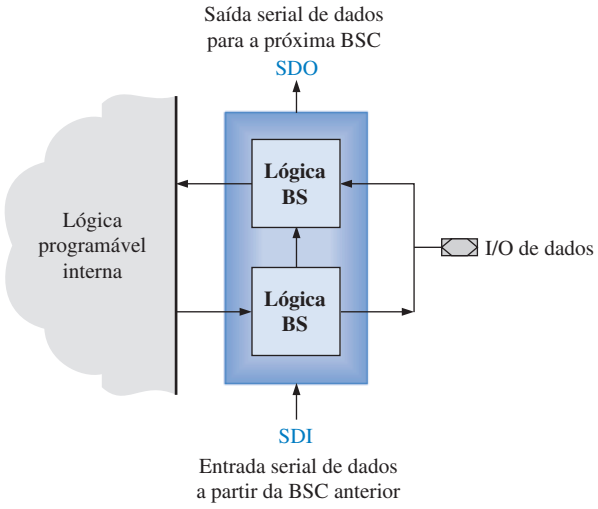
Diagrama lógico *boundary scan*.

A Célula Boundary Scan (BSC)

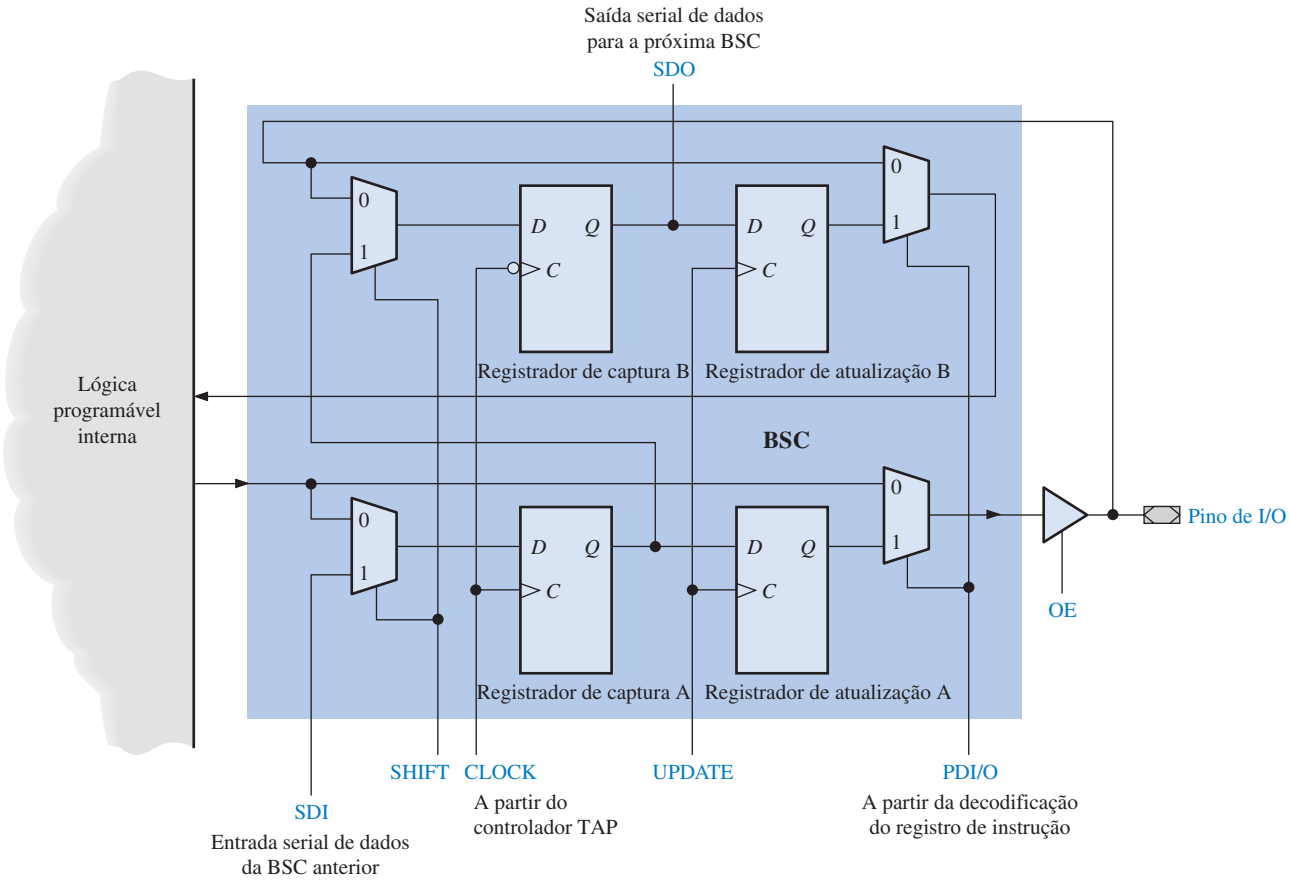
O registrador *boundary scan* é feito de células *boundary scan*. A Figura 11–64 mostra um diagrama em bloco de uma BSC básica. Conforme indicado, os dados podem ser deslocados serialmente para dentro e para fora da BSC. Além disso, os dados podem ser deslocados para a BSC a partir da lógica programável interna, de um pino de entrada do dispositivo, ou de uma BSC anterior. Adicionalmente, os dados podem ser deslocados da BSC para a lógica de programação interna, para o pino de saída do dispositivo ou para a próxima BSC.

A arquitetura de uma célula *boundary scan* genérica é mostrada na Figura 11–65. A célula consiste de dois circuitos lógicos idênticos cada um contendo dois flip-flops e dois multiplexadores. Essencialmente, um circuito permite o deslocamento dos dados da lógica programável interna para um pino de saída do dispositivo. O outro circuito permite o deslocamento de dados de um pino de entrada do dispositivo para a lógica de programação interna.

Existem cinco modos no qual a BSC pode operar em termos de fluxo de dados. O primeiro modo permite que os dados se desloquem de forma serial da BSC anterior para a próxima, confor-

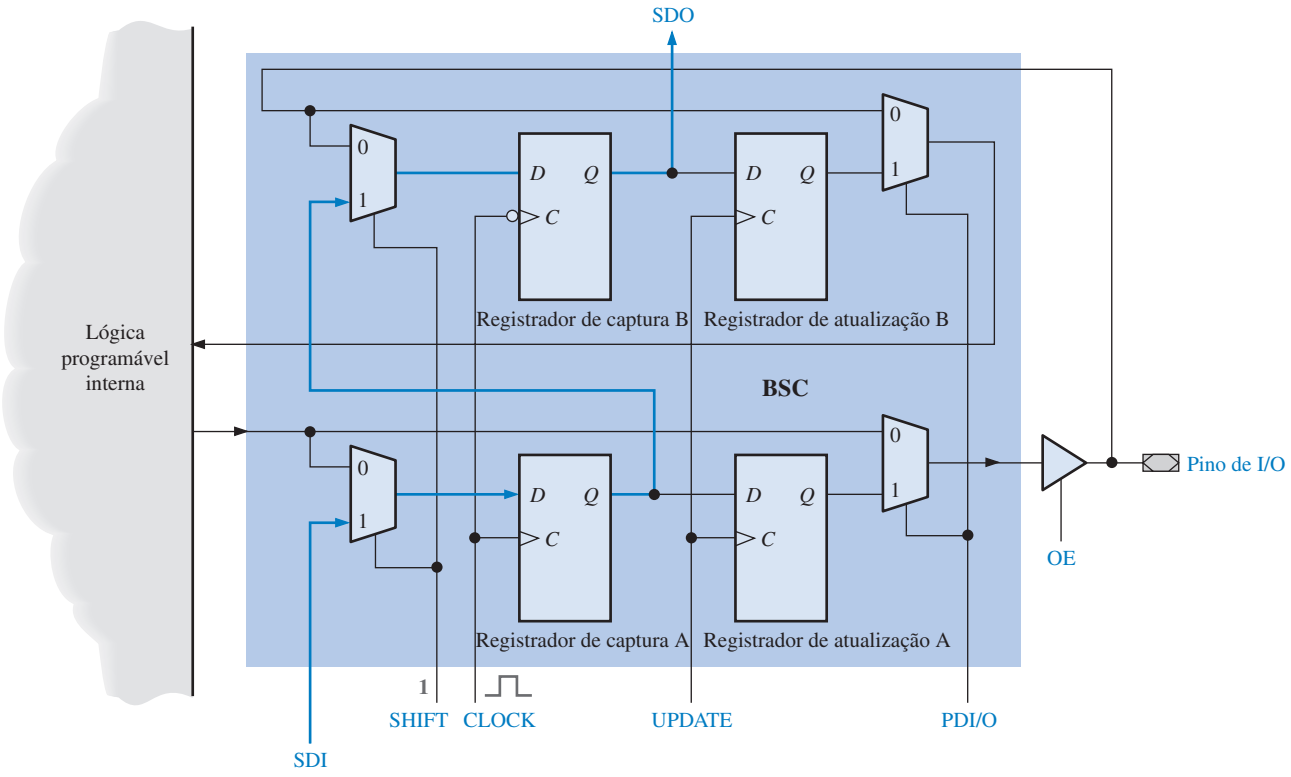


◀ **FIGURA 11–64**
Uma BSC bidirecional básica.



▲ **FIGURA 11–65**
Arquitetura representativa de uma célula *boundary scan* típica.

me ilustra a Figura 11–66. Um nível 1 na entrada SHIFT seleciona a linha SDI. O dado nessa linha é colocado no Registrador de captura A na borda positiva do CLOCK. O dado é então colocado no Registrador de captura B na borda negativa do CLOCK e aparece na linha SDO. Isso é equivalente ao deslocamento serial de dados através do registrador *boundary scan*.



▲ **FIGURA 11–66**

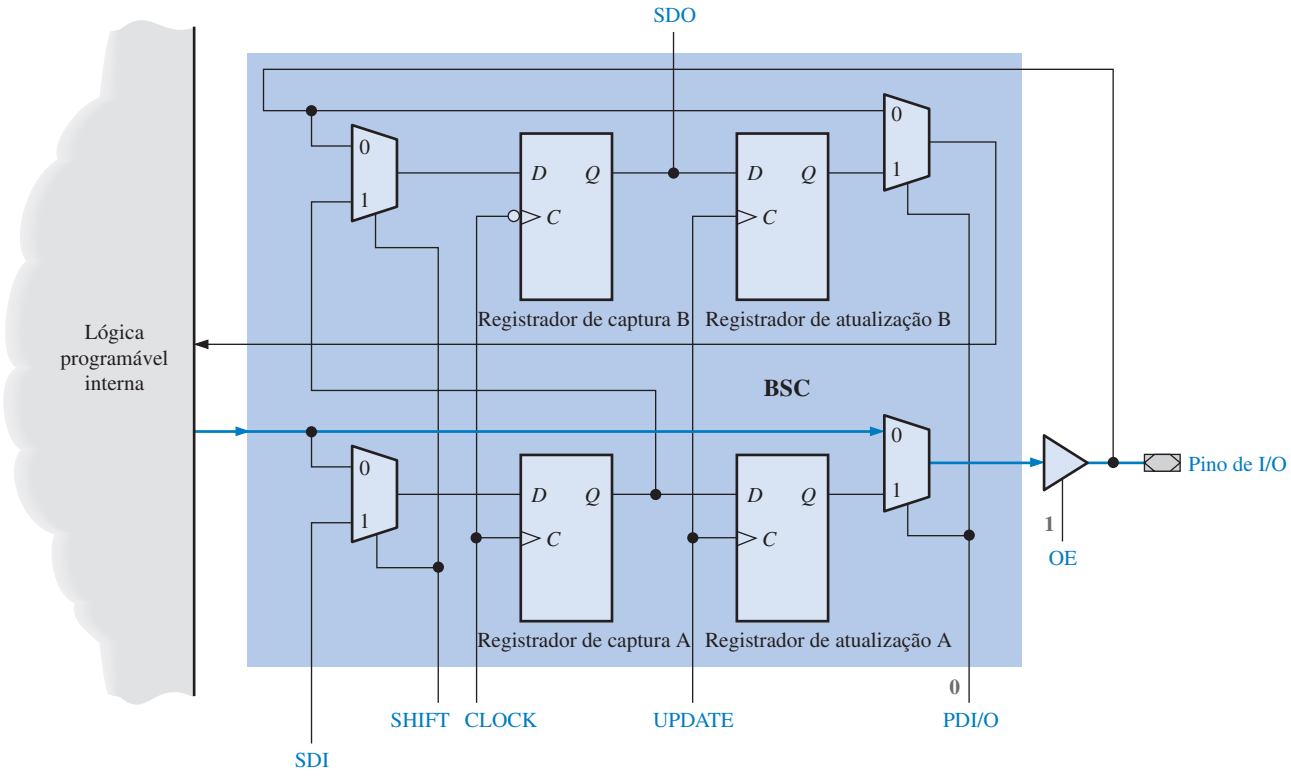
Percurso para um deslocamento serial de dados de uma BSC para a próxima. Existe um nível 1 na entrada SHIFT quando um pulso de CLOCK é aplicado. As linhas em laranja indicam o fluxo de dados.

O segundo modo de operação da BSC permite o fluxo direto dos dados da lógica programável interna para um pino de saída do dispositivo, conforme ilustra a Figura 11–67. Um nível 1 na linha de controle PDI/O (I/O de dados em paralelo) seleciona os dados a partir da lógica programável interna. O nível 1 na linha OE (habilitação da saída) habilita o buffer de saída.

O terceiro modo de operação de uma BSC permite o fluxo direto de dados de um pino de entrada do dispositivo para a lógica programável interna, conforme ilustra a Figura 11–68. O nível 0 na linha de controle PDI/O (I/O de dados em paralelo) seleciona os dados a partir do pino de entrada. O nível 0 na linha OE (habilitação da saída) desabilita o buffer de saída.

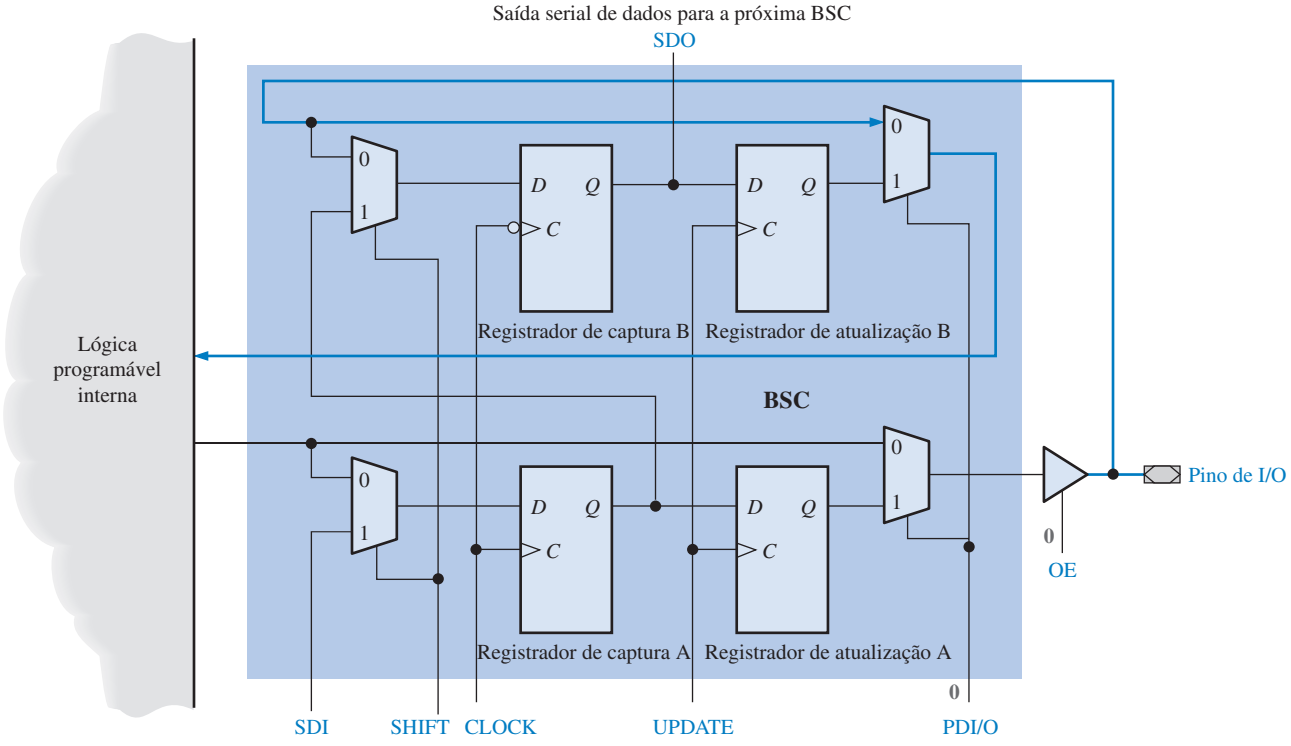
O quarto modo de operação de uma BSC permite o fluxo de dados a partir de SDI para a lógica programável interna, conforme ilustra a Figura 11–69. Um nível 1 na entrada SHIFT seleciona a SDI. O dado na linha SDI é colocado no Registrador de captura A na borda positiva do CLOCK. O dado é então transferido para o Registrador de captura B na borda negativa do CLOCK aparecendo na linha SDO. Um pulso na linha UPDATE (atualização) faz com que o dado passe para o Registrador de atualização B. Um nível 1 na linha PDI/O seleciona a saída do Registrador de atualização B transferindo esse dado para a lógica programável interna. Esse dado também aparece na linha SOD.

O quinto modo de operação de uma BSC permite o fluxo de dados da entrada SDI para um pino de saída do dispositivo e para a saída SDO, conforme ilustra a Figura 11–70. Um nível 1 na entrada SHIFT seleciona a entrada SDI. O dado nessa entrada é colocado no Registrador de captura A na borda positiva do CLOCK. O dado é então transferido para o Registrador de captura B na borda negativa do CLOCK, aparecendo também na linha SDO. Um pulso na linha UPDATE faz com que o dado passe para o Registrador de atualização A. Com um nível 1 em OE, o nível 1 na linha PDI/O seleciona a saída do Registrador de atualização A transferindo esse dado para o pino de saída do dispositivo.



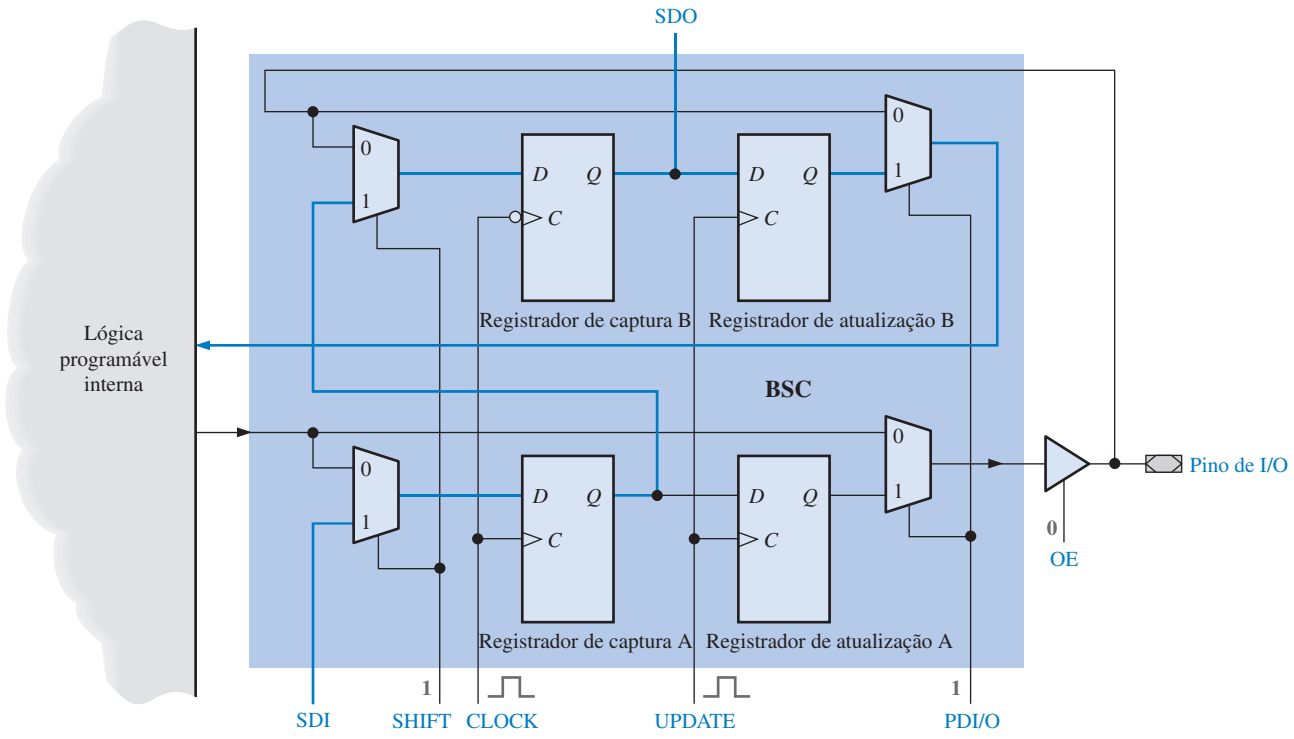
▲ **FIGURA 11–67**

Percurso para a transferência de dados da lógica programável interna para um pino de saída do dispositivo. Existe um nível 0 na linha PDI/O e um nível 1 na linha OE.



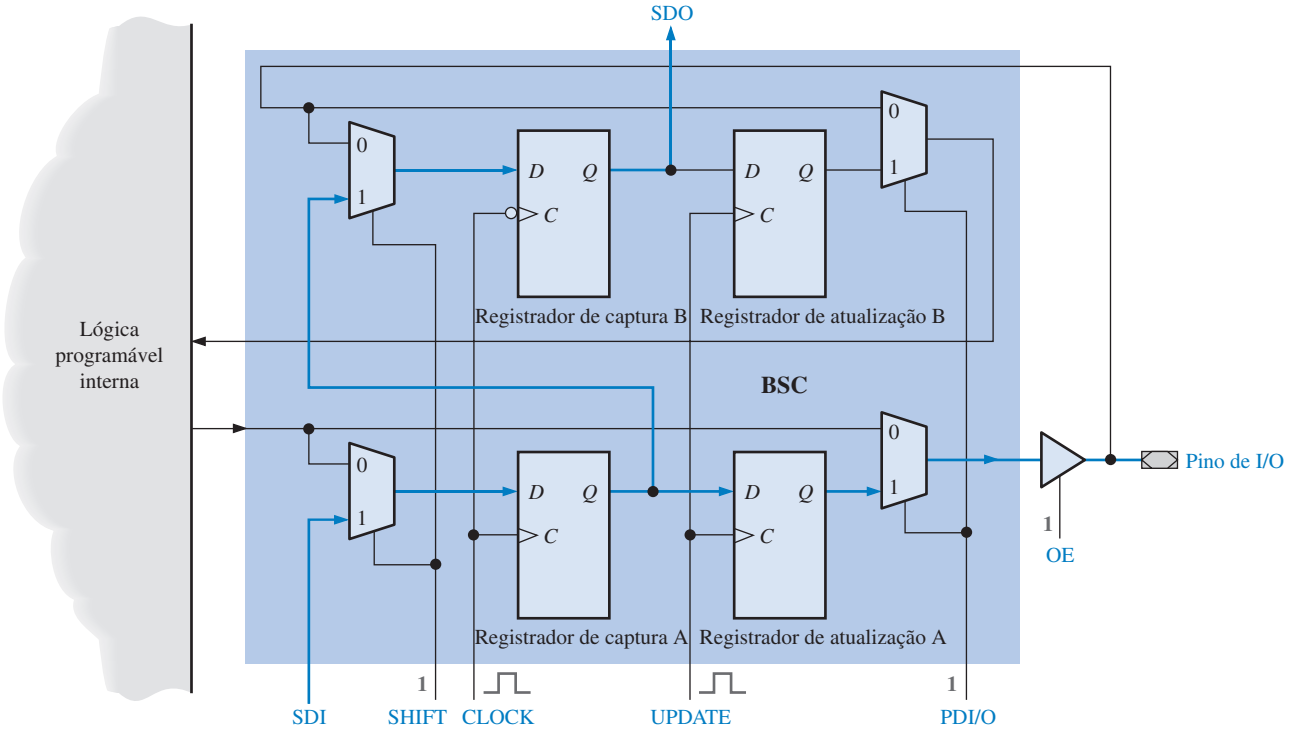
▲ **FIGURA 11–68**

Percurso para a transferência de dados de um pino de entrada de um dispositivo para a lógica programável interna. Existe um nível 0 nas linhas PDI/O e OE.



▲ **FIGURA 11–69**

Percurso para transferência de dados da entrada SDI para a lógica programável interna e para a saída SDO. Existe um nível 1 na linha SHIFT, um nível 1 na linha PDI/O e um nível 0 na linha OE. Um pulso é aplicado na entrada CLOCK seguido de um pulso na linha UPDATE.

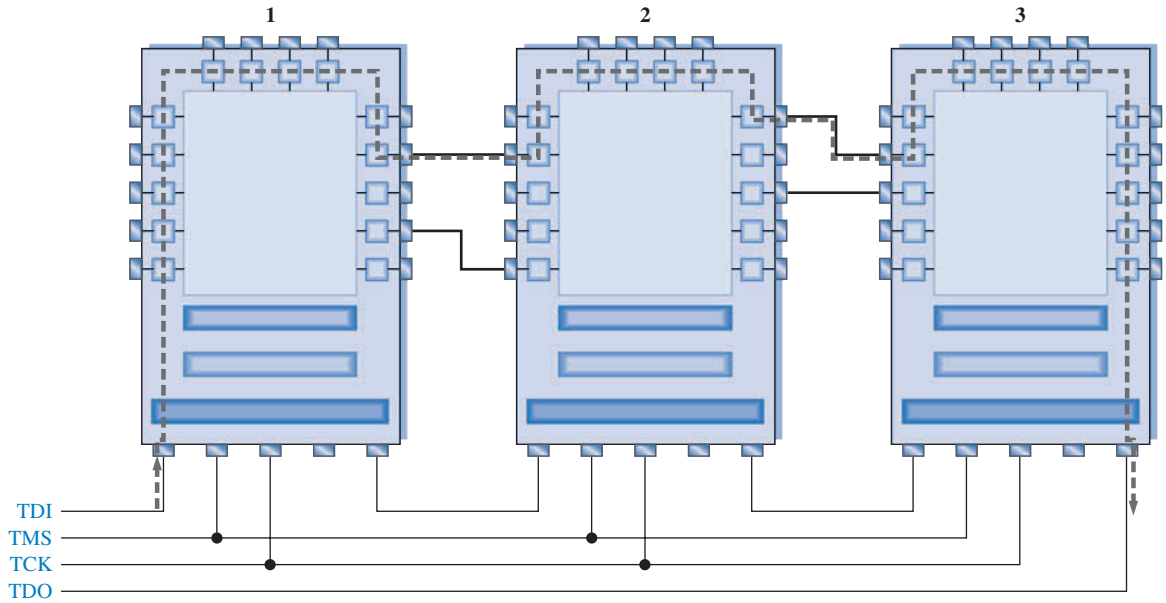


▲ **FIGURA 11–70**

Percurso para transferência de dados da entrada SDI para um pino de saída do dispositivo e para a saída SDO. Existe um nível 1 na linha SHIFT, um nível 1 na linha PDI/O e um nível 1 na linha OE. Um pulso é aplicado na entrada de CLOCK seguido de um pulso na linha UPDATE.

Teste de Boundary Scan de Múltiplos Dispositivos

O teste *boundary scan* pode ser aplicado numa placa de circuito impresso na qual múltiplos dispositivos JTAG (padrão 1149.1 da IEEE) são montados para verificar as interconexões bem como a lógica interna. Esse conceito é ilustrado traçando o percurso dos dados mostrado em cinza na Figura 11–71.



▲ **FIGURA 11–71**

Conceito básico do teste *boundary scan* de múltiplos dispositivos e interconexões. O percurso de teste é mostrado em cinza.

O bit é deslocado para o TDI do dispositivo 1 e, através do registrador BS do dispositivo 1, para a célula onde a conexão a ser testada vai para o dispositivo 2. O bit é deslocado para fora pelo pino de saída do dispositivo 1 e, através da interconexão com o pino de entrada do dispositivo 2, passa para este. O bit continua através do registrador BS do dispositivo 2 para o pino de saída passando, através da interconexão, para o dispositivo 3. Ele é então deslocado através do registrador BS do dispositivo 3 para a saída TDO. Se o bit que aparecer na saída TDO for o mesmo da entrada TDI, as células *boundary scan* através das quais o bit foi deslocado e as interconexões do dispositivo 1 para o 2 e do 2 para o 3 estão boas.

SEÇÃO 11–9 REVISÃO

1. Liste as entradas e saídas *boundary scan* necessárias segundo o padrão 1149.1 da IEEE.
2. O que é a TAP?
3. Determine quais são os registradores obrigatórios na lógica *boundary scan*.
4. Descreva os cinco modos nos quais uma célula *boundary scan* pode operar em termos do fluxo de dados.

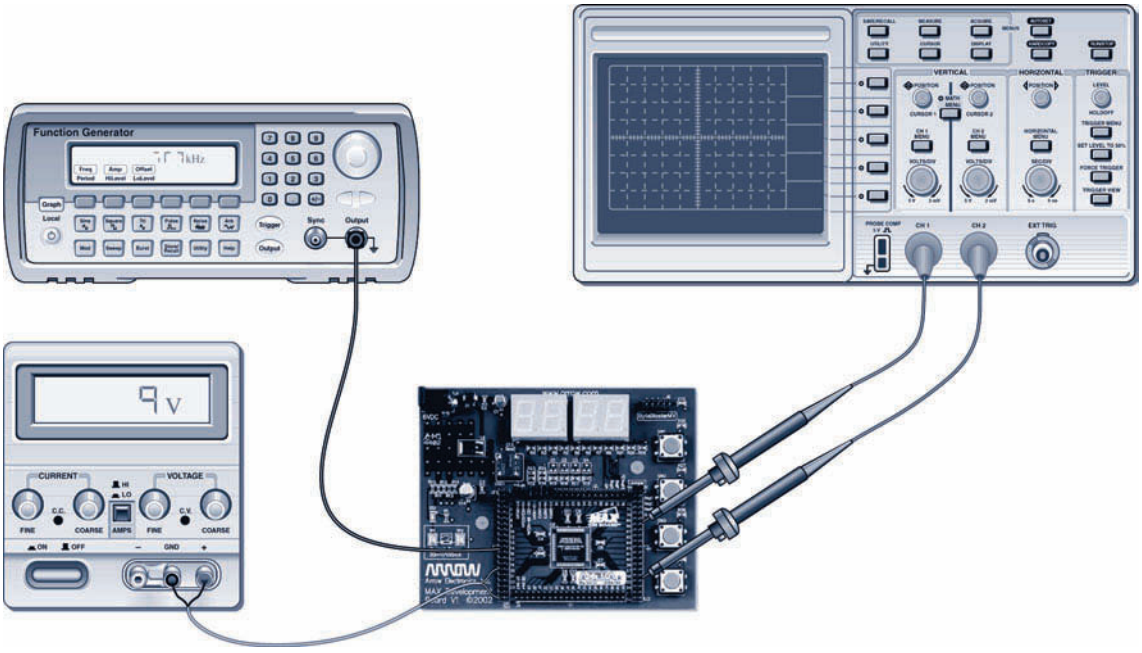
11-10 ANÁLISE DE DEFEITO

Duas formas básicas de testar um dispositivo programado com um projeto lógico são a tradicional e a automatizada. No método tradicional, podem ser usados instrumentos comuns de teste de laboratório para verificar a operação do dispositivo. No método automatizado, três abordagens fundamentais podem ser usadas no teste: *bed-of-nails*, *flying probe* e *boundary scan*.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever o teste tradicional de um dispositivo ■ Descrever as técnicas de teste *bed-of-nails* e *flying probe*, bem como discutir suas limitações ■ Discutir o padrão JTAG ■ Descrever o conceito básico de *boundary scan* ■ Explicar os modos de teste *boundary scan* e discutir resumidamente a BSDL

Após "materializarmos" um projeto lógico num hardware, podemos testar o dispositivo montado numa PCB (placa de circuito impresso). Para projetos relativamente simples, podemos fazer isso usando instrumentos de teste de laboratório como o osciloscópio ou o analisador lógico, gerador de sinal e fonte de alimentação cc. Podemos aplicar sinais de entrada nos pinos de entrada da placa e verificar os pinos de saída observando se as formas de onda estão corretas. Essa abordagem tradicional, ilustrada na Figura 11–72, é prática para um tipo de placa de avaliação e para testes de circuitos protótipos.

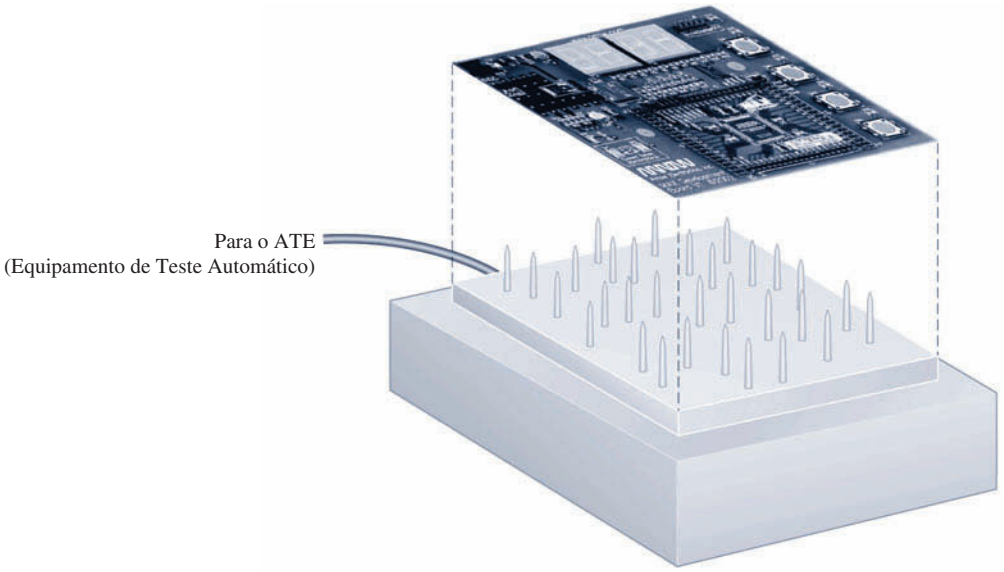


▲ **FIGURA 11–72**

Teste tradicional usando instrumentos de laboratório.

Teste Bed-of-Nails

O teste de uma placa de circuito impresso (PCB) em níveis de produção tem que ser feito automaticamente. O método da **bed-of-nails** (BON) foi uma das primeiras abordagens para teste automatizado. O conceito é ilustrado na Figura 11–73 onde a PCB é colocada em um acessório de fixação com um arranjo de pontas de prova semelhante a pequenas unhas (*nails*) as quais fazem contato com as ilhas de teste na placa. As "*nails*" são arranjadas em um arranjo que se alinha com a disposição das ilhas de teste na placa. Com esse método, os pontos de teste podem ser verificados



◀ **FIGURA 11–73**

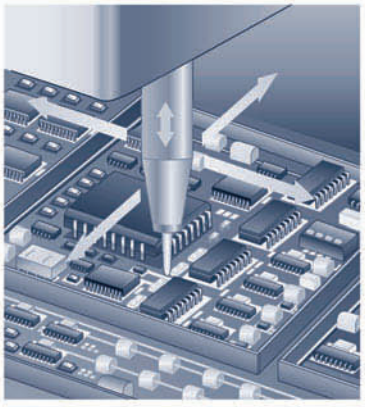
Conceito básico do método *bed-of-nails* (BON) para o teste de uma placa de circuito.

simultaneamente com um equipamento de teste automatizado especial. Basicamente, a finalidade do teste de produção automatizada é determinar qualquer defeito, tais como pinos em circuito aberto ou curto-circuito e componentes errados, faltantes ou desalinhados. Esse processo automatizado não testa essencialmente a funcionalidade do circuito lógico. Considera-se que cada componente tenha sido testado em sua funcionalidade antes de ser inserido na PCB e que os defeitos que surgem são devidos apenas ao processo de fabricação.

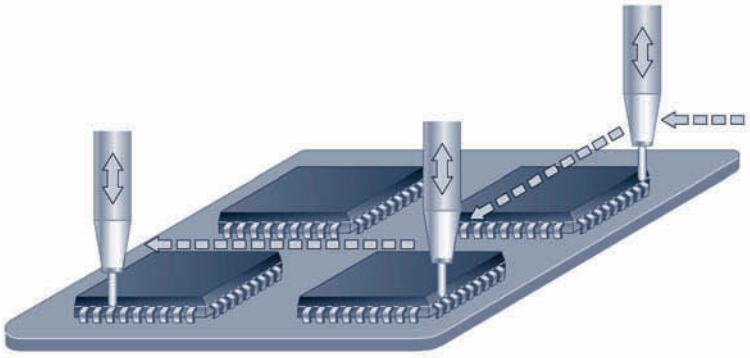
À medida que os dispositivos na forma de CIs se tornam menores e mais complexos, a tendência para a tecnologia de montagem em superfície (SMT) aumenta, sendo que as placas de circuitos passam de dupla face para multicamadas. O aumento da densidade e complexidade das placas de circuito e dispositivos, com um número maior de pinos com espaçamentos cada vez menores, resulta em um acesso limitado para os pontos de teste na placa usando o método BON.

Teste Flying Probe

Um outro método para teste de PCBs é denominado ***flying probe***. A Figura 11–74 mostra uma caracterização típica desse método e sua operação básica. Uma ponta de prova de teste é posicionada acima da placa de circuito a ser testada. A ponta de prova pode ser movimentada automatica-



(a) Movimento em três eixos



(b) Movimento da ponta de prova de um ponto para outro

▲ **FIGURA 11–74**

Ilustração do método *flying probe* testando uma placa de circuito.

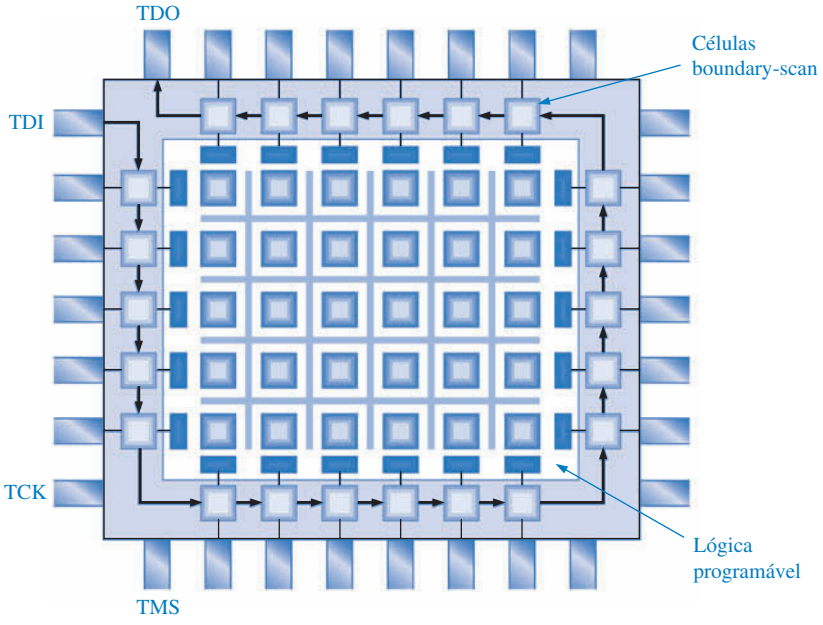
mente ao longo de três eixos (*x*, *y* e *z*) podendo fazer contato com qualquer ponto de teste especificado. O movimento da ponta de prova é controlado por software que usa o leiaute físico da placa para determinar as coordenadas. Muitos testadores que usam esse método, têm múltiplas pontas de prova para o teste de uma placa.

Esse método, *flying probe*, supera algumas das limitações do método *bed-of-nails* (BON). Primeiro, o método BON requer um acessório de fixação diferente para cada tipo de placa de circuito, porém o método *flying probe* não requer acessório. Além disso, o *flying probe* pode acessar mais pontos na placa porque a ponta de prova pode ser movida para qualquer posição e essa ponta pode acessar a parte superior da placa onde estão os componentes. Uma desvantagem do método *flying probe* é a de ser mais lento que o BON e dessa forma geralmente está limitado ao teste de protótipos e circuitos de uma produção pequena.

Teste da Boundary Scan

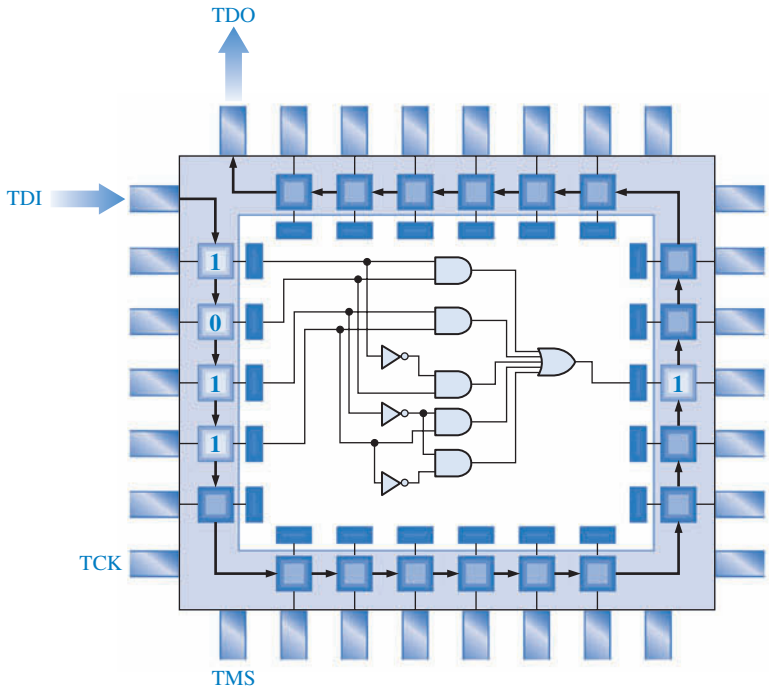
O acesso limitado a pontos de teste levou ao conceito de colocar os pontos de teste dentro dos próprios circuitos integrados. A maioria dos CPLDs e FPGAs inclui a lógica *boundary scan* como parte de sua estrutura interna independente da funcionalidade da lógica programada no dispositivo. Esses dispositivos estão de acordo com o padrão JTAG.

Um circuito, conhecido como célula *boundary scan*, é colocado entre a lógica programável e cada pino de entrada e saída do dispositivo, como mostra a Figura 11–75. As células são basicamente células de memória que armazenam um nível 1 ou 0. As células conectadas às entradas da lógica programável são denominadas de células de entrada, e aquelas conectadas às saídas da lógica programável são as células de saída. O teste ***boundary scan*** é baseado no padrão JTAG (padrão 1149.1 da IEEE). As quatro entradas e saídas JTAG – TDI (*test data in*), TDO (*test data out*), TCK (*test clock*) e TMS (*test mode select*) – são conhecidas como portas de acesso para teste (TAP).



► **FIGURA 11–75**
Conceito básico da lógica *boundary scan* em um dispositivo programável.

Intest Quando as células *boundary scan* são usadas para testar a funcionalidade interna do dispositivo, o modo de teste é denominado de *Intest* (teste interno). O conceito básico de *boundary scan* usando o *Intest* é o seguinte: Um software com um padrão de 1s e 0s é deslocado para dentro do dispositivo via pino TDI e colocado nas entradas da lógica programável. Como resultado da aplicação desses bits de entrada, a lógica produz bit(s) de saída em resposta. O(s) bit(s) de saída resultante(s) é(são) deslocado(s) para o pino TDO e feita a verificação de erros. Uma saída incorreta, é claro, indica uma falha na lógica programada, as células de I/O, ou as células *boundary scan*.



◄ FIGURA 11–76

Exemplo de um padrão de bit no *boundary scan Intest* para a lógica interna.

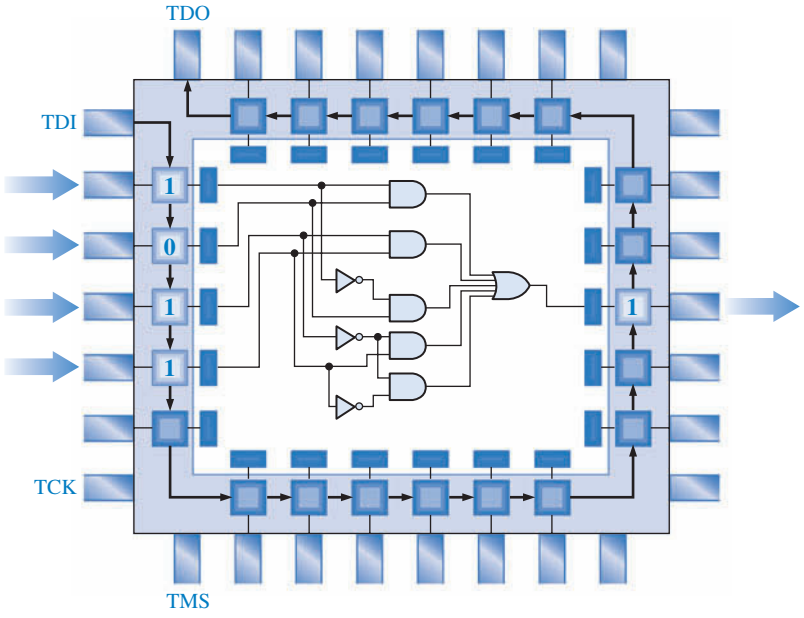
A Figura 11–76 mostra um padrão 1011 do *boundary scan Intest* para um circuito lógico AND-OR que foi programado em um dispositivo. Dezesseis combinações de quatro bits TDI testam o circuito em todos os estados possíveis de acordo com a lista dada na Tabela 11–1. As combinações de 4 bits são deslocadas de forma serial para as células *boundary scan* sendo que a saída correspondente é deslocada para fora, via pino TDO, para verificação. Esse processo é controlado pelo software de teste *boundary scan*.

◄ TABELA 11–1

Padrão de bits para o teste *boundary scan* para o dispositivo programado mostrado na Figura 11–76

TDI	TDO
0000	1
0001	1
0010	0
0011	1
0100	1
0101	1
0110	1
0111	1
1000	1
1001	1
1010	0
1011	1
1100	1
1101	1
1110	1
1111	1

Extest Quando as células *boundary scan* são usadas para testar as conexões externas do dispositivo além de alguma funcionalidade interna, o modo de teste é denominado de *Extest*. O conceito básico de *boundary scan* usando *Extest* é o seguinte: Um software com um padrão de bits 1s e 0s é aplicado nos pinos de entrada do dispositivo e transferido para as células de entrada. Como resultado desses bits, a lógica produz bit(s) de saída em resposta. O(s) bit(s) de saída resultante é(são) obtido(s) do pino de saída do dispositivo e é feita a verificação de erros. Uma saída incorreta, é claro, indica um defeito nas conexões ou interconexões dos pinos de entrada ou saída, um dispositivo incorreto ou ainda um dispositivo instalado de forma errada. Obviamente, alguns defeitos internos também podem ser detectados no modo *Extest*. Por exemplo, defeitos em células *boundary scan*, células de I/O ou certos defeitos na lógica programável que produzem uma saída incorreta. A Figura 11–77 mostra um exemplo de *boundary scan Extest* que testa as quatro entradas e a saída do circuito lógico.

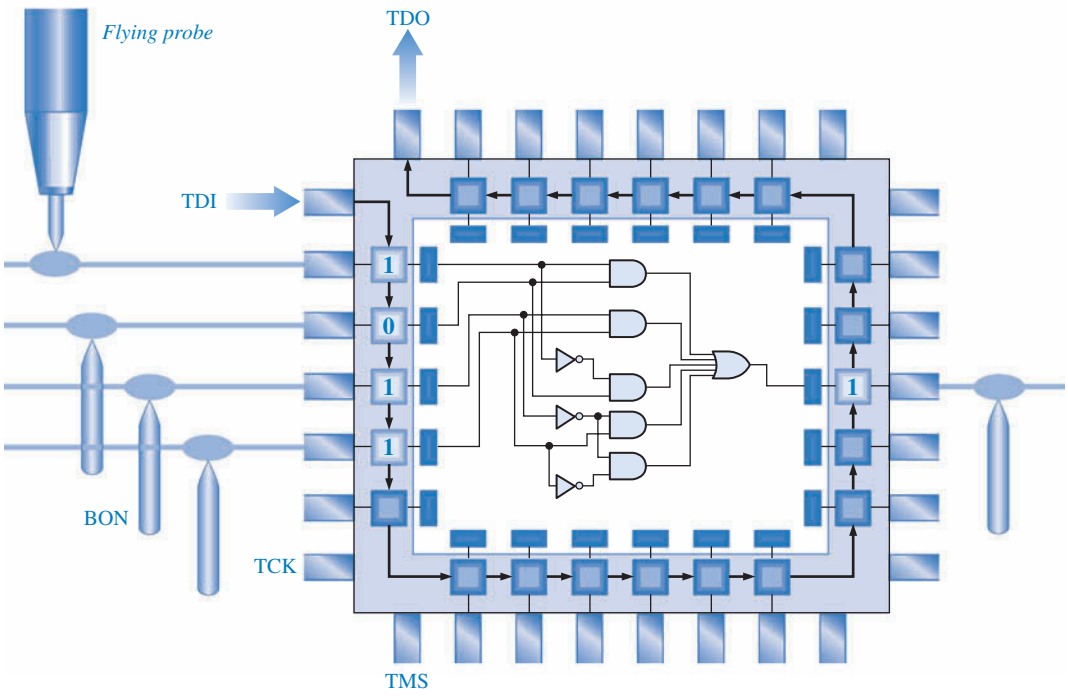


► **FIGURA 11–77**
Exemplo de um padrão de bit no *boundary scan Extest* para defeitos externos.

Se um defeito for detectado no modo *Extest*, este defeito pode ser externo (uma conexão ruim de um pino) ou interno (uma conexão defeituosa, uma célula *boundary scan* ou um elemento lógico) do dispositivo. Então, para isolar o defeito detectado no Extest, um Intest deve ser executado em seguida ao Extest. Se ambos indicam um defeito, então esse defeito é interno ao dispositivo.

No modo Extest, é necessário testar com uma ponta de prova os pinos de entrada e saída do dispositivo. Esses pinos têm que estar disponíveis em um conector na placa de circuito ou em uma ilha de teste, assim eles podem ser verificados por um equipamento de teste automático. Os pinos que não fazem parte do conector de placa JTAG podem ser testados usando o método BON e/ou o método *flying probe*. Essas abordagens combinadas são ilustradas na Figura 11–78.

Linguagem de Descrição para Boundary Scan (BSDL) Esse software de teste é parte do padrão JTAG (padrão 1149.1 da IEEE) e usa VHDL para descrever como a lógica *boundary scan* é implementada em um dispositivo específico e como ela opera. O BSDL provê um formato de dados padrão para descrever como o padrão 1149.1 da IEEE é implementado em um dispositivo de



▲ **FIGURA 11–78**

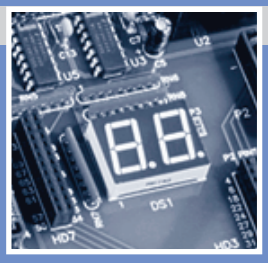
Uma combinação dos testes *boundary scan*, *bed-of-nails* e *flying probe*.

acordo com o JTAG. Quando usamos as ferramentas de teste *boundary scan* que suportam BSDL, podemos geralmente obter o BSDL do fabricante do dispositivo.

Cada dispositivo que contém lógica *boundary scan* dedicada é suportado pelo arquivo BSDL que descreve o dispositivo em particular. Do que descrito no arquivo BSDL citamos o tipo do dispositivo e a descrição das portas que nomeiam os pinos de I/O e os pinos da porta de acesso para teste (TAP) e a indicação da natureza deles como entrada, saída ou bidirecional. O BSDL também provê um mapeamento dos sinais lógicos nos pinos físicos e uma descrição da arquitetura da lógica *boundary scan* contida no dispositivo. Um padrão de teste de bit para o teste do dispositivo pode ser definido usando BSDL.

SEÇÃO 11–10 REVISÃO

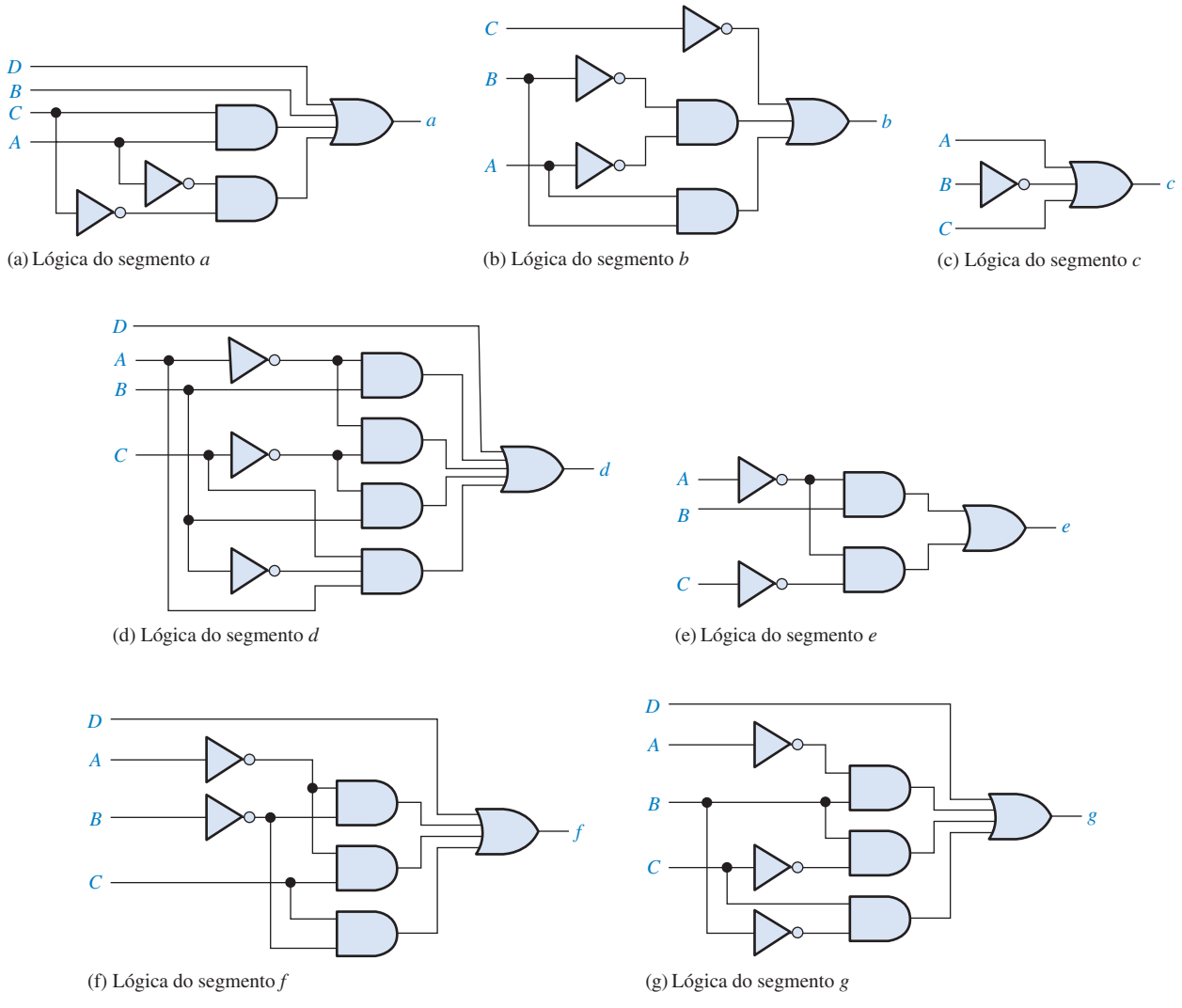
1. Descreva o conceito básico do método BON para teste de placa.
2. Quais as limitações do método BON?
3. Em que o método de teste *flying probe* difere do BON?
4. Explique o conceito básico de *boundary scan*.
5. Quais são os dois modos de teste de *boundary scan*?
6. Cite os nomes dos quatro sinais JTAG usados no *boundary scan*.
7. O que é BSDL?



APLICAÇÕES EM SISTEMAS DIGITAIS

Nesse tópico de Aplicações em Sistemas Digitais, as ferramentas genéricas de software de desenvolvimento que usa o procedimento de inserção via esquemático, que aprendemos nesse capítulo, são aplicadas na lógica do decodificador de BCD para 7 segmentos que foi desenvolvida no Capítulo 4. Apenas os passos principais são mostrados aqui numa abordagem genérica para ilustrar o conceito básico.

A Figura 11–79 mostra os circuitos lógicos individuais para cada um dos sete segmentos. Cada um desses circuitos é inserido como um arquivo separado e então convertido para a forma de um bloco. Após a inserção de todos os arquivos com a lógica dos segmentos, eles são combinados para formar um único bloco que é o decodificador completo. Todos os sete circuitos lógicos devem ser inseridos um de cada vez para criar o que é conhecido como esquema "plano". Entretanto, usamos a abordagem hierárquica do projeto para termos toda a lógica na tela do Editor Gráfico de uma vez e mais fácil de ser usada. Essa abordagem é preferida quando o



(a) Lógica do segmento *a*

(b) Lógica do segmento *b*

(c) Lógica do segmento *c*

(d) Lógica do segmento *d*

(e) Lógica do segmento *e*

(f) Lógica do segmento *f*

(g) Lógica do segmento *g*

▲ **FIGURA 11–79**

Circuitos lógicos dos sete segmentos individuais.

esquema é bastante complexo e pode ser dividido em diversas partes. Além disso, nas situações onde várias pessoas estão trabalhando nos circuitos que mais tarde serão combinados em um circuito ou sistema maior, a abordagem hierárquica é essencial.

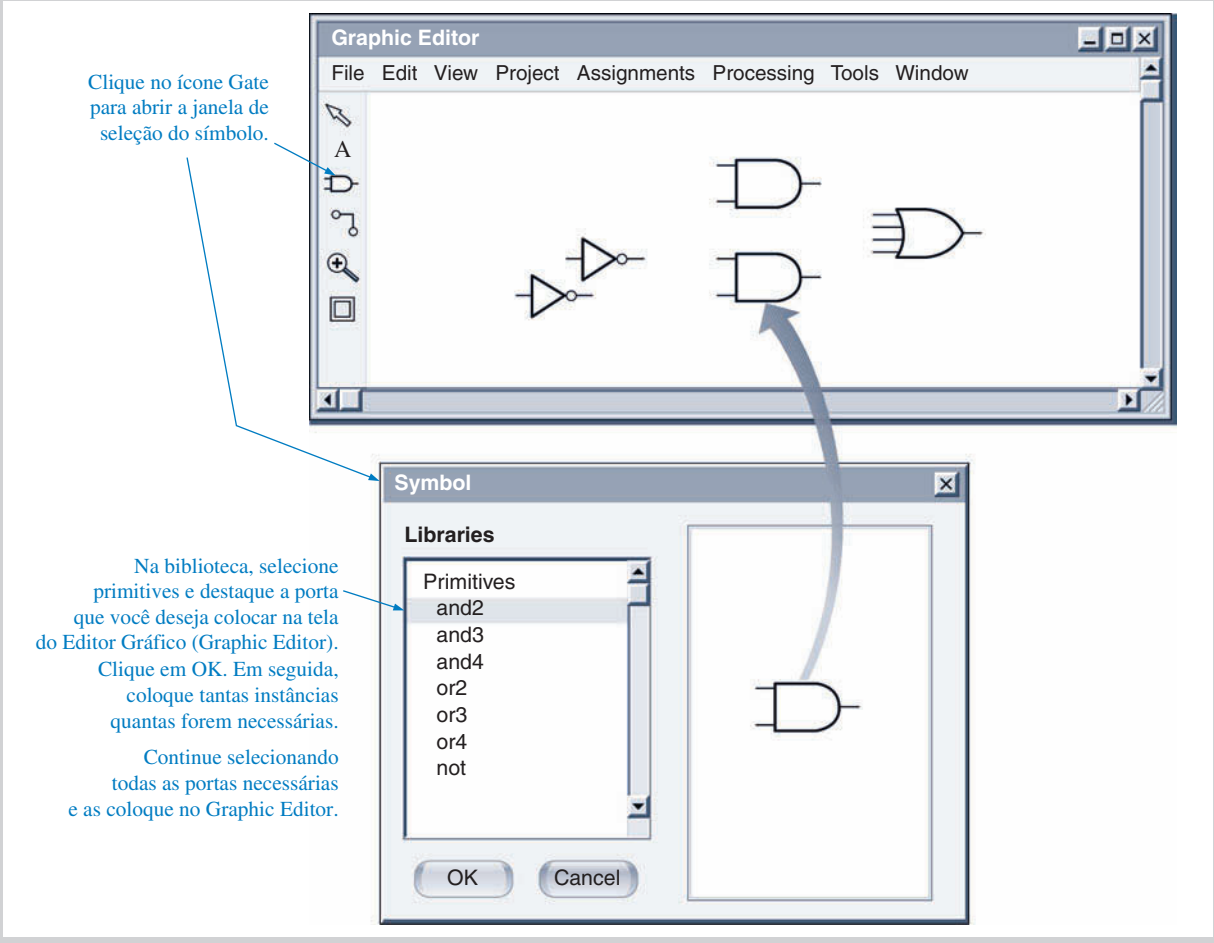
Cada um dos sete circuitos lógicos é inserido individualmente, convertido para o símbolo de um bloco e salvo. Quando todos os sete circuitos forem inseridos, cada bloco será inserido na tela no modo gráfico. Todos os símbolos dos blocos são então conectados às entradas e saídas. Saiba que essa é uma descrição genérica, mas que ilustra o conceito de algumas das principais ferramentas encontradas na maioria dos softwares, tais como Altera Quartus II e Xilinx ISE.

Inserção do Projeto da Lógica do Segmento *a*

Após abrir o software, criamos um projeto para a lógica de 7 segmentos e abrimos a tela do Editor Gráfico (Graphic Editor). Para colocar o símbolo das portas lógicas na tela, clique, com o mouse, no ícone da porta conforme indicado na Figura 11–80. Aparecerá uma tela com o símbolo que permite selecionar as portas que queremos obter a partir da biblioteca do software. As portas lógicas são chamadas de **primitivas** (*primitives*) e podem ser selecionadas a partir de uma lista quando passamos para o cabeçalho de primitivas.

Para o segmento *a*, duas instâncias de porta AND de 2 entradas são selecionadas e colocadas na tela como mostrado. Em seguida, uma porta OR de 4 entradas é selecionada e uma instância é colocada na tela. Finalmente, o inversor (NOT) é selecionado e duas instâncias dele são colocadas na tela.

Em seguida, na janela *Symbol* (símbolo), selecione *Pins* (pinos) a partir da biblioteca. Qualquer pino de entrada ou saída pode ser especificado, conforme ilustrado na Figura 11–81. Para esse circuito em particular, quatro pinos de entrada e um de saída foram colocados na tela do Graphic Editor, conforme mostra a Figura 11–82.



▲ **FIGURA 11–80**

Ilustração da seleção e inserção dos símbolos lógicos no Editor Gráfico (Graphic Editor).

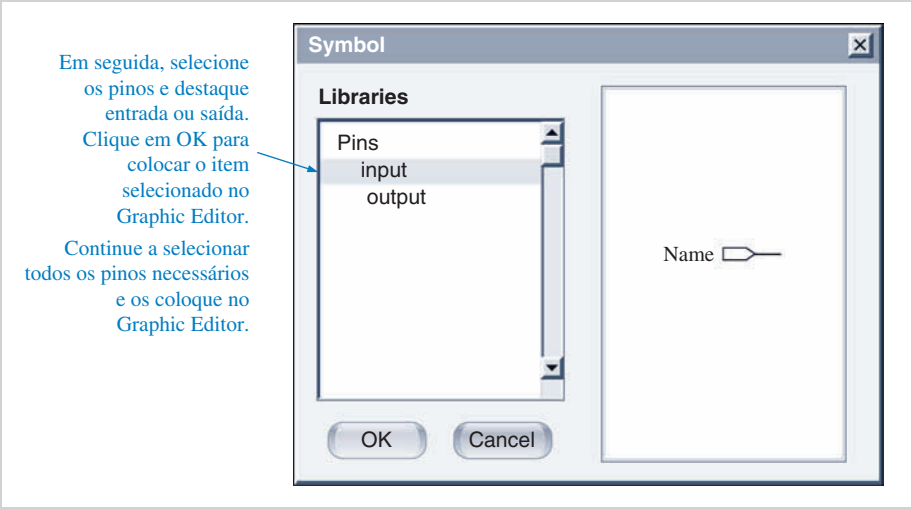


FIGURA 11–81
Seleção dos pinos de entrada e saída na janela *Symbol*.

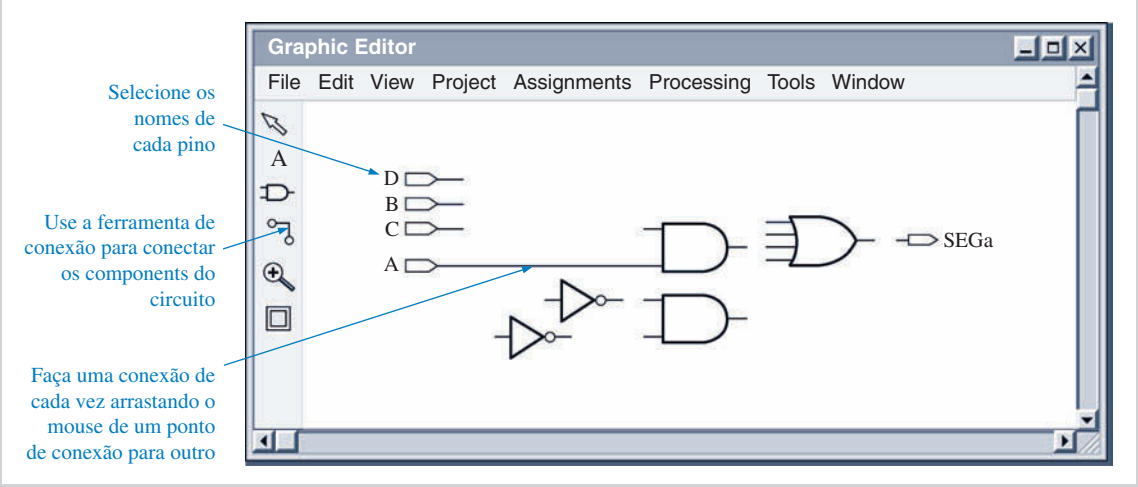


FIGURA 11–82
Colocação dos pinos e realização das conexões do circuito no Graphic Editor.

O esquema completo para a lógica do segmento *a* é mostrado na Figura 11–83.

Compilação do Projeto

Após a inserção do projeto, o próximo passo é compilá-lo. O compilador é uma ferramenta de software que gerencia o processo do fluxo do projeto. A **ferramenta ajustador** no compilador seleciona as melhores interconexões, atribuições de pinos e atribuições de células lógicas para encaixar o projeto no dispositivo destino selecionado. Geralmente, aparece uma caixa de diálogo do compilador indicando o progresso da operação através de uma barra gráfica juntamente com a porcentagem realizada da operação, conforme mostra a Figura 11–84. Uma indicação aparecerá para mostrar se a compilação foi realizada, ou não, com sucesso.

Simulação Funcional

A maioria dos pacotes de software tem pelo menos dois tipos de ferramentas de simulação: simulação funcional e simulação de temporização. A **simulação funcional** verifica a funcionalidade do circuito lógico e deve ser feita logo que o circuito tenha sido compilado com sucesso.

O primeiro passo para a realização dessa simulação é especificar um sinal como entrada ou saída associando um nome a ele. Em seguida, cria-se uma forma de onda de cada vez selecionando o intervalo de tempo e especificando os níveis como ALTO (1) ou BAIXO (0). Isso é feito intervalo por intervalo até que a forma de onda esteja completa. Isso se repete para todas as outras formas de onda de entrada, conforme indicado na Figura 11–85.

Quando todas as formas de onda de entrada estiverem especificadas, a simulação é iniciada e uma forma de onda de saída é gerada, conforme mostra a Figura 11–85 para a lógica do segmento *a*.

Criação do Símbolo de um Bloco

Uma simulação com sucesso significa que o circuito lógico funciona conforme esperado a partir do ponto de vista funcional; ou seja, a lógica está correta.

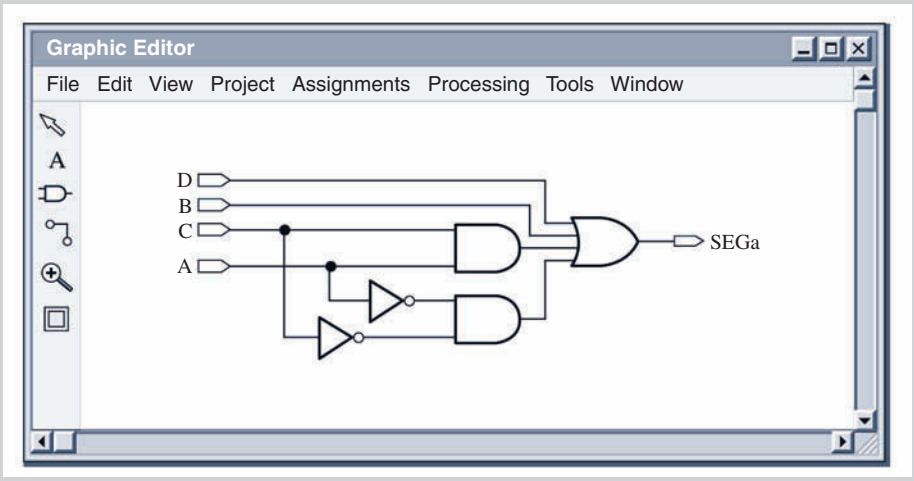


FIGURA 11–83

Diagrama lógico completo do circuito lógico para o segmento *a*.

FIGURA 11–84

Caixa de diálogo do compilador (*compiler*).

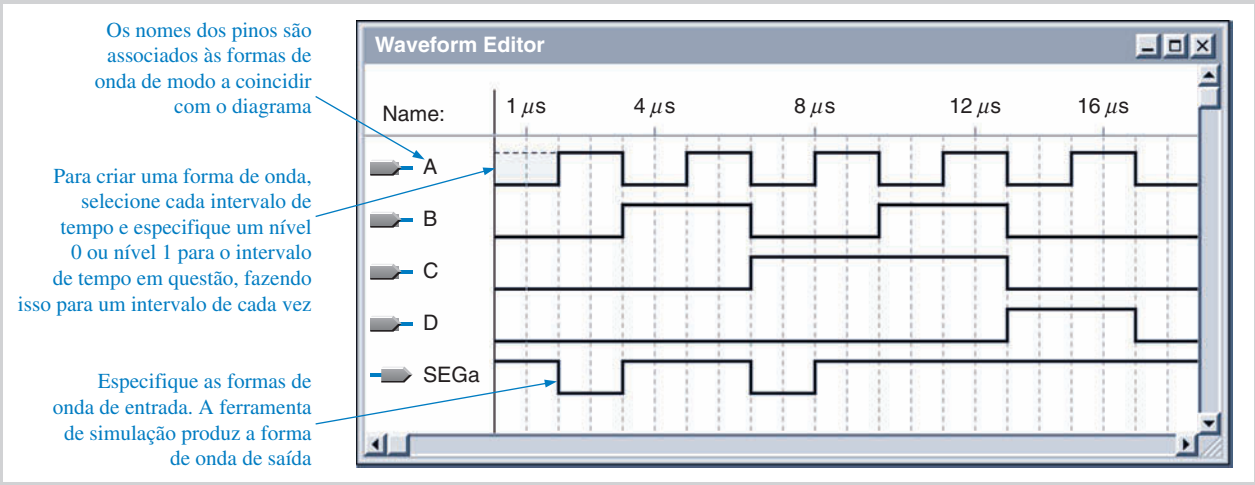


FIGURA 11–85

Formas de onda de entrada e a forma de onda de saída resultante para a lógica do segmento *a*.

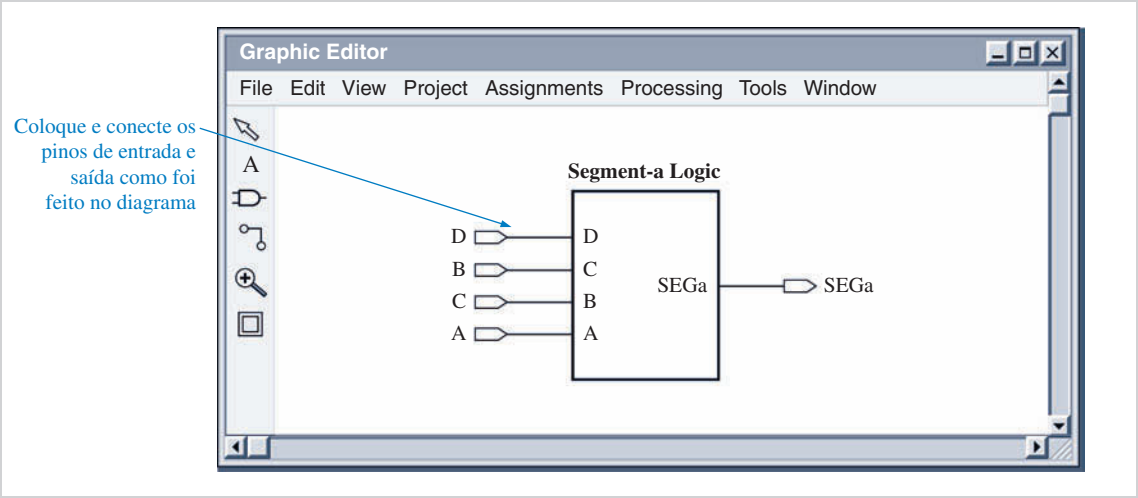
O próximo passo é converter o esquema lógico no símbolo de um bloco, conforme ilustra a Figura 11–86, e salvá-lo para usar mais tarde.

Uma vez que o símbolo do bloco esteja salvo, ele pode ser acessado para uso no projeto final.

Inserção do Projeto para os Segmentos de *b* a *g*

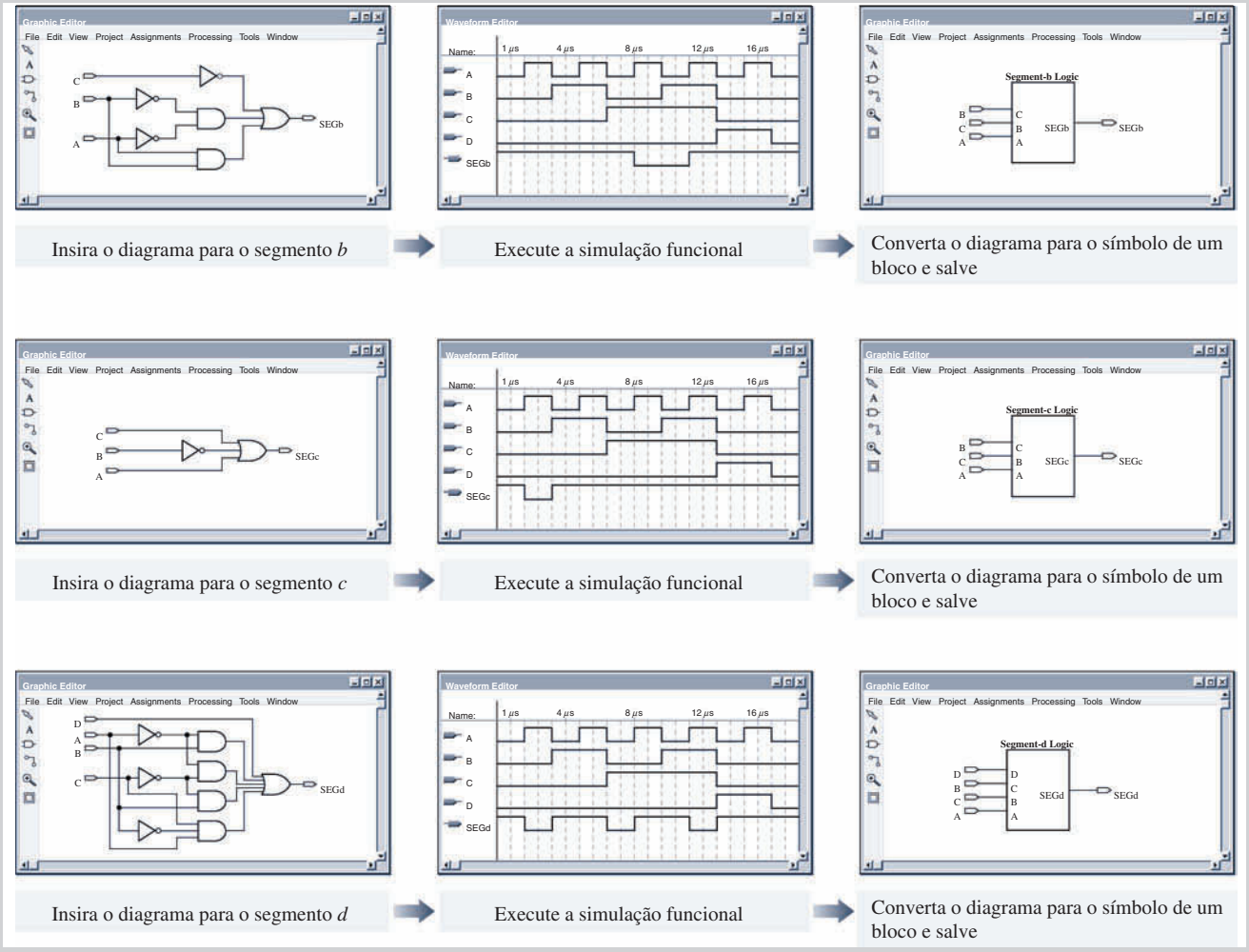
O mesmo procedimento geral que foi usado para inserir, simular e salvar como o símbolo de um bloco a lógica do segmento *a* é repetido para cada um dos outros seis circuitos lógicos. A Figura 11–87 mostra as telas para a lógica dos segmentos *b*, *c* e *d*.

A Figura 11–88 mostra as telas para as lógicas dos segmentos e, *f* e *g*. As simulações funcionais para os segmentos e, *f* e *g* são deixadas como atividade.



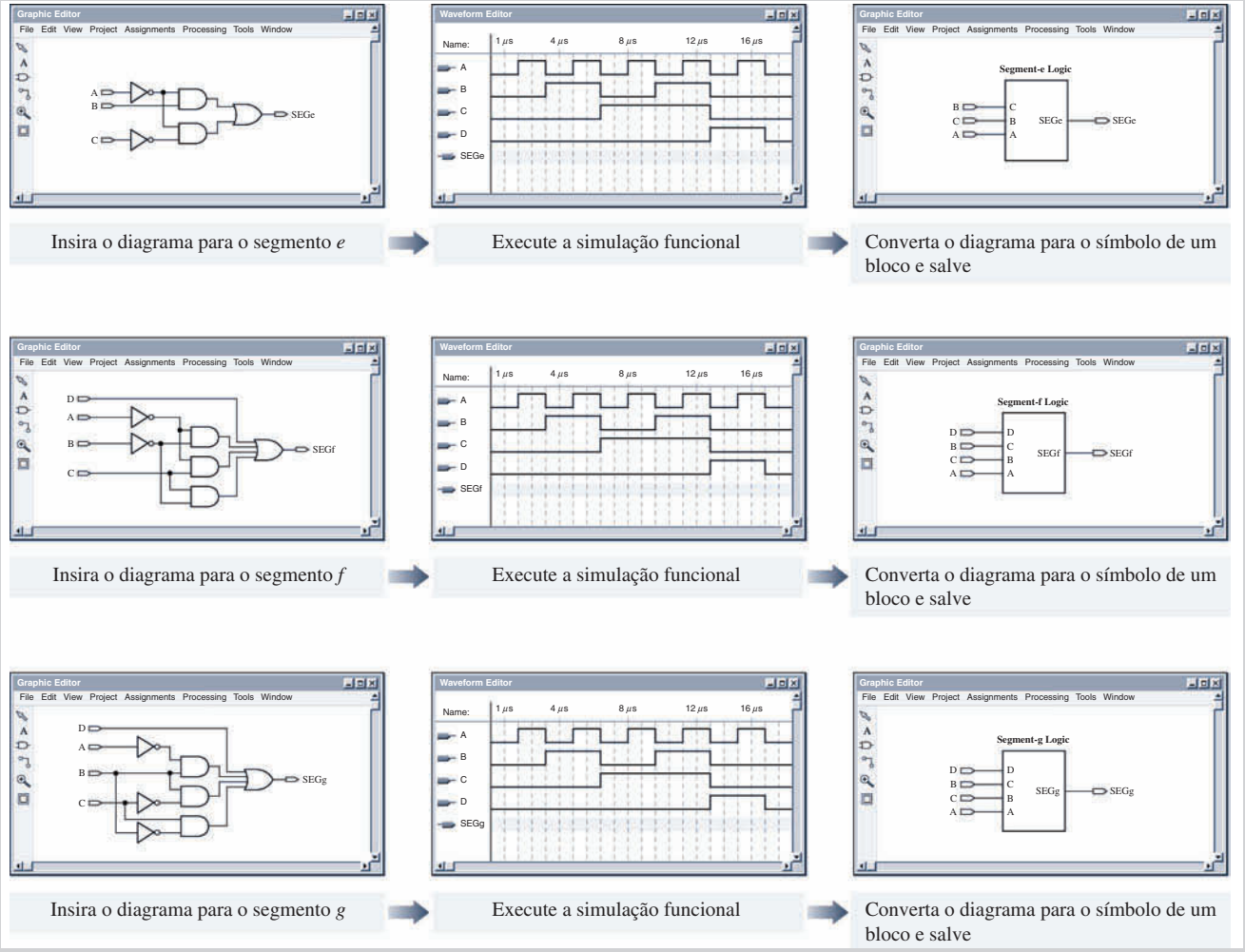
▲ **FIGURA 11–86**

Diagrama lógico para o segmento *a* convertido no símbolo de um bloco.



▲ **FIGURA 11–87**

Telas para os diagramas lógicos dos segmentos (*b*, *c* e *d*).



▲ **FIGURA 11–88**

Telas para os diagramas lógicos dos segmentos (e, *f* e *g*). As formas de onda de saída em cada caso devem ser completadas como um exercício.

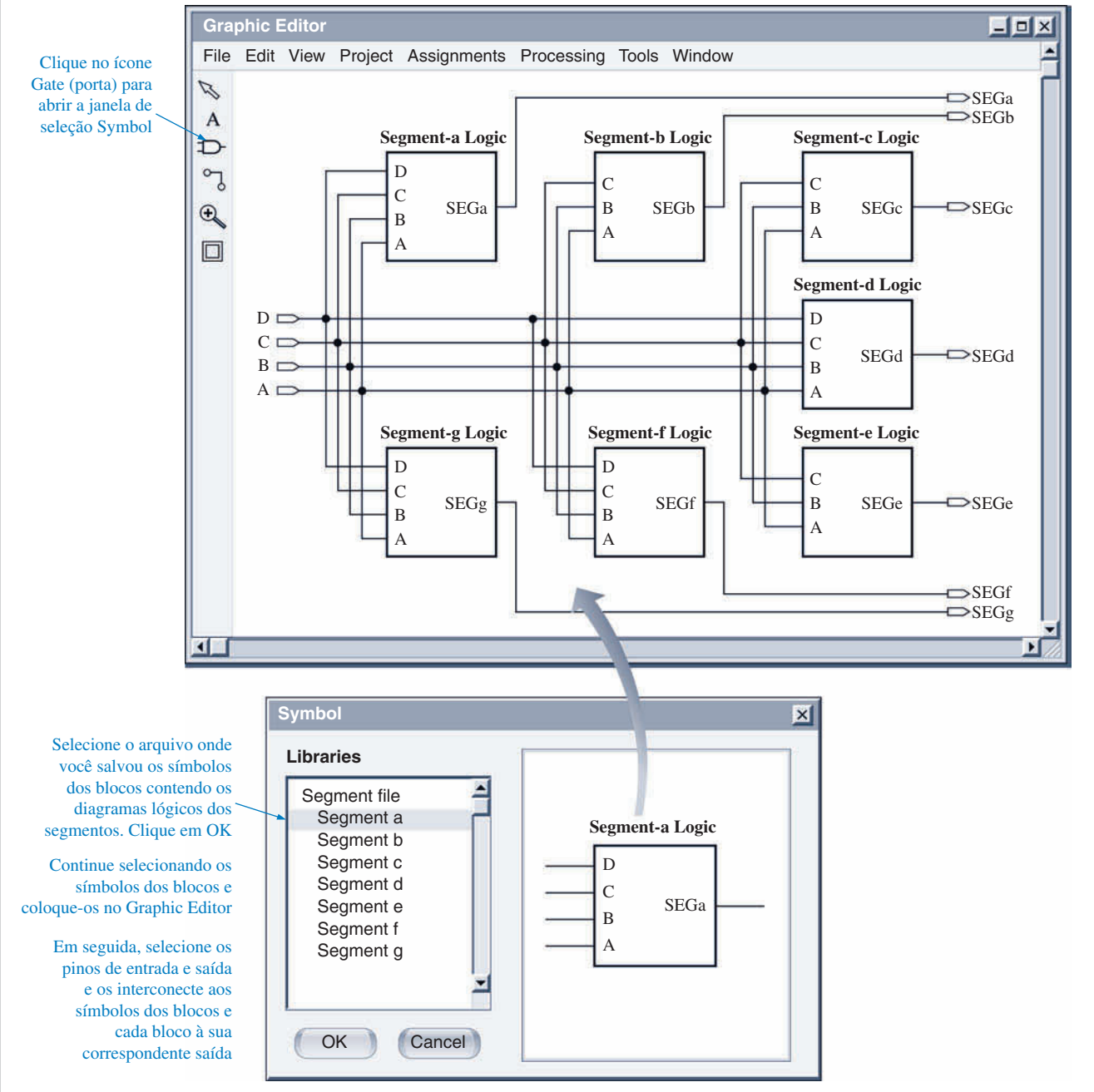
Diagrama em Bloco Final

Todos os símbolos de bloco para os circuitos lógicos dos segmentos foram salvos e agora podem ser usados no diagrama lógico completo dos 7 segmentos. Os símbolos são acessados usando a janela Symbol, da mesma forma como foi feito para a seleção das portas lógicas. Os arquivos armazenados são então abertos aparecendo na forma de lista, como mostra a Figura 11–89. Os pinos de entrada e saída acrescentados e todos os símbolos dos blocos são interconectados de forma a se obter o diagrama lógico de segmentos completo.

Simulação de Temporização

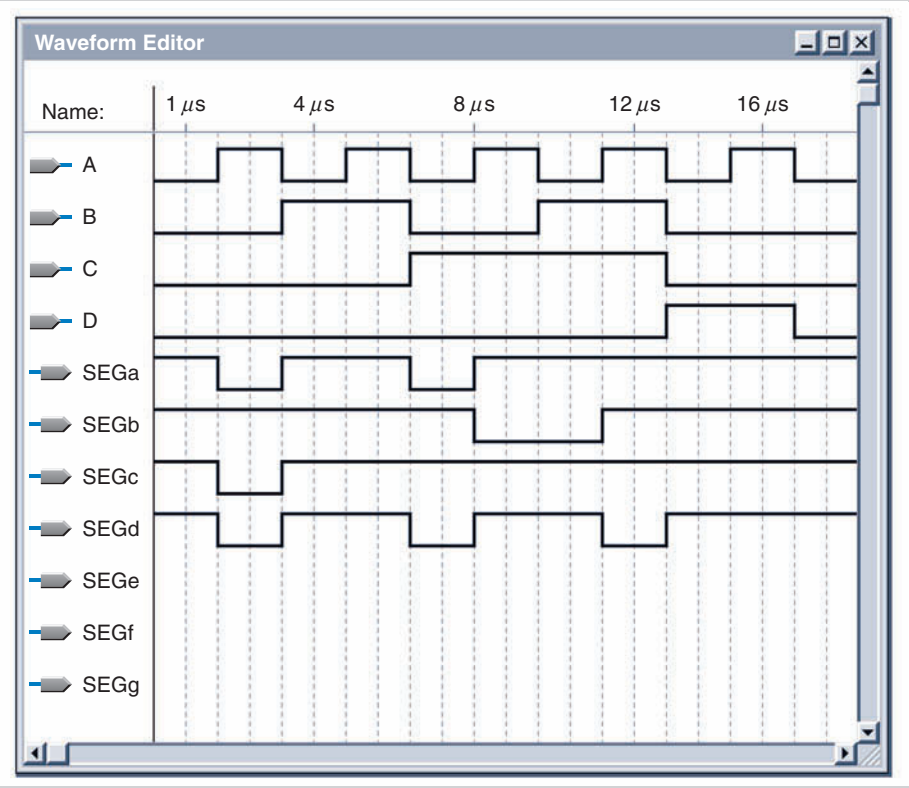
Após a inserção do circuito completo no Graphic Editor, ele tem que ser compilado da mesma forma que cada um dos circuitos lógicos para o correspondente segmento. Após a compilação ter sido feita com sucesso, a simulação de temporização deve ser executada. A **simulação de temporização** leva em conta os atrasos de propagação de cada porta no projeto bem como a funcionalidade do circuito. Todas as formas de onda de entrada são especificadas conforme feito para a simulação funcional. Uma simulação de temporização sem problemas aparentes, tais como *glitches*, aparece no editor de forma de onda (*waveform editor*), conforme mostra a Figura 11–90. Caso existam *glitches*, retornamos ao projeto e procuramos corrigir a condição que poderia ser um problema em potencial.

Uma simulação de temporização feita com sucesso significa que o circuito lógico funciona conforme esperado em termos de funcionalidade e temporização. O próximo passo é converter o diagrama de múltiplos blocos no símbolo de um único bloco, conforme ilustrado na Figura 11–91, para um possível uso futuro.



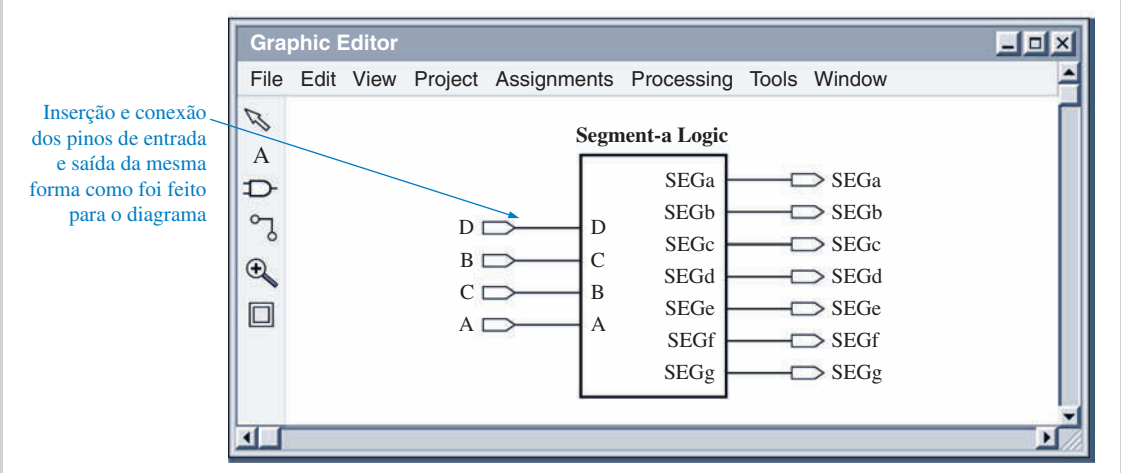
▲ **FIGURA 11–89**

Seleção, inserção e interconexão do circuito lógico dos segmentos no Graphic Editor.



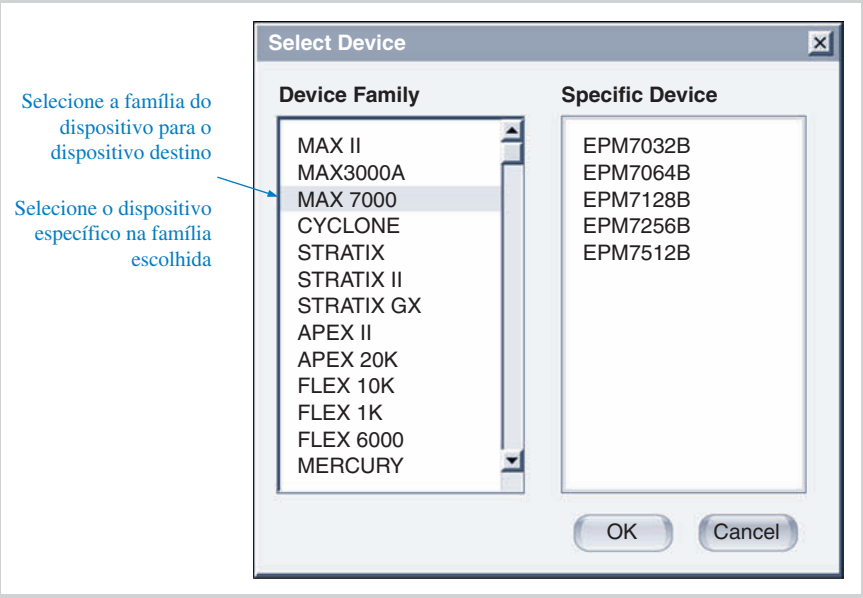
▲ **FIGURA 11–90**

Simulação de temporização ideal para o circuito lógico de 7 segmentos mostrado na Figura 11–89. As formas de onda dos segmentos e, *f* e *g* devem ser completadas como um exercício.

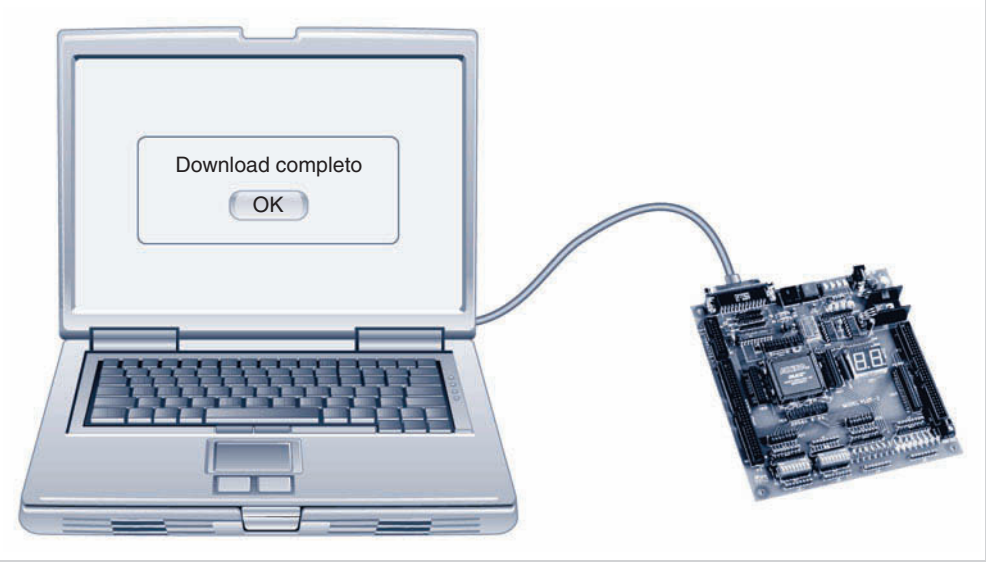


▲ **FIGURA 11–91**

Decodificador BCD para 7 segmentos completo como um símbolo de bloco único.



▶ **FIGURA 11–92**
Seleção do dispositivo destino.



▲ **FIGURA 11–93**
O projeto foi transferido (download) para o dispositivo destino situado na placa de desenvolvimento.

Programação do Dispositivo Destino

Considerando que a simulação de temporização não apresente problemas, o próximo passo é programar o dispositivo destino com o circuito lógico de 7 segmentos.

Comece a seqüência de programação e selecione a interface, tal como a JTAG. Em seguida, selecione o dispositivo destino a partir da janela Select Device (Seleção do Dispositivo), conforme mostra a Figura 11–92. Saiba que uma porcentagem muito pequena da capacidade de um típico dispositivo programável é usada para um circuito do tamanho desse que exemplificamos. Milhares de circuitos com um número similar de portas podem ser programados num único PLD. Normalmente a limitação é o número de entradas e saídas que estão disponíveis no encapsulamento do dispositivo destino.

Os pacotes de software geralmente permitem que associemos os números dos pinos do dispositivo às entradas e saídas. Entretanto, caso não façamos essa operação, a maioria dos softwares a faz automaticamente para nós e disponibiliza uma lista de atribuições.

Após a seleção do dispositivo, o projeto é transferido (download), conforme indicado pela mensagem Download Complete (*Download Completo*) na Figura 11–93.

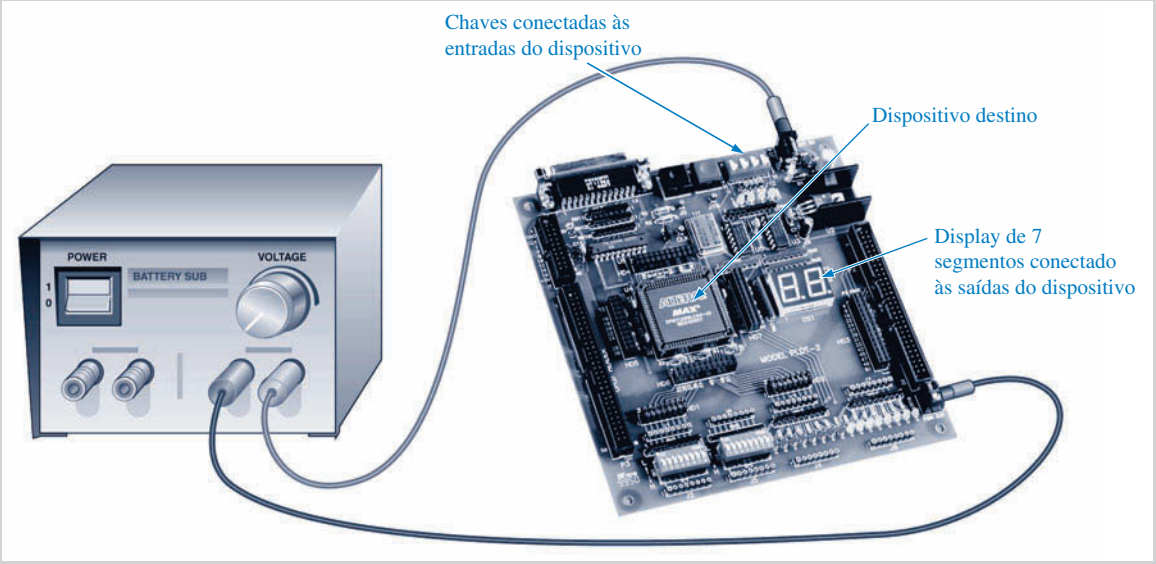
Teste no Próprio Circuito

Após a transferência do projeto para o dispositivo destino, o teste do hardware é geralmente o próximo passo. Antes desse passo, apenas a simulação por software foi feita para verificar a operação. Agora, instrumentos reais são conectados ao circuito na placa de desenvolvimento, sinais de entrada são aplicados e sinais de saída são observados. O método de abordagem depende do tipo e complexidade do circuito lógico transferido para o dispositivo, mas geralmente é necessário uma fonte de sinal, tal como um gerador de funções ou um gerador de padrões, para alimentar as entradas e um osciloscópio, ou um analisador lógico, para observar as saídas.

No caso do circuito decodificador de 7 segmentos transferido para o dispositivo destino, um teste simples pode ser feito usando os recursos disponíveis na placa de desenvolvimento. As entradas podem ser conectadas às chaves na placa e as saídas são conectadas ao display de 7 segmentos, também situado na placa de desenvolvimento. Após a conexão da fonte de alimentação, podemos inserir, por intermédio das chaves, a seqüência do código BCD e observar a saída no display. A maioria das placas de desenvolvimento tem essas características, conforme indicado na Figura 11–94.

Tarefas do Sistema

- *Atividade 1* Verificar se a forma de onda da saída SEGa, mostrada na Figura 11–85, está correta.
- *Atividade 2* Verificar se as formas de onda para os segmentos *b*, *c* e *d*, mostrados na Figura 11–87, estão corretas.
- *Atividade 3* Determinar as formas de onda de saída para os segmentos e, *f* e *g*, para as formas de onda de entrada mostradas nas Figuras 11–88 e 11–90.



▲ **FIGURA 11–94**

Exemplo de dispositivo destino montado numa placa de desenvolvimento sendo testado com chaves e display de 7 segmentos situados na mesma placa.

RESUMO

- Um PAL é um dispositivo SPLD programável apenas uma vez (OTP), o qual consiste em um arranjo de portas AND programável conectado a um arranjo de portas OR fixo.
- A estrutura de um dispositivo PAL permite a implementação de qualquer expressão lógica de soma-de-produtos com um número definido de variáveis.
- O dispositivo GAL é essencialmente um PAL que pode ser programado.
- Em um dispositivo PAL ou GAL, uma macrocélula consiste geralmente em uma porta OR e alguma lógica de saída associada.
- O dispositivo PAL 16V8 é um tipo comum de lógica de arranjo programável.
- O dispositivo GAL 22V10 é um tipo comum de lógica de arranjo genérico.
- Um CPLD é um dispositivo lógico programável complexo que consiste basicamente de múltiplos arranjos SPLD com interconexões programáveis.

- Cada arranjo SPLD em um CPLD é denominado bloco de arranjo lógico (LAB).
- A MAX 7000 é uma família Altera de CPLDs.
- Na família CPLD MAX 7000 a densidade varia de 2 LABs a 16 LABs, dependendo do dispositivo em particular da série, sendo que cada LAB contém dezesseis macrocélulas.
- O dispositivo CPLD MAX II da Altera difere dramaticamente da família MAX 7000 e é conhecido como um CPLD "*post-macrocell*".
- O CPLD MAX II usa tabelas de busca (LUT) em vez de arranjos AND/OR.
- A arquitetura da família CPLD CoolRunner II da Xilinx é baseada na estrutura de um PLA em vez de um PAL.
- A família CoolRunner II contém CPLDs que variam de 32 a 512 macrocélulas.
- Uma macrocélula pode ser configurada em dois modos: o modo combinacional ou o modo registrado.
- Um FPGA (arranjo de porta programável por ação de campo) tem uma arquitetura diferente, não usa arranjos do tipo PAL/PLA e tem densidades muito maiores que CPLDs típicas.
- A maioria dos FPGAs usa tecnologia de processo baseada em anti-fusível ou em SRAM.
- Cada bloco lógico configurável (CLB) em um FPGA é constituído de múltiplos módulos lógicos menores e uma interconexão programável local que é usada para conectar módulos lógicos dentro do CLB.
- FPGAs são baseados em arquitetura LUT.
- LUT significa *tabela de busca* (*look-up table*), que é um tipo de memória que é programada e usada para gerar funções lógicas combinacionais de soma-de-produtos.
- Um núcleo rígido consiste de uma parte da lógica embutida em um FPGA a qual é colocada pelo fabricante para prover uma função específica a qual não pode ser programada.
- Um núcleo flexível consiste de uma parte da lógica embutida em um FPGA a qual tem algumas características programáveis.
- Os projetos próprios dos fabricantes são qualificados de *propriedade intelectual* (IP).
- A Altera produz diversas famílias de FPGAs incluindo Stratix II, Stratix, Cyclone e ACEX.
- A Xilinx tem duas linhas principais de FPGAs, a Spartam e a Virtex, sendo que existem diferentes famílias em cada uma delas.
- O processo de programação é geralmente referido como fluxo de projeto.
- O dispositivo a ser programado é geralmente denominado de dispositivo destino.
- Nos pacotes de software para lógica programável as operações são controladas por um programa denominado compilador.
- Durante o *download*, a seqüência de bits gerada representa o projeto final e é enviado para o dispositivo destino para configurá-lo automaticamente.
- O método BON foi um das primeiras abordagens para a realização de testes automáticos em placas de circuito.
- Outro método para testes em placas de circuito impresso (PCBs) é denominado *flying probe*.
- Um método de testar internamente um dispositivo programável é denominado *boundary scan*, o qual é baseado no padrão JTAG (padrão 1149.1 da IEEE).
- A lógica *boundary scan* em um CPLD consiste de um registrador *boundary scan*, um registrador de *bypass* (desvio), um registrador de instrução e uma porta de acesso para teste (TAP).

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Bed-of-nails Um método para teste automático de placas de circuito impresso no qual a placa é colocada em um acessório de fixação que se assemelha a uma "cama de unhas" que faz contato com os pontos de teste.

Boundary scan Um método de teste interno de um PLD baseado no padrão JTAG (padrão 1149.1 da IEEE).

CLB Bloco lógico configurável; uma unidade de lógica em um FPGA que é constituído de múltiplos módulos lógicos pequenos e uma interconexão programável local que é usada para conectar módulos lógicos dentro do CLB.

Compilador Uma programa aplicativo de um pacote de software de desenvolvimento que controla a operação do software.

CPLD Um dispositivo lógico programável complexo que consiste basicamente de múltiplos arranjos de SPLDs com interconexões programáveis.

Dispositivo destino O dispositivo lógico programável a ser programado.

Download O passo final no fluxo do projeto no qual o projeto lógico é implementado no dispositivo destino.

Ferramenta ajustador Uma ferramenta de software compilador que seleciona as melhores interconexões, designações de pinos, e designações de células lógicas para encaixar um projeto no dispositivo destino selecionado.

Fluxo do projeto O processo ou seqüência de operações realizadas para programar um dispositivo destino.

Flying probe Um método para teste automático de placas de circuito impresso no qual uma ponta de prova, ou mais de uma, se move de um local para outro fazendo contato com pontos de teste.

FPGA Arranjo de portas programáveis por ação de campo; um dispositivo lógico programável que usa a LUT como elemento lógico básico e geralmente emprega a tecnologia de processo baseada em antifusível ou SRAM.

GAL Um tipo reprogramável de SPLD que é similar a um dispositivo PAL exceto que o primeiro usa uma tecnologia de processo reprogramável, tal como a EEPROM (E^2PROM), em vez de fusíveis.

Inserção via esquemático Um método de colocar um projeto lógico em um software usando símbolo esquemáticos.

Inserção via texto Um método de colocar um projeto lógico em um software usando um linguagem de descrição de hardware (HDL).

LAB Bloco de arranjo lógico; um arranjo lógico SPLD em um CPLD.

LUT Tabela de busca; um tipo de memória que pode ser programada para produzir funções de soma-de-produtos.

Macrocélula Parte de um dispositivo PAL, GAL ou CPLD que geralmente consiste de uma ou mais portas OR e alguma lógica de saída associada.

PAL Um tipo de SPLD programável apenas uma vez que consiste de um arranjo programável de portas AND que se conecta a um arranjo fixo de portas OR.

Primitivo Um elemento lógico básico tal como uma porta lógica ou um flip-flop, pinos de entrada/saída, GND e V_{CC}.

Propriedade intelectual (IP) Projetos próprios de um fabricante de dispositivos lógicos programáveis.

Registrado Modo operacional de uma macrocélula que usa um flip-flop.

Simulação de temporização Um processo via software que usa informação dos atrasos de propagação e os dados da *netlist* para testar a operação lógica e a temporização para o pior caso de um projeto.

Simulação funcional Um processo feito por software que testa a operação funcional ou lógica de um projeto.

AUTOTESTE

As respostas estão no final do capítulo.

1. Os dois tipos de SPLD são:
(a) CPLD e PAL **(b)** PAL e FPGA
(c) PAL e GAL **(d)** GAL e SRAM

2. Um dispositivo PAL consiste em
(a) um arranjo AND programável e um arranjo OR programável
(b) um arranjo AND programável e um arranjo OR fixo
(c) um arranjo AND fixo e um arranjo OR programável
(d) um arranjo fixo AND/OR

3. Uma macrocélula consiste em
(a) uma porta OR fixa e outra lógica associada
(b) um arranjo OR programável outra lógica associada
(c) uma porta AND fixa e outra lógica associada
(d) um arranjo AND/OR fixo com um flip-flop

4. O dispositivo 16V8 é um tipo de
(a) CPLD **(b)** GAL
(c) PAL **(d)** FPGA

5. A estrutura AND/OR básica de SPLDs e CPLDs produz tipos de expressões Booleanas conhecidas como
(a) produto-de-somas **(b)** soma-de-produtos
(c) produto-de-complementos **(d)** soma-de-complementos

6. O termo LAB significa
(a) bloco AND lógico **(b)** bloco de arranjo lógico
(c) último bit acionado **(d)** bloco de montagem lógica

7. A denominação MAX 7000 se refere a
(a) uma família de CPLDs **(b)** uma família de SPLDs
(c) uma família de FPGAs **(d)** um tipo de software

8. A denominação CoolRunner II se refere a
(a) uma família de CPLDs **(b)** uma família de SPLDs
(c) uma família de FPGAs **(d)** um tipo de software

9. Os dois modos de operação de uma macrocélula são
(a) entrada e saída **(b)** registrado e seqüencial
(c) combinacional e registrado **(d)** paralelo e compartilhado

10. Quando uma macrocélula é configurada para produzir uma função de soma-de-produtos, ela está no modo
(a) combinacional **(b)** paralelo
(c) registrado **(d)** compartilhado

11. Uma macrocélula típica consiste em
(a) portas, multiplexadores e flip-flop **(b)** portas e um registrador de deslocamento
(c) um contador de código Gray **(d)** um arranjo lógico fixo

12. Baseada na complexidade dos seus blocos configuráveis (CLBs), uma FPGA pode ser classificada como
(a) volátil ou não-volátil
(b) programável ou reprogramável
(c) granulação fina (*fine grained*) ou granulação grossa (*coarse grained*)
(d) plataforma ou embutida

13. FPGAs não-voláteis são geralmente baseadas em tecnologia
(a) fusível **(b)** antifusível
(c) EEPROM **(d)** SRAM

14. Diz-se que uma FPGA com uma função lógica embutida que não pode ser programada é
(a) não-volátil **(b)** de plataforma
(c) de núcleo rígido **(d)** núcleo flexível

15. Projetos de núcleo rígido são geralmente desenvolvidos pelos fabricantes de FPGAs sendo de propriedade destes. Esses projetos são denominados de
(a) propriedade intelectual **(b)** lógica proprietária
(c) projetos de cliente **(d)** padrão IEEE

16. Para a inserção via texto de um projeto lógico,
(a) tem que ser usados símbolos lógicos **(b)** tem que ser usado uma HDL
(c) é usada apenas a álgebra Booleana **(d)** tem que ser usado um código especial

17. Em uma simulação funcional, o usuário tem que especificar
(a) o dispositivo destino **(b)** a forma de onda de saída
(c) as formas de onda de entrada **(d)** uma HDL

18. A saída final da fase de síntese do fluxo de um projeto é
(a) uma *netlist* **(b)** uma seqüência de bits
(c) uma simulação de temporização **(d)** os números dos pinos do dispositivo

19. EDIF significa
(a) formato de intercâmbio de dispositivo eletrônico **(b)** acessório integrado de projeto elétrico
(c) função de entrada destrutível eletricamente **(d)** formato de intercâmbio de projeto eletrônico

20. A *boundary scan* TAP significa
(a) ponto de acesso para teste **(b)** porta de arranjo para teste
(c) porta de acesso para teste **(d)** percurso de acesso de terminal

21. Uma *boundary scan* típica contém células
(a) apenas de flip-flops **(b)** de flip-flops e multiplexadores lógicos
(c) de latches e flip-flops **(d)** de latches e codificador

22. Um método de teste de placa de circuito impresso que usa um acessório de fixação com muitos contatos fixos para os pontos de teste da placa é denominado
(a) tradicional **(b)** *flying probe*
(c) *bed-of-nails* **(d)** *boundary scan*

23. Um método de teste automatizado de placa de circuito impresso que usa um contato de ponto de teste em movimento é denominado
(a) tradicional **(b)** *flying probe*
(c) *bed-of-nails* **(d)** *boundary scan*

24. O padrão JTAG tem as seguintes entradas e saídas
(a) Intest, Extest, TDI, TDO **(b)** TDI, TDO, TCK, TMS
(c) ENT, CLK, SHF, CLR **(d)** TCK, TMS, TMO, TLF

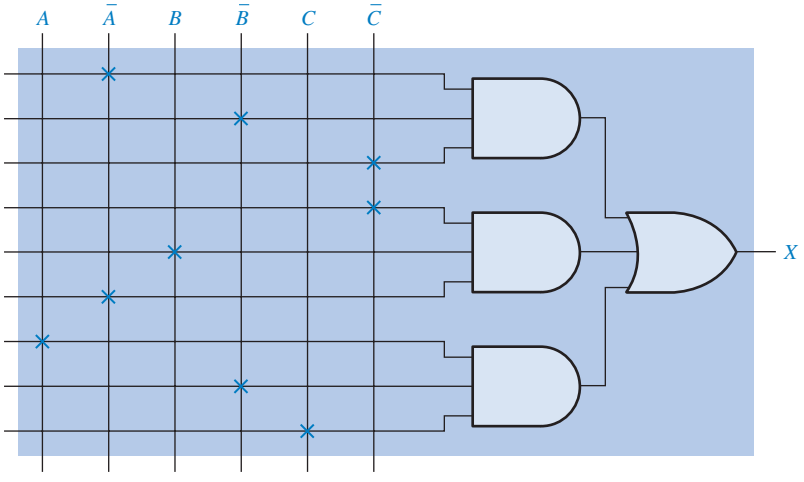
25. O acrônimo BSDL significa
(a) lógica digital padrão de placa **(b)** transferência *boundary scan*
(c) latch digital biestável **(d)** linguagem de descrição *boundary scan*

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 11–1 Lógica programável: SPLDs e CPLDs

1. Determine a expressão de saída Booleana para o arranjo PAL simples mostrado na Figura 11–95. As marcas na forma de X representam conexões implementadas.

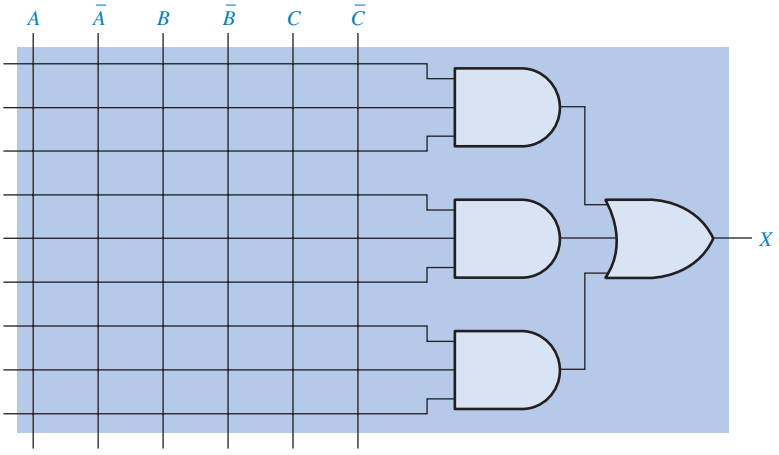


▶ FIGURA 11–95

2. Mostre como o arranjo do dispositivo PAL típico mostrado na Figura 11–96 deve ser programado para implementar cada uma das seguintes expressões de soma-de-produtos. Use um X para indicar uma conexão efetuada. Simplifique as expressões, se necessário, para encaixá-las no arranjo típico de PAL mostrado.

(a) $Y = A\overline{B}C + \overline{A}B\overline{C} + ABC$

(b) $Y = A\overline{B}C + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}BC$



► FIGURA 11–96

3. Interprete cada um dos números dos dispositivos PAL.
(a) PAL16L2 **(b)** PAL12H6

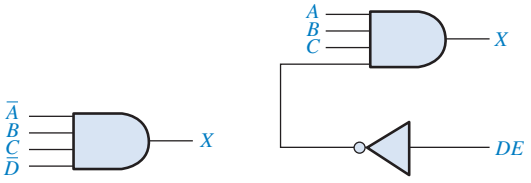
4. Explique como funciona uma saída de polaridade programada.

5. Descreva em que um CPLD difere de um SPLD.

SEÇÃO 11–2 CPLDs Altera

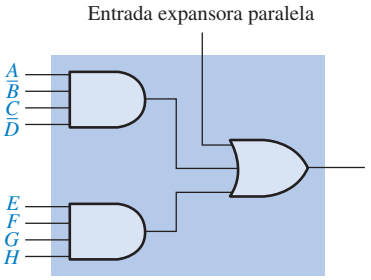
6. Consulte o diagrama em bloco de um MAX 7000, dado na Figura 11–11, e determine o número de
(a) entradas do bloco PIA para um bloco LAB
(b) saídas de um bloco LAB para o bloco PIA
(c) entradas do bloco de controle de I/O para o bloco PIA
(d) saídas de um bloco LAB para um bloco de controle de I/O

7. Determine o termo-produto para a porta AND no arranjo CPLD mostrado na Figura 11–97(a). Se a porta AND for expandida, como mostra a Figura 11–97(b), determine a saída de soma-de-produtos.



► FIGURA 11–97

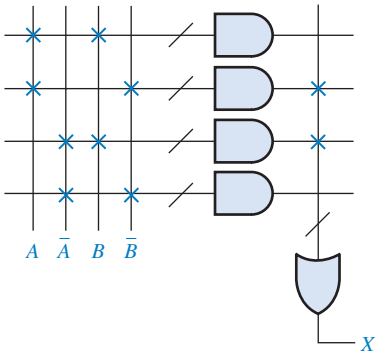
8. Determine a saída do circuito lógico da macrocélula mostrada na Figura 11–98 se $AB\overline{C}D + \overline{A}BCD$ for aplicada na entrada expansora paralela.



► FIGURA 11–98

SEÇÃO 11–3 CPLDs Xilinx

9. Determine a saída do PLA visto na Figura 11–99. As marcas em X representam as conexões efetuadas.



▶ FIGURA 11–99

10. Consulte o diagrama em bloco de um CPLD CoolRunner II, mostrada na Figura 11–21, e determine o número de

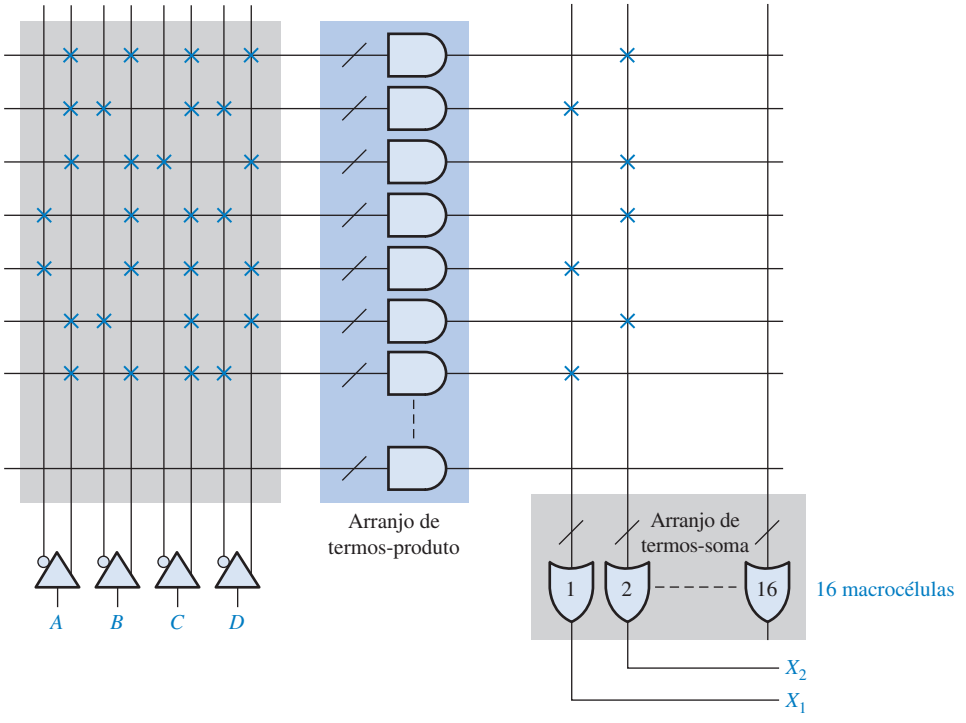
(a) entradas do bloco AIM para um bloco FB

(b) saídas de um bloco FB para o bloco AIM

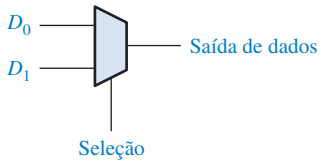
(c) entradas do bloco de I/O para o bloco AIM

(d) saídas de um bloco FB para um bloco de I/O

11. Determine as expressões de saída para X_1 e X_2 a partir das macrocélulas 1 e 2 mostradas na Figura 11–100.



▲ FIGURA 11–100



▶ FIGURA 11–101

SEÇÃO 11–4 Macrocélulas

12. Determine a saída de dados para o multiplexador visto na Figura 11–101 para cada uma das seguintes condições:

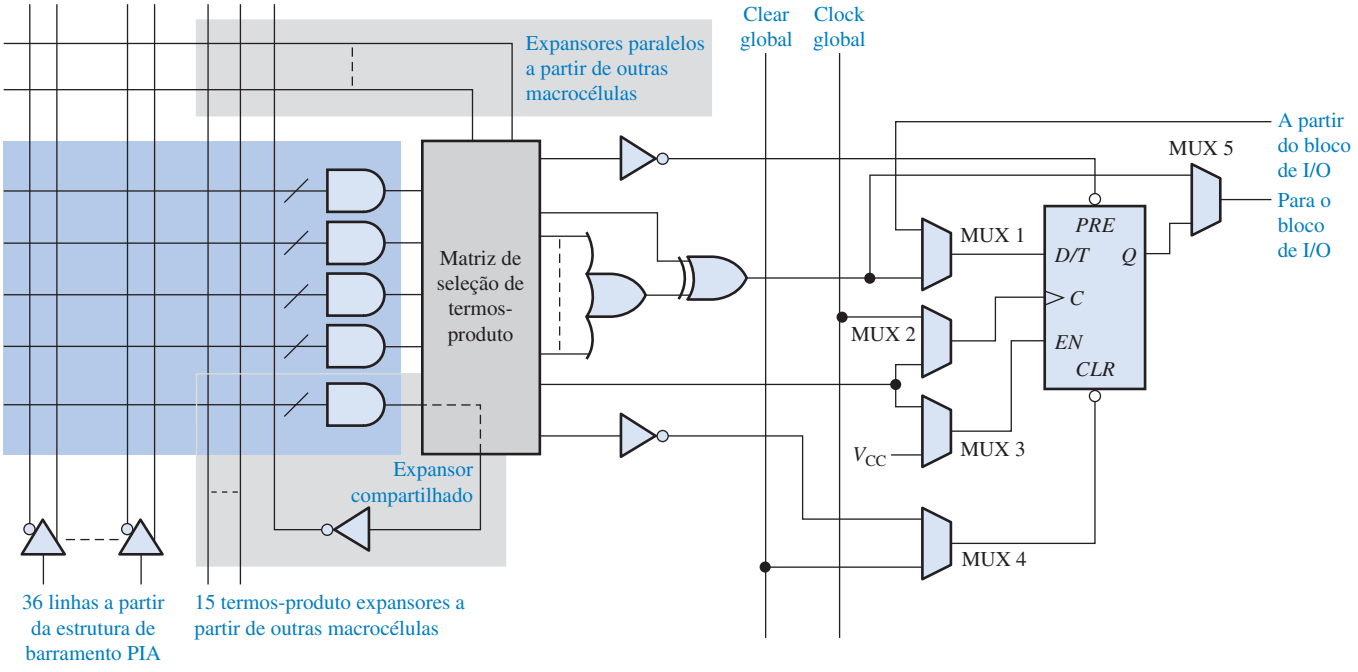
(a) $D_0 = 1$, $D_1 = 0$, Seleção = 0

(b) $D_0 = 1$, $D_1 = 0$, Seleção = 1

13. Determine como a Macrocélula mostrada na Figura 11–102 é configurada (combinacional ou registrada) e o bit de dado que está na saída (para o bloco I/O) para cada uma das seguintes condições. O flip-flop é do tipo *D*. Consulte a Figura 11–101 para ver o arranjo da entrada de dados do MUX.

(a) saída da EX-OR = 1, saída *Q* do flip-flop = 1, a partir do bloco de I/O = 1, seleção do MUX 1 = 1, seleção do MUX 2 = 0, seleção do MUX 3 = 0, seleção do MUX 4 = 0 e seleção do MUX 5 = 0

(b) saída da EX-OR = 0, saída *Q* do flip-flop = 0, a partir do bloco de I/O = 1, seleção do MUX 1 = 1, seleção do MUX 2 = 0, seleção do MUX 3 = 1, seleção do MUX 4 = 0 e seleção do MUX 5 = 1



▲ FIGURA 11–102

14. Para a macrocélula CPLD dada na Figura 11–103, as seguintes condições são programadas: Seleção do MUX 1 = 1, seleção do MUX 2 = 1, seleção do MUX 3 = 01, seleção do MUX 4 = 0 e seleção do MUX 5 = 1, seleção do MUX 6 = 11, seleção do MUX 7 = 11, seleção do MUX 8 = 1 e a saída da porta OR = 1. O flip-flop é do tipo *D* e as entradas do MUX são D_0 na parte superior e D_n na parte inferior.

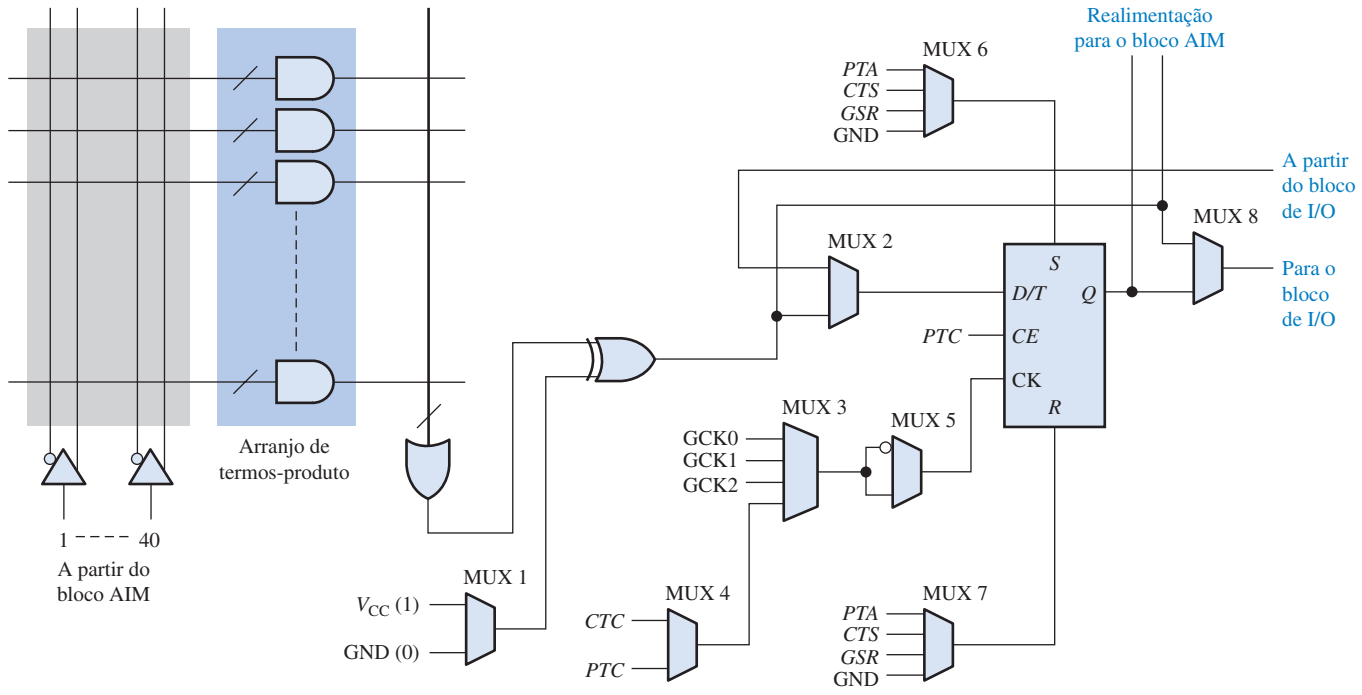
(a) A macrocélula está configurada para lógica combinacional ou registrada?

(b) Qual clock é aplicado no flip-flop?

(c) Qual é o bit de dado na entrada D do flip-flop?

(d) Qual é a saída do MUX 8?

15. Repita o Problema 14 para seleção do MUX 1 = 0.



▲ **FIGURA 11–103**

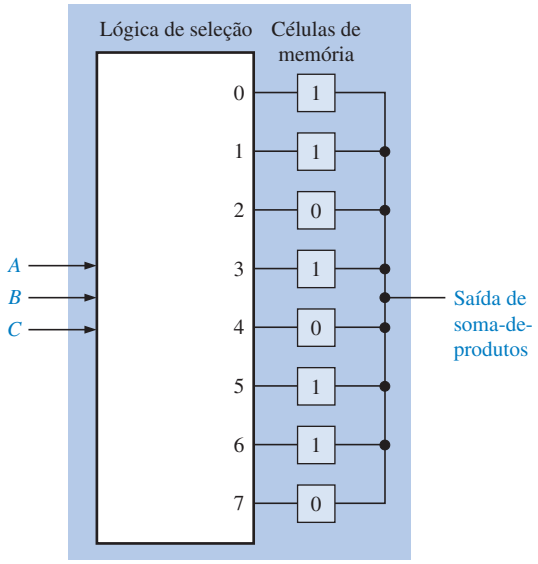
SEÇÃO 11–5 Lógica Programável: FPGAs

16. Geralmente, quais elementos constituem um bloco lógico configurável (CLB) em um FPGA? Quais elementos constituem um módulo lógico?

17. Determine a expressão de saída da LUT para as condições internas mostradas na Figura 11–104.

18. Mostre como reprogramar a LUT mostrada na Figura 11–104 para produzir a seguinte saída de soma-de-produtos:

$\overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$



▶ **FIGURA 11–104**

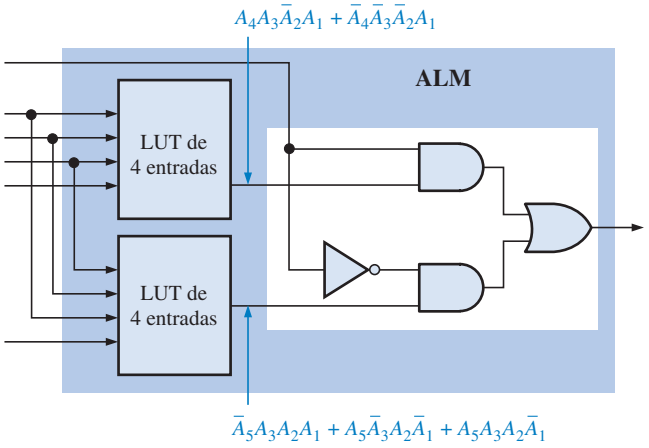
SEÇÃO 11–6 FPGAs Altera

19. Cite os nomes dos elementos básicos que constituem um módulo lógico adaptável (ALM) na FPGA Stratix II.

20. Liste os modos de operação de um ALM.

21. Mostre um ALM configurado no modo normal para produzir uma função de soma-de-produtos de 4 variáveis e uma outra de 2 variáveis.

22. Determine a saída de soma-de-produtos final para bloco ALM mostrado na Figura 11–105.

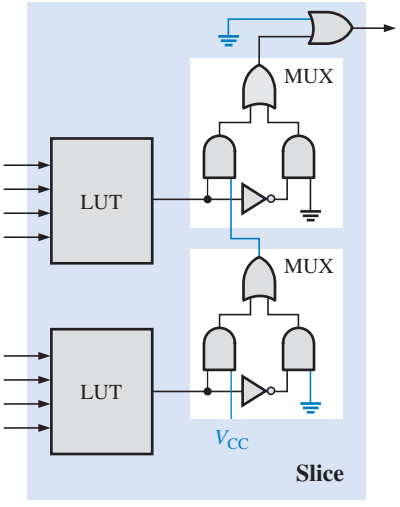


▶ FIGURA 11–105

SEÇÃO 11–7 FPGAs Xilinx

23. Use uma ou mais *slices* mostradas na Figura 11–106 para produzir a função de soma-de-produtos a seguir:

$A_7A_6A_5A_4A_3A_2A_1A_0 + B_7B_6B_5B_4B_3B_2B_1B_0$



▶ FIGURA 11–106

24. Uma *slice* de um FPGA Virtex é mostrada na Figura 11–106. Mostre como uma ou mais dessas *slices* podem ser configuradas para produzir a função de soma-de-produtos a seguir:

$A_7\overline{A}_6A_5A_4 + \overline{A}_3A_2A_1\overline{A}_0 + \overline{B}_7\overline{B}_6B_5B_4 + B_3\overline{B}_2\overline{B}_1B_0$

Considere que os elementos em laranja, bem como as LUTs, sejam reconfiguráveis.

25. Determine o número de *slices* (Figura 11–106) necessário para gerar a expressão:

$A_7A_6A_5A_4A_3A_2A_1A_0$

26. Determine o número de *slices* necessário para gerar a expressão:

$A_7A_6A_5A_4A_3A_2A_1A_0 + B_7B_6B_5B_4B_3B_2B_1B_0 + C_7C_6C_5C_4C_3C_2C_1C_0$

SEÇÃO 11–8 Software para Lógica Programável

27. Mostre o diagrama lógico que poderíamos inserir no Graphic Editor para o circuito por cada um dos programas VHDL a seguir:

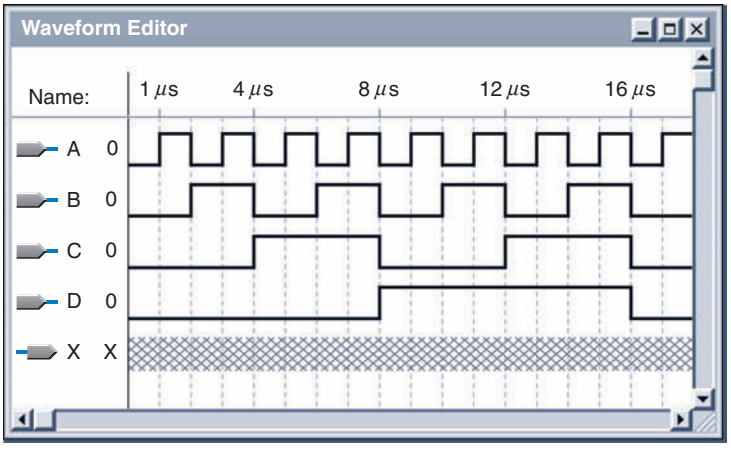
a.
```
entity AND_OR is
    port (A0, A1, A2, A3: in bit; X: out bit);
end entity AND_OR;
architecture LogicFunction of AND_OR is
begin
    X <= (A0 and A1) or (A2 and not A3);
end architecture LogicFunction;
```

b.
```
entity LogicCircuit is
    port (A, B, C, D: in bit; X: out bit);
end entity LogicCircuit;
architecture Function of LogicCircuit is
begin
    X <= (A and B) or (C and D) and
         (A and not B) and (not C and not D);
end architecture Function;
```

28. Mostre o circuito lógico que poderia ser inserido no Graphic Editor para a expressão Booleana a seguir. Simplifique-a, se possível, antes de inseri-la.

$$X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

29. As formas de onda de entrada para o circuito lógico descrito no Problema 28 são conforme mostradas no Waveform Editor (Editor de Formas de Onda) mostrado na Figura 11–107. Determine a forma de onda de saída que é produzida após executar a simulação.



▶ FIGURA 11–107

30. Repita o Problema 29 para a seguinte expressão Booleana:

$$X = \overline{A}BC\overline{D} + A\overline{B}\overline{C}D + ABCD + AB\overline{C}\overline{D} + \overline{A}B\overline{C}D$$

SEÇÃO 11–9 Lógica Boundary Scan

31. Em uma célula *boundary scan* dada, considere o fluxo serial de dados de uma BSC anterior para a próxima. Descreva o que acontece quando os dados passam através da BSC dada.

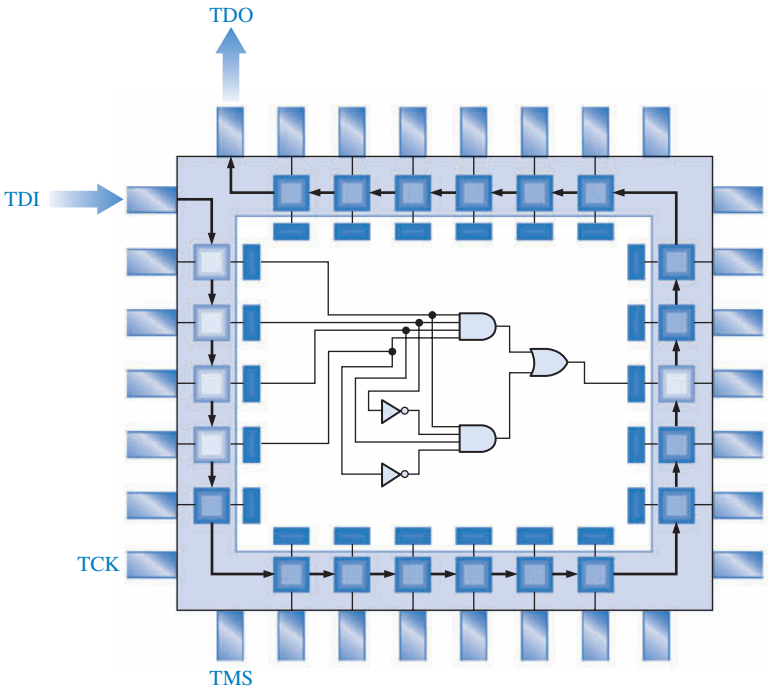
32. Descreva a condição e o que acontece com uma determinada BSC quando o dado passa diretamente da lógica programável interna para um pino de saída do dispositivo.

33. Descreva a condição e o que acontece com uma determinada BSC quando o dado passa diretamente de um pino de entrada do dispositivo para a lógica programável interna.

34. Descreva o percurso dos dados para a transferência de dados da entrada SDI para a lógica programável interna.

SEÇÃO 11–10 Análise de Defeito

35. Descreva um padrão de bit para teste *boundary scan* para testar o circuito lógico programado no dispositivo mostrado na Figura 11–108 para todas as combinações de entrada limite.

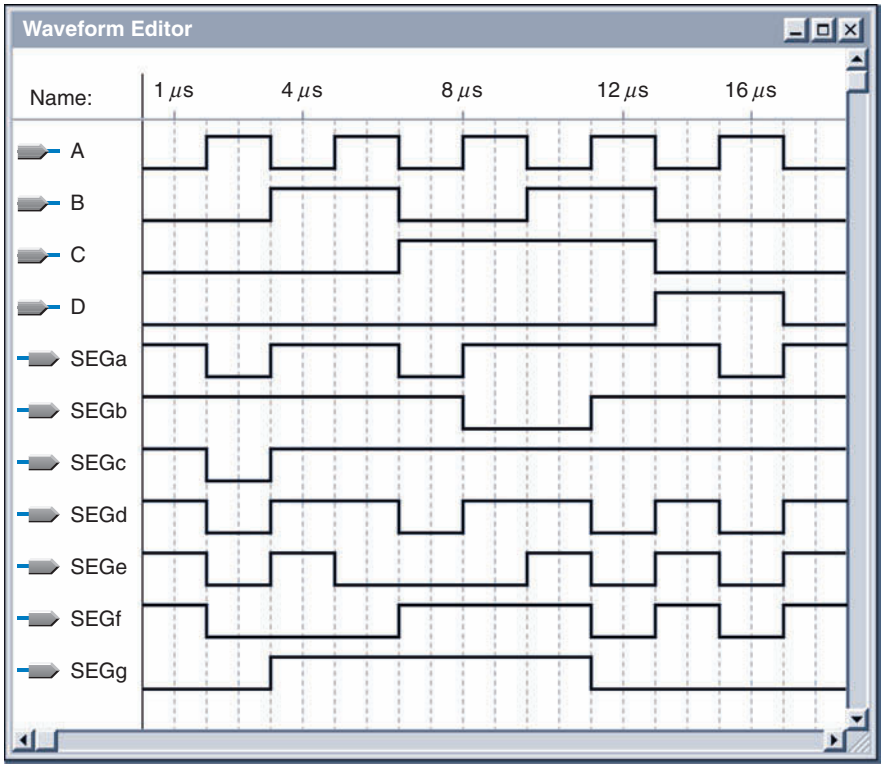


▶ FIGURA 11–108

Aplicações em Sistemas Digitais

36. Se o circuito lógico para sete segmentos mostrado na Figura 11–79 for inserido no Graphic Editor como um diagrama plano, quantos e quais elementos podem ser eliminados?

37. Uma simulação para o circuito lógico de 7 segmentos é mostrada no Waveform Editor na Figura 11–109. Determine que problema pode estar acontecendo com o circuito simulado.



▶ FIGURA 11–109

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 11–1 Lógica Programável: SPLDs e CPLDs

1. PAL: Lógica de Arranjo Programável
2. GAL: Lógica de Arranjo Genérico
3. Um dispositivo GAL é reprogramável. Um dispositivo PAL é programável apenas uma vez.
4. Basicamente, uma macrocélula consiste de uma porta OR e uma lógica de saída associada incluindo um flip-flop.
5. CPLD: Dispositivo Lógico Programável Complexo.

SEÇÃO 11–2 CPLDs Altera

1. LAB: Bloco de Arranjo Lógico
2. Um LAB consiste de 16 macrocélulas na família MAX 7000.
3. Um expansor compartilhado é usado para aumentar o número de termos produtos a partir de uma macrocélula fazendo uma operação AND com termos-soma adicionais (termos-produto complementados) a partir de outras macrocélulas.
4. Um expansor paralelo é usado para aumentar o número de termos-produto a partir de uma macrocélula fazendo uma operação OR com termos-produto a partir de outras macrocélulas em um LAB.
5. A família MAX II é organizado em uma arquitetura linha/coluna e usa LUTs em suas macrocélulas. A família MAX 7000 é organizada em uma arquitetura de coluna tradicional e usa lógica de soma-de-produtos em suas macrocélulas.

SEÇÃO 11–3 CPLDs Xilinx

1. A Altera usa arquitetura PAL. A Xilinx usa arquitetura PLA.
2. Um PLA tem um arranjo AND programável e um arranjo OR programável.
3. Um PAL tem um arranjo OR fixo.
4. FB: Bloco funcional.

SEÇÃO 11–4 Macrocélulas

1. A porta EX-OR é usada como um inversor programável para os dados. Ela pode ser programada para inverter ou não o dado de entrada.
2. Combinacional e registrada.
3. Registrada se refere ao uso de flip-flop.
4. Multiplexador.

SEÇÃO 11–5 Lógica Programável: FPGAs

1. Geralmente, um FPGA é organizado com uma estrutura de interconexões de linha/coluna e usa LUTs em vez de lógica AND/OR para a geração de funções lógicas combinacionais.
2. CLB: Bloco Lógico Configurável.
3. LUT: Tabela de pesquisa. Um tipo programável de memória que é usada para armazenar e gerar funções lógicas combinacionais.
4. Uma interconexão local é usada para conectar módulos lógicos dentro de uma CLB. Uma interconexão global é usada para conectar um CLB com outros CLBs.
5. Um núcleo é a parte da lógica embutida em uma FPGA para prover uma função específica.
6. *Propriedade intelectual* se refere aos projetos de núcleos rígidos que são desenvolvidos pelo fabricante da FPGA sendo de propriedade deste.

SEÇÃO 11–6 FPGAs Altera

1. O LAB (Bloco de Arranjo Lógico) é a unidade de projeto básica na família Stratix II.
2. Tipicamente, existem oito ALMs em um LAB.
3. Uma LUT produz funções lógicas combinacionais em um ALM.
4. Dois.
5. Memória e DSP (processamento de sinais digitais).

SEÇÃO 11–7 FPGAs Xilinx

1. Um CLB consiste de oito células lógicas ou quatro slices.
2. Uma LC (célula lógica) consiste de uma LUT e lógica associada.
3. Uma slice consiste de duas células lógicas (LCs).
4. Uma cadeia em cascata é dois ou mais slices conectadas para expandir uma expressão de soma-de-produtos.
5. ASMBL: bloco modular específico de aplicação.

SEÇÃO 11–8 Software para Lógica Programável

1. Inserção de projeto, simulação funcional, síntese, implementação, simulação de temporização, download.
2. Um computador que executa o software de desenvolvimento do PLD, um equipamento de programação ou uma placa de desenvolvimento e um cabo de interface.
3. Uma *netlist* provê informação necessária para descrever um circuito.
4. A simulação funcional vem antes da simulação de temporização.

SEÇÃO 11–9 Lógica Boundary Scan

1. TDI, TMS, TCK, TRST, TDO
2. TAP: porta de acesso para teste
3. Registrador *boundary scan*, registrador de *bypass*, registrador de instrução e TAP.
4. Transferência de dados de SDI para SDO, transferência de dados da lógica programável interna para um pino de saída do dispositivo, transferência de dados da entrada SDI para a lógico programável interna e transferência de dados da entrada SDI para um pino de saída do dispositivo.

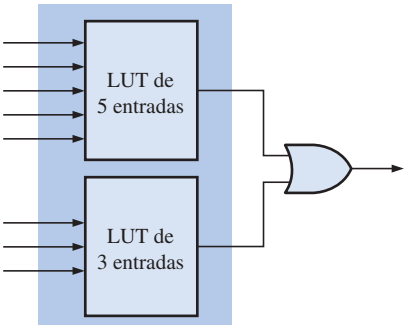
SEÇÃO 11–10 Análise de Defeito

1. O teste BON utiliza um acessório que consiste de um arranjo fixo de pontas de prova de teste (semelhante a unhas) no qual a placa de circuito a ser testada é colocada. Cada ponta de prova de teste faz contato com um ponto de teste na placa de circuito de forma que possa ser feita uma medição.
2. O método BON é limitado pela densidade de dispositivos lógicos programáveis, tornando muitos pontos de contato inacessíveis no circuito da placa.
3. No equipamento com *flying probe*, uma ou mais pontas de prova se movem de um ponto de teste para um outro na PCB segundo uma seqüência programável.
4. *Boundary scan* habilita o teste interno e a programação de um dispositivo lógico programável e testa as interconexões entre dois ou mais dispositivos. E é baseado no padrão JTAG (padrão 1149.1 da IEEE). *Boundary scan* usa lógica específica interna ao dispositivo em teste.
5. Intest e Extest.
6. TDI, TDO, TCK, TMS.
7. BSDL: Linguagem de descrição *boundary scan.*

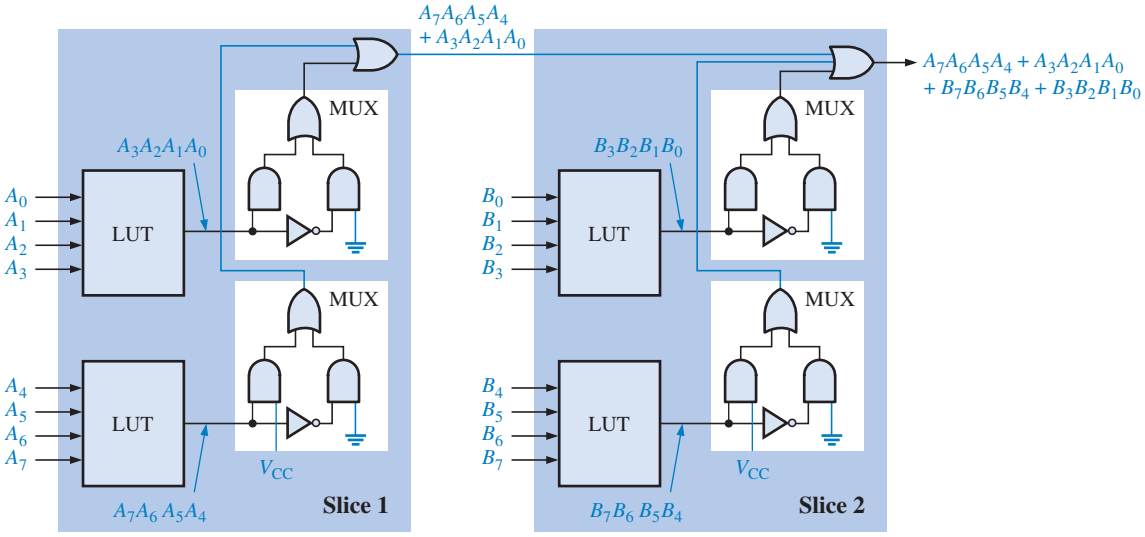
PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

6–1. $X = \overline{B}C + \overline{A}B\overline{C} + \overline{A}\,\overline{B} + C$ **6–2.** Dezesseis **6–3.** Dezesseis; dezesseis

6–4. Veja a Figura 11–110. **6–5.** Veja a Figura 11–111.



▶ FIGURA 11–110



▲ FIGURA 11–111

AUTOTESTE

1. (c)	**2.** (b)	**3.** (a)	**4.** (c)	**5.** (b)	**6.** (b)	**7.** (a)	**8.** (a)	**9.** (c)
10. (a)	**11.** (b)	**12.** (c)	**13.** (b)	**14.** (b)	**15.** (a)	**16.** (a)	**17.** (c)	**18.** (a)
19. (d)	**20.** (c)	**21.** (b)	**22.** (c)	**23.** (b)	**24.** (b)	**25.** (d)		

12

INTRODUÇÃO AOS COMPUTADORES

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Nomear as unidades básicas de um computador
- Nomear os elementos básicos de um microprocessador
- Explicar a operação básica de uma CPU Intel
- Explicar a arquitetura básica de um microprocessador Intel
- Explicar a operação de barramento multiplexado do microprocessador Intel
- Discutir o modelo de software do processador Pentium da Intel
- Descrever um simples programa em linguagem *assembly*

- Descrever os sete grupos de instruções de um processador Intel
- Distinguir entre linguagem *assembly* e linguagem de máquina
- Comparar o atendimento de um I/O por consulta, por interrupção ativada pelo I/O e a interrupção por software
- Descrever as funções dos dispositivos PIC e PPI
- Definir e explicar a vantagem do DMA
- Explicar como funções são interfaceadas através do uso de sistemas de barramentos
- Definir as características básicas e aplicações dos padrões de barramentos internos: PCI e ISA
- Definir as características básicas e aplicações dos padrões de barramentos externos: RS232C, IEEE 1394 (FireWire), USB, IEEE 488 (GPIB) e SCSI

TERMOS IMPORTANTES

- Porta de I/O
- Programa
- CPU
- Interrupções
- Periféricos
- Microprocessador
- Barramento de endereço
- Barramento de dados
- Barramento de controle
- Linguagem de máquina
- Linguagem *assembly*
- Linguagem de alto nível
- Tristate
- Modem
- FireWire
- USB
- GPIB
- SCSI

INTRODUÇÃO

Este capítulo provê uma sinopse introdutória ao estudo dos computadores, microprocessadores e barramentos. Naturalmente, um único capítulo torna-se limitado porque para cada um dos tópicos das seções poderiam ser dedicados um ou mais capítulos. Entretanto, tenha em mente que a finalidade aqui é uma sinopse introdutória. Uma abordagem completa de computadores e microprocessadores acontece normalmente em uma disciplina posterior. Para obter mais informações sobre microprocessadores, incluindo folhas de dados, acesse o site da Intel (www.intel.com).

As famílias de microprocessadores da Intel são discutidas de forma geral. O processador Intel 8086/8088 é usado como "modelo" para ilustrar os conceitos básicos de um microprocessador com as recentes melhorias apresentadas com o Pentium descritas com mais detalhes. O microprocessador 8086/8088 foi a primeira geração da família Intel 80X86. Embora o Pentium seja mais poderoso e contenha características avançadas, existe uma relação em termos de arquitetura e funções básicas tal como a estrutura de registradores.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

12-1 O COMPUTADOR BÁSICO

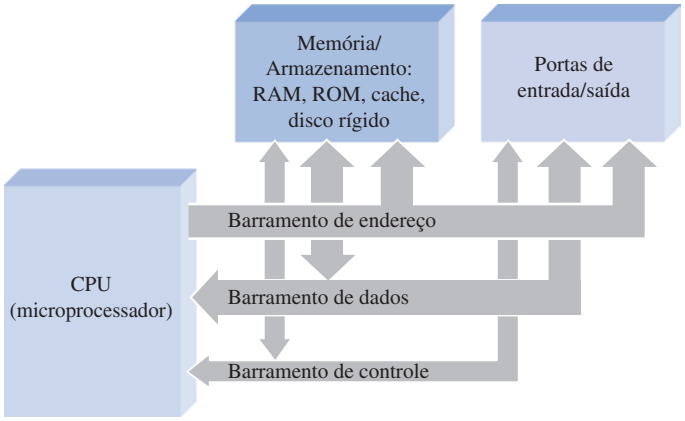
Computadores com finalidades especiais controlam diversas funções em automóveis ou eletrodomésticos, no controle de processos de fabricação em industrias, em jogos eletrônicos para entretenimento e são usados em sistemas de navegação tais como o GPS (*Global Positioning System* – Sistema de Posicionamento Global), para citar algumas áreas. Entretanto, o tipo mais familiar de computador é o computador de propósito geral que pode ser programado para diferentes tipos de tarefas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever os elementos básicos de um computador ■ Discutir a função de cada parte de um computador ■ Explicar o que é um dispositivo periférico

Todos os computadores consistem de blocos funcionais básicos que incluem uma *unidade central de processamento* (CPU), *memória* e *portas de entrada/saída*. Esses blocos funcionais são interconectados por três barramentos internos, conforme mostra o diagrama em bloco da Figura 12–1. Os três barramentos são o *barramento de dados*, *barramento de endereço* e o *barramento de controle*. Os dispositivos de entrada e saída são conectados através de portas de entrada/saída (I/O). Uma **porta** de I/O é uma interface física do computador através da qual os dados trafegam entre o computador os periféricos.

► **FIGURA 12–1**
Diagrama em bloco de um computador básico.



NOTA: COMPUTAÇÃO

Grace Hopper, uma matemática e pioneira na programação, desenvolveu habilidades de análise de defeito consideráveis como oficial naval trabalhando com o computador Harvard Mark I nos anos de 1940. Ela identificou e documentou no livro de registro do Mark I o primeiro *bug* (besouro) real num computador. Era uma maripousa que havia ficado presa em um dos relés eletromecânicos dentro da máquina, provocando um funcionamento errado do computador. Desse fato em diante, quando se pergunta se alguma coisa está sendo feita para solucionar um problema, aqueles que trabalham com computador respondem que estão fazendo "*debugging*" ("tirando os besouros") do sistema. O termo travado e a busca por problemas em um computador (ou outro dispositivo eletrônico), particularmente em software, passou a ser conhecido como *debugging*.

As instruções e os dados são armazenados na memória em locais específicos determinados pelo **programa**, uma lista de instruções idealizada para solucionar um problema específico. Cada local tem um único endereço associado. As instruções são obtidas pela CPU através da colocação de um endereço no barramento de endereço. As instruções são transferidas via barramento de dados à medida que são solicitadas pela CPU. Esta executa as instruções seqüencialmente; freqüentemente, as instruções modificam os dados armazenados na memória ou obtidos a partir de um dispositivo de entrada. Os dados processados podem ser armazenados de volta na memória ou enviados para um dispositivo de saída via barramento de dados. Os sinais no barramento de controle são gerados pela CPU para coordenar todas as operações.

Unidade Central de Processamento (CPU)

A **CPU** é o "cérebro" do computador; ela supervisiona tudo que o computador faz. A CPU é um microprocessador com circuitos associados que controlam a execução dos programas (softwares). Basicamente, a CPU obtém (busca) cada instrução do programa a partir da memória e realiza (executa) essa instrução.

Após a execução de uma instrução, a CPU se move para a próxima e, na maioria dos casos, ela pode operar mais que uma instrução de cada vez. Esse processo de "busca e execução" é repetido até que todas as instruções de um programa específico sejam executadas. Por exemplo, um

programa aplicativo necessita somar uma série de números. As instruções para somar os números estão armazenadas na forma de códigos binários os quais "orienta" a CPU a buscar uma série de números na memória, efetuar a soma deles e armazenar o resultado de volta na memória.

Memórias e Armazenamento

Diversos tipos de memórias são usados em um computador típico. A *RAM* (memória de acesso aleatório) armazena dados em binário e programas temporariamente durante o processamento. Os dados são os números e as outras informações, e os programas são listas de instruções. Os dados podem ser escritos na RAM e lidos a partir dela em qualquer momento. A RAM é volátil, significando que a informação é perdida se a alimentação for desligada ou faltar energia. Portanto, qualquer dado ou programa que necessita ser salvo, deve ser movido para uma memória não-volátil (tal como um CD ou disco rígido) antes de ser desenergizado.

A *ROM* (memória apenas de leitura) armazena um programa do sistema, que é permanente, denominado **BIOS** (*Basic Input/Output System* – sistema de entrada/saída básico) e certas localizações dos programas de sistema na memória. A ROM é não-volátil, o que significa que ela retém o que é armazenado, mesmo quando a tensão de alimentação é *desligada*. Conforme o nome da memória sugere, os programas e dados em uma ROM não podem ser alterados. Algumas vezes ela é conhecida como "firmware" porque o software é permanente para um determinado sistema.

O BIOS é o menor nível do sistema operacional de um computador. Ele contém instruções que "dizem" à CPU o que fazer quando o sistema é energizado; a primeira instrução executada está no BIOS. Ele controla as funções de inicialização básicas que incluem um autoteste e uma autocarga para carregar o restante do sistema operacional em disco. Além disso, o BIOS armazena endereços de programas do sistema que tratam determinadas requisições de periféricos denominadas **interrupções**, que provocam uma parada temporária no atual processamento.

A memória *cache* é uma pequena RAM que é usada para armazenar uma quantidade limitada de dados usados freqüentemente os quais podem ser acessados de forma muito mais rápida em comparação com a RAM principal. A *cache* armazena a informação "à mão" que será usada novamente em vez da CPU ter que recuperá-la da memória principal. A maioria dos microprocessadores tem uma *cache* interna denominada de nível 1, ou simplesmente L1. Uma memória *cache* externa é um chip de memória à parte denominada de nível 2, ou L2.

O *disco rígido* é o principal meio de armazenamento em um computador porque ele pode armazenar grandes quantidades de dados e é não-volátil. O sistema operacional de alto nível (acima do BIOS) bem como os softwares aplicativos e os arquivos de dados são armazenados no disco rígido.

Um *dispositivo removível de armazenamento* é uma parte da maioria dos sistemas de computador. Os tipos mais comuns de meios de armazenamento removíveis são os CDs, disquetes e discos Zip (meio de armazenamento magnético). Os disquetes têm capacidade de armazenamento limitada em cerca de 1,44 MB (megabyte). Os CDs são disponibilizados na forma de CD-ROMs (memória apenas de leitura em CD) e CD-RWs (regravável) e podem armazenar uma enorme quantidade de dados (tipicamente 650 MB). Zip drives armazenam tipicamente 250 MB.

Portas de Entrada/Saída

Geralmente, o computador envia dados para um dispositivo periférico através de uma porta de saída e recebe informações através de uma porta de entrada. As portas de I/O podem ser configuradas pelo software como entrada ou saída. O teclado, mouse, monitor de vídeo, impressora e outros periféricos se comunicam com a CPU através de portas de I/O individuais. As portas de I/O são geralmente classificadas como portas de I/O seriais, com uma única linha de dados, ou paralelas, com múltiplas linhas de dados.

Barramentos

Os periféricos são conectados às portas de I/O do computador com barramentos de interfaces padronizados. Um barramento pode ser visto como uma "estrada" para o tráfego dos sinais digitais a qual consiste de um conjunto de conexões físicas, bem como especificações elétricas para os sinais. Como exemplos de barramentos seriais temos FireWire e USB (barramento serial universal). O barramento paralelo mais comum é simplesmente chamado de *barramento paralelo*, o qual es-

tá conectado à porta de I/O normalmente denominada de porta da impressora (embora essa porta possa ser usada para outros periféricos). Um outro exemplo de barramento paralelo, para conexão de instrumentos de laboratório em um computador, é denominado de barramento de interface de propósito geral (GPIB).

Os três tipos básicos de barramentos internos que interconectam a CPU com a memória e dispositivos de armazenamento e com as portas de I/O são os barramentos de endereço, dados e controle. Esses barramentos são geralmente agrupados no que é chamado de *barramento local*. O barramento de endereço é usado pela CPU para especificar posições, ou endereços, de memória e para selecionar portas de I/O. O barramento de dados é usado para transferir instruções e dados entre a CPU, memórias e portas de I/O. O barramento de controle é usado para transferir sinais de controle gerados ou recebidos pela CPU.

Software de Computador

Além do hardware, outro aspecto principal de um computador é o software. O software comanda o hardware. As duas principais categorias de software usados em computadores são o do sistema e os aplicativos.

Software do Sistema O software do sistema é denominado sistema operacional do computador e permite ao usuário estabelecer uma interface com o computador. Os sistemas operacionais mais comuns usados em computadores desktop e laptop são Windows, Maços e UNIX. Muitos outros sistemas operacionais são usados em computadores com finalidades especiais e em computadores mainframes (de grande porte).

Os softwares do sistema realizam duas funções básicas: gerenciar todo o hardware e o software no computador. Por exemplo, o sistema operacional gerencia e distribui os espaços no disco. Ele também provê uma interface consistente entre softwares aplicativos e o hardware. Isso permite um programa aplicativo funcionar em diversos computadores os quais podem deter detalhes de hardware diferentes.

O sistema operacional do seu computador lhe permite ter diversos programas em execução ao mesmo tempo. Isso é denominado de multitarefa. Por exemplo, podemos usar o processador de texto enquanto fazemos o *download* de algum arquivo da Internet e imprimimos uma mensagem de e-mail.

Softwares Aplicativos Usamos softwares aplicativos para realizar um determinado trabalho ou tarefa. A Tabela 12–1 apresenta uma lista de diversos tipos de softwares aplicativos.

▶ **TABELA 12–1**
Softwares aplicativos

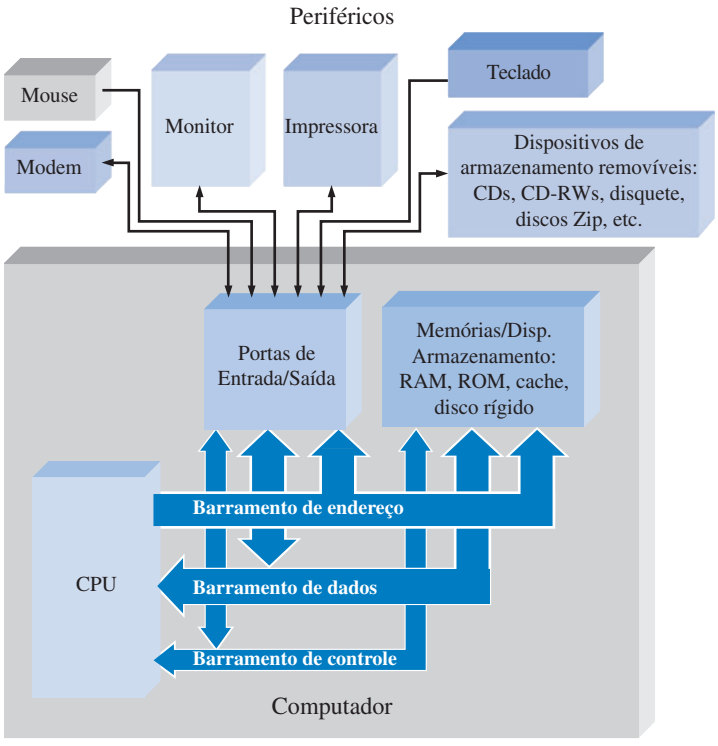
APLICAÇÃO	FUNÇÃO	EXEMPLOS
Processador de texto	Preparação de documentos de texto	Microsoft World, World Perfect
Desenho	Preparação de figuras e desenhos técnicos	CorelDraw, Freehand, Ilustrator
Planilha eletrônica	Manipulação de nmeros e pala vras em um arranjo	Excel, Lotos 123
Editoração eletrônica	Preparação de jornais, panfletos, livros e outros materiais impressos	Quark XPress, Pagemaker
Edição de fotografia	Edição de imagens digitais, mais efeitos especiais nas imagens	Photoshop, Image Expert
Contabilidade	Preparação de imposto, contabilidade	Quickbooks, Turbotax, MYOB
Apresentações	Preparação de apresentação de slides e apresentações técnicas	PowerPoint, Harvard Grafics
Gerenciamento de dados	Manipulação de grandes bases de dados	Filemaker, Access
Multimídia	Edição de vídeo digital, produção de imagens em movimento para apresentações	Premier, Dreamweaver, After Effects
Reconhecimento de voz	Conversão de voz em texto	NaturallySpeaking
Preparação de websites	Ferramentas para criar páginas da web e websites na Internet	FrontPage, Acrobat
Simulação de circuito	Projeto e teste de circuitos eletrônicos	Multisim

Seqüência de Operação Quando ligamos o computador, acontece o seguinte:

1. O BIOS da ROM é carregado na RAM e o autoteste é realizado para verificar os principais componentes e a memória. Além disso, o BIOS provê informação sobre dispositivos de armazenamento, seqüência de boot e outras operações semelhantes.
2. O sistema operacional (tal com o Windows) instalado no disco rígido é carregado na RAM.
3. Os programas aplicativos (tal como o Microsoft Word) estão armazenados no disco rígido. Quando selecionamos um deles, ele é carregado na RAM. Algumas vezes, apenas uma parte é carregada quando necessário.
4. Os arquivos que os aplicativos necessitam são carregados do disco rígido para a RAM.
5. Quando um arquivo é salvo e o aplicativo é fechado (finalizado), o arquivo é escrito de volta no disco rígido e tanto o aplicativo quanto o arquivo são removidos da RAM.

Sistema de Computador

O diagrama em bloco visto na Figura 12–2 mostra os principais elementos num sistema de computador típico e como eles são interconectados. Para um computador realizar uma determinada tarefa, ele tem que se comunicar com o "mundo externo" interfaceando com pessoas, detectando dispositivos ou controlando-os de alguma forma. Para fazer isso, existe um teclado para entrada de dados, um mouse, um monitor de vídeo, uma impressora, um modem e um *drive* de CD na maioria dos sistemas básicos. Esses são denominados de **periféricos**.



◀ **FIGURA 12–2**
Diagrama em bloco básico de um sistema de computador típico incluindo os periféricos comuns. O computador propriamente dito é mostrado no bloco cinza.

SEÇÃO 12–1 REVISÃO

As respostas estão no final do capítulo.

1. Quais são os principais elementos ou blocos de um computador?
2. Qual é a diferença entre RAM e ROM?
3. O que são periféricos?
4. Qual é a diferença entre hardware e software de um computador?

12-2 MICROPROCESSADORES

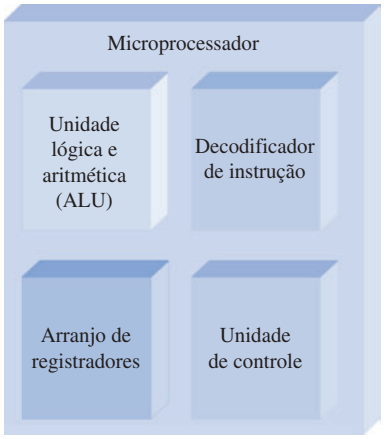
O **microprocessador** é um circuito integrado digital que pode ser programado com uma série de instruções para executar diversas operações sobre os dados. Um microprocessador é a CPU do computador. Ele pode realizar operações lógicas e aritméticas, mover dados de um local para outro e tomar decisões baseadas em certas instruções.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever os elementos básicos de um microprocessador ■ Discutir os barramentos do microprocessador ■ Discutir o conjunto de instruções de um microprocessador

Elementos Básico

Um microprocessador consiste de algumas unidades, cada uma projetada para uma função específica. O projeto e a organização das unidades específicas são denominados de arquitetura (não confundir o termo com o elemento VHDL). A arquitetura determina o conjunto de instruções e o processo para a execução dessas instruções. As quatro unidades básicas que são comuns a todos os microprocessadores são a unidade lógica e aritmética (ALU), o decodificador de instrução, o arranjo de registradores e a unidade de controle, conforme mostra a Figura 12–3.



▶ **FIGURA 12–3**

Unidade Lógica e Aritmética A **ALU** é o principal elemento de processamento do microprocessador. Ela é gerenciada pela unidade de controle para realizar operações aritméticas (adição, subtração, multiplicação e divisão) e operações lógicas (NOT, AND, OR e EX-OR), bem como diversas outras operações. Os dados para a ALU são obtidos a partir do arranjo de registradores.

Decodificador de Instrução O decodificador de instrução pode ser considerado como parte da ALU, embora o tratemos como uma função separada nessa discussão porque as instruções e a decodificação delas são importantes para a operação do microprocessador. O microprocessador realiza uma determinada tarefa definida pelo programa que consiste de uma lista de instruções armazenadas na memória. O decodificador de instruções toma cada instrução em binário, na ordem em que elas aparecem na memória, e as decodifica.

Arranjo de Registradores O **arranjo de registradores** é um conjunto de registradores internos ao microprocessador. Durante a execução de um programa, os dados e os endereços de memória são temporariamente armazenados em registradores que constituem esse arranjo. A ALU pode acessar os registradores de forma bastante rápida, tornando a execução do programa mais eficiente. Alguns registradores são classificados como de propósito geral, significando que eles podem

ser usados para qualquer finalidade determinada pelo programa. Outros registradores têm capacidades e funções específicas e não podem ser usados como registradores de propósito geral. Todavia, outros registradores são denominados de invisíveis ao programa, usados apenas pelo microprocessador não estando disponíveis para o programador.

Unidade de Controle A **unidade de controle** é encarregada do processamento das instruções uma vez decodificadas. Essa unidade provê a temporização e sinais de controle para transferir dados para dentro e para fora do microprocessador e para sincronizar a execução de instruções.

Barramentos do microprocessador

Os três barramentos mencionados anteriormente são as conexões pelas quais o microprocessador faz mover instruções, endereços e dados.

Barramento de Endereço O **barramento de endereço** é uma "via de mão única" através da qual o microprocessador envia um código de endereço de uma memória ou outro dispositivo externo. O tamanho, ou a extensão, do barramento de endereço é especificado pelo número de vias ou bits. Os primeiros microprocessadores tinham dezesseis linhas de endereços e podiam endereçar 65.536 (2^{16}) posições diferentes de memória. Quanto mais bits contiverem num endereço, maior o número de posições de memória que podem ser acessadas. O número de bits de endereço tem avançado ao ponto de, no Pentium 4, termos 36 bits de endereço sendo possível acessar acima de 68 G (68.000.000.000) de posições de memória.

Barramento de Dados O **barramento de dados** é uma "via de mão dupla" na qual dados ou códigos de instruções são transferidos para o microprocessador ou os resultados de operações ou cálculos são enviados para fora do microprocessador. Os primeiros microprocessadores tinham barramentos de dados de 8 bits. Os microprocessadores atuais têm barramentos de dados de até 64 bits.

Barramento de Controle O **barramento de controle** é usado pelo microprocessador para coordenar suas operações e se comunicar com dispositivos externos. As linhas do barramento de controle também são usadas para inserir estados de espera especiais para dispositivos mais lentos, evitando assim contenção de barramento (choque de dados), uma condição que pode ocorrer se dois ou mais dispositivos tentarem se comunicar ao mesmo tempo.

Programação de um Microprocessador

Todos os microprocessadores trabalham com um conjunto de instruções que implementam as operações básicas. O Pentium, por exemplo, tem centenas de variações no seu conjunto de instruções divididas em sete grupos básicos.

- Transferência de dados
- Aritméticas e lógicas
- Manipulação de bit
- *Loops* e *jumps* (saltos)
- *Strings*
- Sub-rotinas e interrupções
- Controle

Cada instrução consiste de um grupo de bits (1s e 0s) que é decodificada pelo microprocessador antes de ser executada. Essas instruções em código binário são denominadas linguagem de máquina e é a única linguagem que o microprocessador reconhece. Os primeiros computadores foram programados com instruções escritas em código binário, um trabalho tedioso e propenso a erros. Esse método primitivo de programação em código binário tem evoluído para um formato de maior nível representado por palavras em inglês para formar o que conhecemos como linguagem assembly. Isso será discutido na Seção 12–4.

Progresso Tecnológico

O primeiro microprocessador, o Intel 4004, foi introduzido em 1971. Basicamente, ele era capaz de fazer somas e subtrações com apenas 4 bits de cada vez. Em 1974, o Intel 8080 se tornou o primeiro microprocessador a ser usado como uma CPU em um microcomputador. O chip 8080 tinha 6000 transistores, um barramento de dados de 8 bits e operava com uma freqüência de clock de 2 MHz. O 8080 podia executar 0,64 milhões de instruções por segundo (MIPS). A família Intel tem evoluído desde o 8080, passando por vários processadores, até o Pentium 4. Esse último microprocessador (ele pode não ser mais o último no momento em que o leitor estiver apreciando esse texto) tem cerca de 42.000.000 de transistores no chip e um barramento de dados de 64 bits. Ele opera com uma freqüência de clock acima de 3 GHz e tem um desempenho de aproximadamente 1700 MIPS. Os conjuntos de instruções também mudaram drasticamente, mas o Pentium 4 executa qualquer código de instrução que o 8086 executa, o dispositivo de 1979 que surgiu após o 8080.

O número de transistores disponíveis tem alto impacto no desempenho e nos tipos de tarefas que um microprocessador é capaz de fazer. Por exemplo, ao se ter um número maior de transistores no chip, possibilitou uma tecnologia denominada *pipelining*. Basicamente, essa tecnologia permite que mais de uma instrução seja executada ao mesmo tempo. Além disso, os microprocessadores modernos têm múltiplos decodificadores de instrução, cada um com o seu próprio *pipeline*. Isso permite que diversos fluxos de instruções sejam processados simultaneamente.

SEÇÃO 12–2 REVISÃO

1. Quais são os quatro elementos básicos de um microprocessador?
2. Quais são os três tipos de barramentos em um microprocessador?
3. Qual é a função de um microprocessador em um microcomputador?
4. Quais são as três operações básicas que um microprocessador realiza?
5. O que é *pipelining*?

12-3 UMA FAMÍLIA ESPECÍFICA DE MICROPROCESSADOR

A família de microprocessadores Intel original tem sofrido grandes transformações ao longo dos anos desde o 8086/8088 até a família Pentium, tanto em velocidade quanto em complexidade. Entretanto, o conjunto básico de registradores e outras características do 8086/8088 foram retidos (e expandidos) através do processo evolucionário de forma que todos os processadores Intel posteriores respondem às mesmas instruções (bem como a diversas instruções novas) conforme os dispositivos originais. Esta seção começa com uma introdução limitada dos conceitos básicos da arquitetura, operação e programação de um microprocessador. A seção termina com considerações gerais das principais mudanças na estrutura de registradores que forma o modelo de software do novo processador. A abordagem é para mostrar um processador básico e discutir as melhorias na linha Intel conforme ela evoluiu.

Ao final do estudo desta seção você deverá ser capaz de:

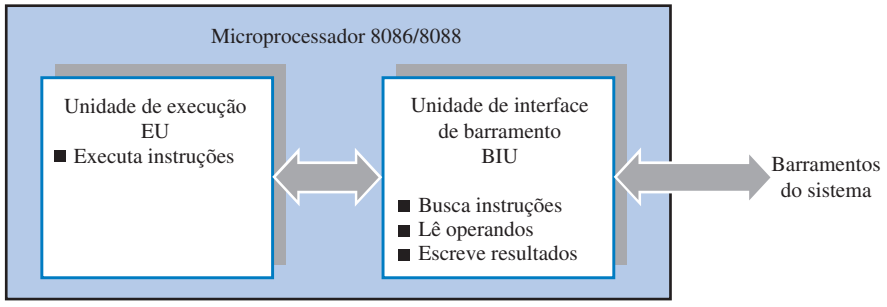
■ Discutir a operação de um microprocessador básico ■ Descrever a unidade de interfaceamento de barramento ■ Citar a finalidade dos registradores de segmento ■ Citar a finalidade do ponteiro de instruções ■ Descrever a unidade de execução ■ Descrever o conjunto geral de registradores ■ Citar a finalidade do registrador de flags ■ Discutir o modelo de software do processador Pentium

Operação básica

Um microprocessador executa um programa seguindo ciclicamente três passos:

1. Buscar uma instrução na memória e colocá-la na CPU.
2. Decodificar a instrução; se outras informações forem requeridas pela instrução, buscar as outras informações. No passo da decodificação, o contador de programa é atualizado apontando para a próxima instrução.
3. Executar a instrução (fazer o que a instrução "diz"). Os resultados são enviados para os registradores e para a memória durante esse passo.

A arquitetura do microprocessador 8086/8088 é dividida em duas unidades: a unidade de execução (**EU**), que executa instruções, e a unidade de interface de barramento (**BIU**), que faz a interface com os barramentos do sistema e busca as instruções, lê os operandos e escreve resultados. Essas unidades são mostradas na Figura 12–4.



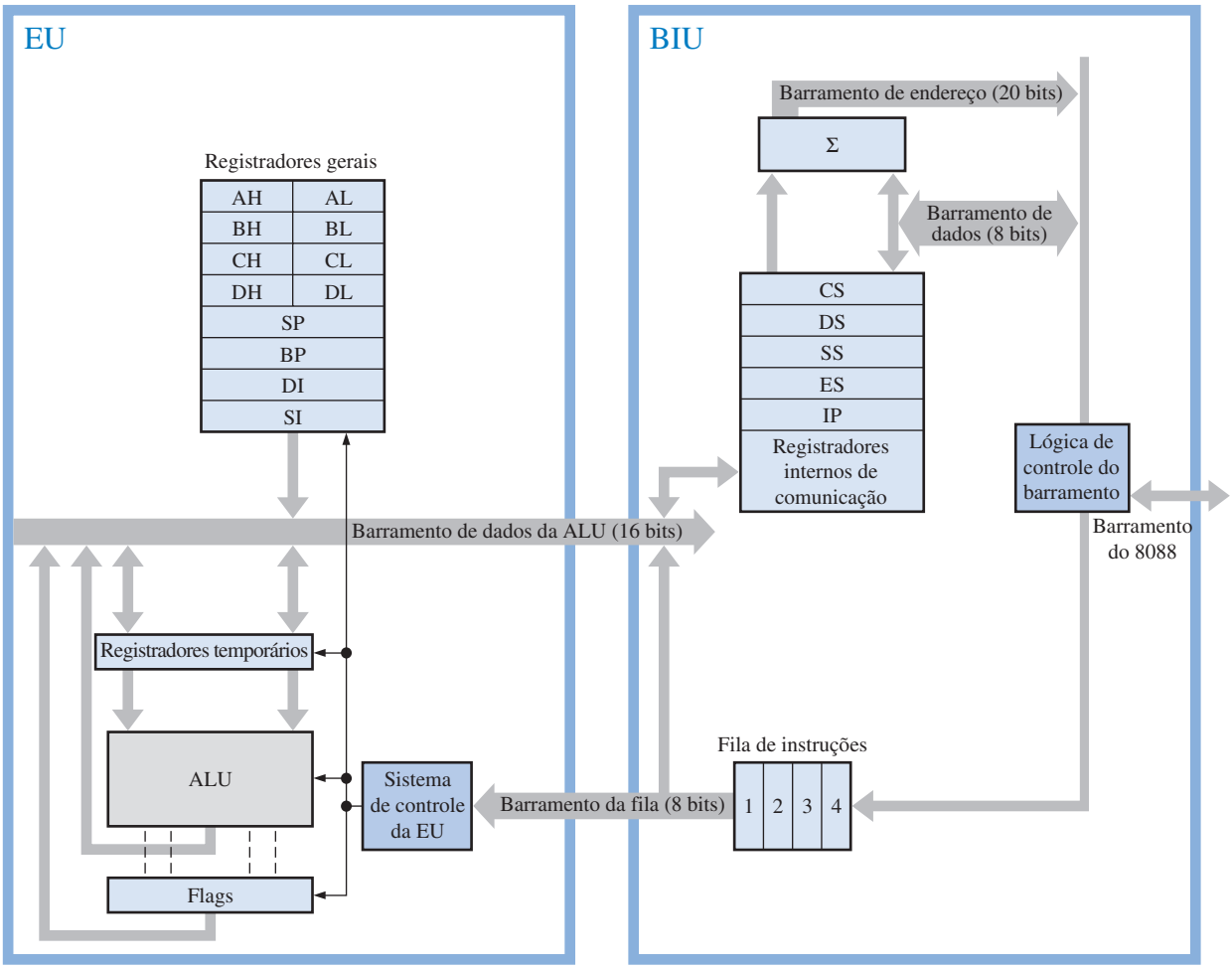
▲ **FIGURA 12–4**

O 8086/8088 tem duas unidades internas separadas, (a EU e a BIU).

A BIU realiza todas as operações de barramentos para a EU, tais como transferência de dados da memória para I/O. Enquanto que a EU está executando instruções, a BIU "olha à frente" buscando mais instruções da memória. Essa ação é denominada prefetching (pré-busca) ou *pipelining*. O conceito de *prefetching* é permitir ao processador executar instruções ao mesmo tempo em que a próxima instrução é buscada, eliminando o tempo de inatividade. As instruções buscadas previamente são armazenadas em uma memória interna de alta velocidade denominada de **fila** de instruções. A fila permite que a BIU mantenha a EU suprida de instruções. A unidade EU não tem que esperar que a próxima instrução seja buscada da memória; mas em vez disso, recupera a próxima instrução diretamente da fila num tempo muito menor. No Pentium, esse processo leva um passo a mais. Duas unidades de execução completas habilitam duas instruções para serem executadas ao mesmo tempo contanto que elas sejam independentes. Certos compiladores são projetados para tirar vantagens de duas unidades de execução através de um processo conhecido como **emparelhamento de instruções** para remover dependências.

Arquitetura Básica do 8086/8088

A Figura 12–5 é um diagrama em bloco da arquitetura (organização interna) de um microprocessador 8088. Externamente, o 8088 tem 20 bits de endereço que podem endereçar 1 MB (1.048.576 bytes) de memória e usa um barramento de dados de 8 bits. Internamente, o 8088 tem um barramento de dados de 16 bits e uma fila de 4 bytes. O 8086 é idêntico exceto que ele tem um barramento de dados externo de 16 bits e uma fila de instruções de 6 bytes.



▲ **FIGURA 12–5**

Organização interna do microprocessador 8088.

Unidade de Interface de Barramento (BIU)

As partes principais da BIU são a fila de instruções de 4 bytes, os registradores de segmento (CS, DS, SS e ES), o ponteiro de instruções (IP) e o bloco de soma de endereços (∑). Os barramentos internos de dados de 16 bits e o barramento da fila (Q) interconectam a BIU e a EU.

Fila de Instruções A fila de instruções aumenta a velocidade média na qual um programa é executado (denominado de **processamento**) armazenando até quatro bytes (seis no 8086). Conforme descrito anteriormente, essa técnica permitiu ao 8088 fazer essencialmente duas coisas: busca e execução, ao mesmo tempo. Essa característica foi expandida em processadores subseqüentes incluindo filas mais rápidas e muito maiores.

Registradores de Segmento Os processadores 8086/8088 têm quatro registradores de segmento (CS, DS, SS e ES) que são registradores de 16 bits usados no processo de formação de um endereço de 20 bits. Um **segmento** é um bloco de 64 kB de memória e pode começar em qualquer ponto de 1 MB (1.048.576 bytes) de espaço de memória, contanto que comece num limite de 16 bytes (uniformemente divisível por 16).

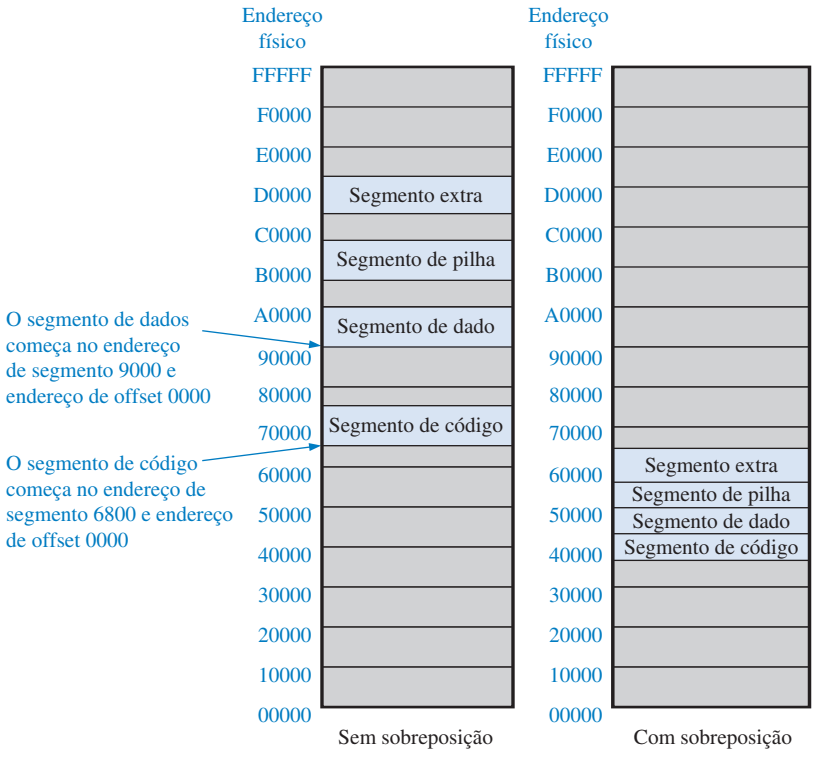
No projeto do 8086/8088 e processadores subseqüentes, a Intel escolheu um único método de geração de endereço físico de 20 bits usando dois registradores de 16 bits. Um dos registradores que formam o endereço físico (ou real) é sempre um registrador de segmento; o outro registrador é um registrador geral de 16 bits que contém a informação de endereço. A principal vantagem do

método selecionado é permitir que o código seja facilmente realocável. Um **código realocável** pode ser movido para qualquer lugar dentro do espaço de memória sem alteração do código básico.

Cada um dos quatro segmentos identifica o endereço inicial de um bloco de 64 kB (65.536 bytes) representando uma "janela" do espaço de memória completo de 1 M (20 bits). O endereço inicial de um segmento é representado pelo número de 16 bits no registrador de segmento mais 4 bits subentendidos anexados à direita os quais são sempre considerados zeros. Em outras palavras, os registradores de segmento contêm os 16 bits mais significativos que representam o endereço físico inicial do segmento.

Os quatro registradores de segmento (CS, DS, SS e ES) podem ser alterados pelo programa para apontar para outro bloco de 64 kB se necessário. (Para códigos menores, normalmente não é necessário alterar os segmentos). Os quatro segmentos podem estar situados em locações separadas dentro do espaço de memória ou podem se sobrepor, dependendo do tamanho e dos requisitos de um código em particular. Eles ainda podem ser definidos com o mesmo bloco de 64 kB. No 8086/8088, os segmentos de memória endereçáveis atualmente são aqueles definidos pelo endereço de segmento contido no registrador CS (segmento de código), no registrador DS (segmento de dado), no registrador SS (segmento de pilha) e no registrador ES (segmento extra). Nos processadores posteriores, foram acrescentados outros registradores de segmento.

Conforme mencionado, dentro de cada segmento são 64 kB de memória. Para encontrar uma posição de memória, um endereço de segmento é combinado com um endereço de *offset*. O endereço de segmento representa os dezesseis bits mais significativos (quatro dígitos hexa) do endereço físico que representa o endereço inicial de um segmento. O endereço de *offset* é dezesseis bits adicionais que representam a "distância" do início do segmento para o endereço físico dentro do segmento. A Figura 12–6 ilustra como a memória é dividida em segmentos e mostra exemplos de segmentos sem sobreposição e com sobreposição.



◀ **FIGURA 12–6**
Segmentos sem sobreposição e com sobreposição no primeiro 1 MB de memória. Cada segmento representa 64 kB.

Ponteiro de Instrução (IP) e Bloco de Soma de Endereço Os 16 bits do **IP** (ponteiro de instrução) aponta para o *offset* da próxima instrução a ser executada que está na memória. O IP tem sempre como referência o registrador CS (segmento de código); assim, o endereço físico da próxima instrução é formado pela combinação do segmento de código e ponteiro de instrução. O IP

contém sempre o endereço de *offset* da próxima instrução e o registrador CS sempre contém o endereço de segmento. Esse endereço é mostrado em linguagem assembly como CS:IP.

Para formar um endereço físico de 20 bits da próxima instrução, o endereço de *offset* de 16 bits em IP é somado ao endereço de segmento contido no registrador CS, o qual foi deslocado quatro bits para a esquerda, conforme indicado na Figura 12–7. Conforme mencionado anteriormente, o número binário 0000 situa-se nas posições menos significativas. A adição é então feita pelo bloco de soma de endereço.

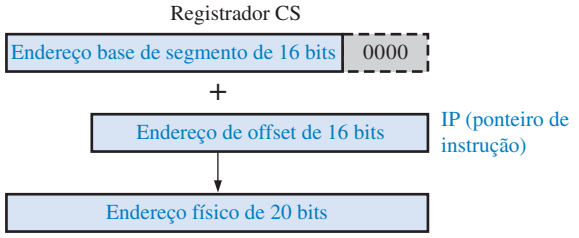


FIGURA 12–7
Formação do endereço físico de 20 bits a partir do endereço base de segmento e do endereço de *offset*.

A Figura 12–8 ilustra o endereçamento de uma posição de memória por segmento: método do *offset*. Nessa figura, $A000_{16}$ está no registrador de segmento e $A0B0_{16}$ é o IP. Quando o registrador CS é deslocado e somado ao IP, obtemos $A0000_{16} + A0B0_{16} = AA0B0_{16}$ para o endereço físico.

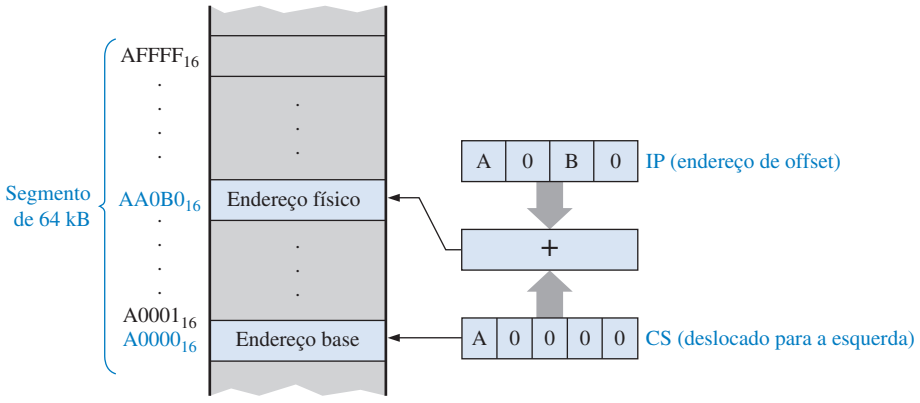


FIGURA 12–8
Ilustração do método de endereçamento segmentado.

EXEMPLO 12–1

O conteúdo hexadecimal do registrador CS e do IP são mostrados na Figura 12–9. Determine o endereço físico da próxima instrução na memória.

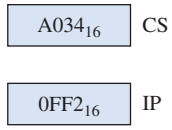
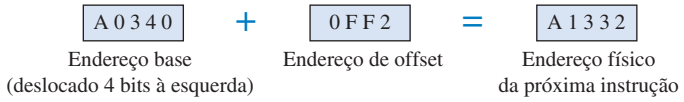


FIGURA 12–9

Solução Deslocar o registrador base CS quatro bits (um dígito hexa) à esquerda coloca efetivamente um 0_{16} na posição LSD, conforme mostra a Figura 12–10. O endereço base deslocado e o endereço de *offset* são somados para produzir um endereço físico de 20 bits.



▶ **FIGURA 12–10**

Problema relacionado* Determine o endereço físico se o registrador CS contiver $6B4D_{16}$.

* As respostas estão no final do capítulo.

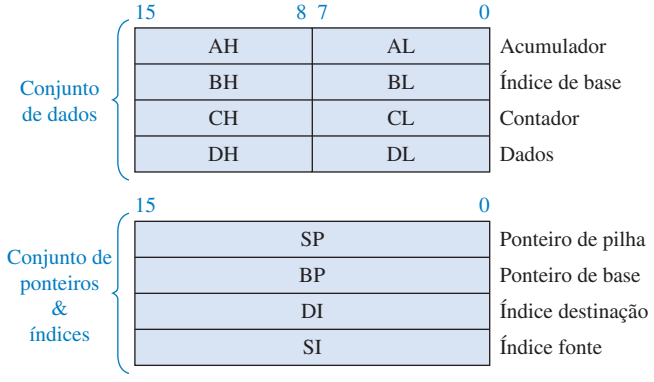
É importante entender como o método de segmento:*offset* é usado para formar o endereço físico; entretanto, em programação normalmente não é necessário para o programador especificar o endereço físico real. Esse trabalho é feito pelo programa assembler usando rótulos fornecidos pelo programador. Quando um endereço físico é necessário, o programador geralmente o especifica com o método de segmento:*offset*. Assim, o endereço do Exemplo 12–1 seria dado como simplesmente A034:0FF2.

Unidade de Execução (EU)

A EU decodifica instruções buscadas pela BIU, gera os sinais de controle apropriados e executa as instruções. As partes principais da EU são a unidade lógica e aritmética (ALU), os registradores gerais e os flags.

ALU Essa unidade faz todas as operações aritméticas e lógicas, trabalhando com operandos de 8 e 16 bits.

Registradores Gerais Esse conjunto de registradores de 16 bits é dividido em dois conjuntos de quatro registradores cada um, conforme mostra a Figura 12–11. Um conjunto consiste de registradores de dados e o outro de registradores de índice e ponteiros. Os registradores de índice e **ponteiro** são geralmente usados para manter endereços de *offset* (conforme usado aqui, um ponteiro se refere a uma posição de memória específica). No caso do ponteiro de pilha (SP) e do ponteiro base (BP), a referência padrão para formar um endereço físico é o segmento de pilha (SS). Os ponteiros de índice (SI e DI) e o registrador base (BX) geralmente padrão para o registrador de segmento de dados (DS) (uma exceção é feita para certas instruções dessa regra geral).



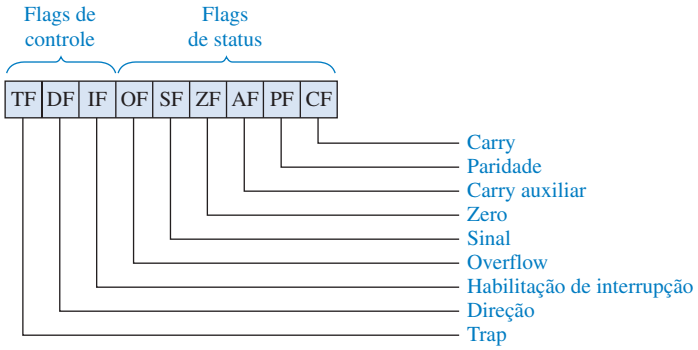
▲ **FIGURA 12–11**

Conjunto de registradores gerais.

Cada um dos registradores de dados de 16 bits (AX, BX, CX, DX) tem duas seções de 8 bits com acesso em separado. Dependendo do programa, eles podem ser usados como um registrador de 16 bits ou como dois registradores de 8 bits. Os bytes de ordem inferior do registrador de dados são designados como *AL*, *BL*, *CL* e *DL*. Os bytes de ordem superior são designados como *AH*, *BH*, *CH* e *DH*. Esses registradores podem ser usados na maioria das operações aritméticas e lógicas de qualquer maneira especificada pelo programador para armazenar dados antes e depois do processamento. Além disso, alguns desses registradores são usados especificamente por determinadas instruções do programa.

Os registradores ponteiros e de índice são o ponteiro de pilha (SP), o ponteiro de base (BP), o índice destinação (DI) e o índice fonte (SI). Esses registradores são usados em várias formas de endereçamento de memória sob o controle da EU.

Flags O registrador de flags contém nove bits independentes de controle e status (**flags**), como mostra a Figura 12–12. Um flag de status é um indicador de um bit usado para refletir uma certa condição após uma operação aritmética ou lógica realizada pela ALU, tal como um *carry* (CF), um resultado zero (ZF) ou o sinal de um resultado (SF), dentre outros. Os flags de controle são usados para alterar operações do processador em certas situações.



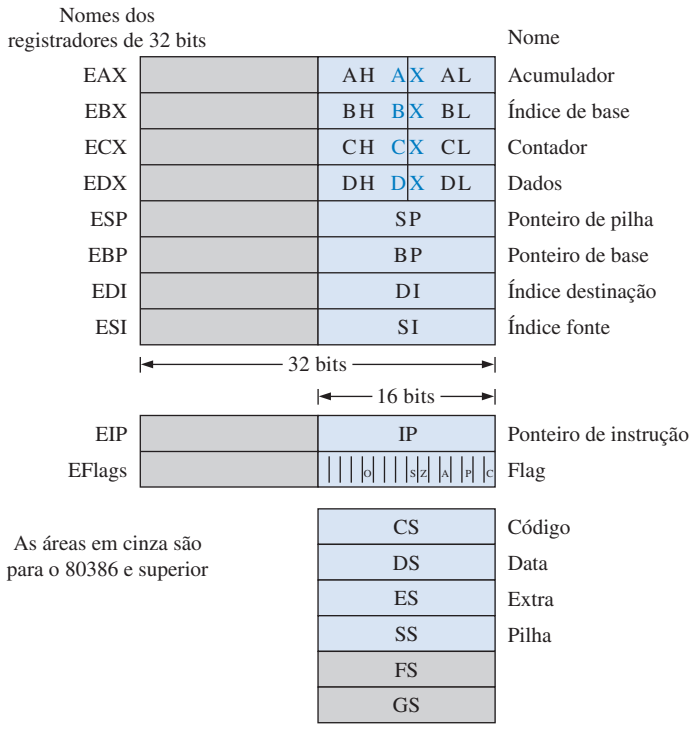
► **FIGURA 12–12**
Flags de controle e status.

Modelo de Software da Família Pentium de Processadores

Com a introdução de novos processadores pela Intel, as capacidades e velocidades aumentaram dramaticamente. Com o processador Pentium, o conceito anterior de *pipeline* introduzido no 8086/8088 foi aumentado para dois inteiros *pipelines*. O co-processador externo foi incorporado dentro do microprocessador e os barramentos de endereço e dados foram bastante expandidos. Outras melhorias (tal como velocidade de clock, ciclos de clock de instrução reduzida, capacidade de predição de desvio e uma unidade de ponto flutuante integral) fazem do Pentium um processador significativamente melhor que os seus predecessores. Além das melhorias no processador, ocorreram muitas melhorias em outras partes do computador (tal como protocolos e tamanhos de barramentos, velocidade, tamanho da memória e custo). Apesar de todas essas mudanças, os projetistas do novo microprocessador mantiveram a compatibilidade com os softwares anteriores; ou seja, o novo Pentium ainda pode executar qualquer um dos softwares dos processadores que o precederam. Isso foi feito mantendo o modelo de software básico (estrutura de registradores) do microprocessador original 8086/8088.

Os registradores descritos anteriormente para o microprocessador 8086/8088 são um subconjunto dos registradores na família Pentium de processadores. Começando com o processador 80386, o conjunto de registradores foi expandido para incluir registradores de 32 bits. Os registradores mantêm os nomes originais mas acrescido da letra E (para Estendido) como um prefixo nos nomes dos registradores; assim, a designação de 32 bits para o registrador AX é EAX. Além disso, dois novos registradores de segmento foram acrescentados. Os registradores estendidos são mostrados na Figura 12–13. As áreas em cinza representam os registradores disponíveis apenas no 386 e versões superiores.

Além dos registradores estendidos, o espaço endereçado na memória foi bastante aumentado com a introdução de novos processadores. Para manter a compatibilidade, a Intel reservou o primeiro 1 MB da memória para a execução de códigos em modo real. O **modo real** é qualquer ope-



◀ **FIGURA 12–13**
Registradores para os processadores Intel do 8086/8088 até o Pentium.

ração que permite ao processador acessar apenas o primeiro 1 MB de memória para simular o 8086/8088. Os códigos escritos para um processador posterior podem executar em modo real no novo processador (embora o inverso não seja estritamente verdadeiro). O código escrito em modo real é geralmente compatível (com algumas exceções) com todos os processadores Intel desde o 8086/8088.

SEÇÃO 12–3 REVISÃO

1. Determine o nome dos registradores de propósito geral no processador Intel.
2. Qual é a finalidade da BIU?
3. A EU faz interface com os barramentos do sistema?
4. Qual é a função da fila de instruções?
5. Qual é a vantagem do método segmento:offset de formação de endereço?
6. O que é emparelhamento de instruções?

12-4 PROGRAMAÇÃO DE UM COMPUTADOR

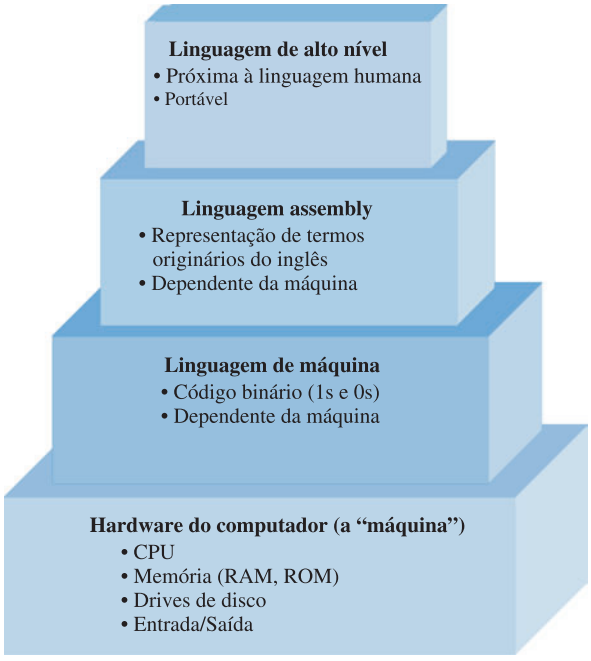
A linguagem assembly é uma forma de expressar uma linguagem de máquina em termos derivados do inglês, assim existe uma correspondência individual. A linguagem assembly tem aplicações limitadas e não é portátil de um processador para outro, de forma que a maioria dos programas de computador é escrita em linguagens de alto nível como C, C++, JAVA, BASIC, COBOL e FORTRAN. As linguagens de alto nível são portáteis, podendo portanto ser usadas em diferentes computadores. As linguagens de alto nível têm que ser convertidas para a linguagem de máquina para um microprocessador específico através de um processo denominado *compilação*.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever alguns conceitos de programação ■ Discutir os níveis de linguagens de programação

Níveis de Linguagens de Programação

Um diagrama hierárquico que relaciona as linguagens de programação de computadores com o hardware é mostrado na Figura 12–14. No nível mais baixo está o hardware do computador (CPU, memória, *drive* de disco, entrada/saída). Em seguida está a **linguagem de máquina** que o hardware entende porque ela é escrita com 1s e 0s (lembre-se, uma porta lógica pode reconhecer apenas nível BAIXO (0) ou nível ALTO (1)). No nível acima da linguagem de máquina está a **linguagem assembly**, na qual os 1s e 0s são representados por palavras originárias do inglês. As linguagens assembly são consideradas de baixo nível porque elas estão bastante próximas da linguagem de máquina e são dependentes da máquina, o que significa que uma determinada linguagem assembly só pode ser usada por um microprocessador específico.



▶ **FIGURA 12–14**
Hierarquia de linguagens de programação em relação ao hardware do computador.

No nível acima da linguagem assembly está a **linguagem de alto nível**, a qual está mais próxima da linguagem humana e mais distante da linguagem de máquina. Uma vantagem da linguagem de alto nível sobre a linguagem assembly é que a primeira é portátil, o que significa que um programa pode ser executado em uma variedade de computadores. Além disso, a linguagem de alto nível é mais fácil de ser lida, escrita e mantida que a linguagem assembly.

Linguagem Assembly

Para evitar a escrita de uma longa seqüência de 1s e 0s para representar as instruções de um microprocessador, são usados termos originários do inglês denominados de mnemônicos ou **códigos de operação**. Cada tipo de microprocessador tem o seu próprio conjunto de instruções mnemônicas que representam os códigos binários das instruções. Todas as instruções mnemônicas para um determinado processador são denominadas de conjunto de instruções. A linguagem assembly usa o conjunto de instruções para criar programas para o microprocessador; como a linguagem assembly é diretamente relacionada à linguagem de máquina (instruções em código binário), ela é classificada com linguagem de baixo nível. A linguagem assembly está a um passo da linguagem de máquina.

A linguagem assembly e a correspondente linguagem de máquina que ela representa é específica para o tipo de microprocessador ou família de microprocessadores. A linguagem assembly não é portátil; ou seja, não podemos executar um programa em linguagem assembly escrito para um tipo de microprocessador em um outro tipo de microprocessador. Por exemplo, um programa em as-

sembly para os processadores Motorola não funciona em processadores Intel. Mesmo dentro uma determinada família, microprocessadores diferentes podem ter conjuntos de instruções diferentes.

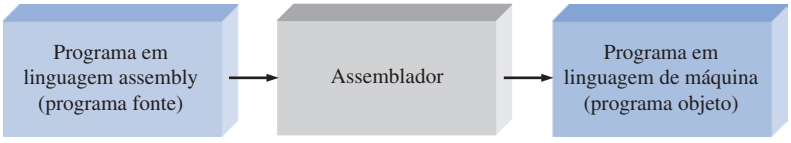
Um **assemblador** (*assembler*) é um programa que converte um programa em linguagem assembly para a linguagem de máquina que é reconhecida pelo microprocessador. Além disso, os programas denominados de ***cross-assemblers*** (assembladores cruzados) traduzem um programa em linguagem assembly de um tipo de microprocessador para uma linguagem assembly de um outro tipo de microprocessador.

A linguagem assembly é raramente usada para criar extensos programas de aplicações. Entretanto, a linguagem assembly é freqüentemente usada em uma sub-rotina (um pequeno programa dentro de um programa maior) que pode ser acessada (ou chamada) a partir de um programa em linguagem de alto nível. A linguagem assembly é útil em aplicações de sub-rotina porque geralmente é executada de forma mais rápida e não tem as restrições de uma linguagem de alto nível. A linguagem assembly também é usada no controle da máquina, tal como em processos industriais. Uma outra área de uso da linguagem assembly é na programação de jogos eletrônicos (videogames).

Conversão de um Programa para a Linguagem de Máquina

Todos os programas escritos em linguagem assembly ou linguagem de alto nível, têm que ser convertidos em linguagem de máquina para que um computador em particular reconheça as instruções do programa.

Assembladores Um assemblador (*assembler*) traduz e converte um programa escrito em linguagem assembly para código de máquina, conforme indicado na Figura 12–15. O termo **programa fonte** é freqüentemente usado para se referir ao programa escrito na linguagem assembly ou de alto nível. O termo **programa objeto** se refere à linguagem de máquina que é tradução de um programa fonte.



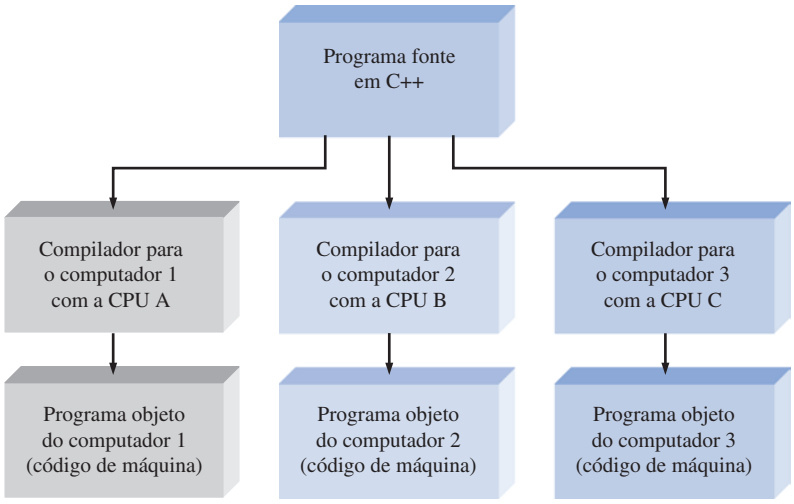
◀ **FIGURA 12–15**
Conversão da linguagem assembly para a de máquina usando um assemblador.

Compiladores Um *compilador* é um programa que compila ou traduz um programa escrito em uma linguagem de alto nível e o converte em código de máquina, como mostra a Figura 12–16. O compilador examina todo o programa fonte e reúne e organiza as instruções. Cada linguagem de alto nível vem com um compilador específico para um computador específico, fazendo a linguagem de alto nível independente do computador no qual é usada. Algumas linguagens de alto nível são traduzidas usando o que chamamos de *interpretador* que traduz cada linha do código do programa em linguagem de máquina.

◀ **FIGURA 12–16**
Conversão de linguagem de alto nível em linguagem de máquina usando um compilador.

Todas as linguagens de alto nível, tais como C, C++, FORTRAN e COBOL, podem ser executadas em qualquer computador. Uma determinada linguagem de alto nível é válida para qualquer computador, porém o compilador que a acompanha é específico para um tipo de CPU em particular. Isso está ilustrado na Figura 12–17, onde o mesmo programa em linguagem de alto nível (escrito em C++ nesse caso) é convertido por diferentes compiladores específicos.

Exemplo de um Programa em Linguagem Assembly Para um simples programa em linguagem assembly, digamos que queremos que o computador some uma lista de números a partir da memória e coloque a soma desses números de volta na memória. O número zero é usado como o



► FIGURA 12–17

Independência da linguagem de máquina em relação a um programa escrito em linguagem de alto nível.

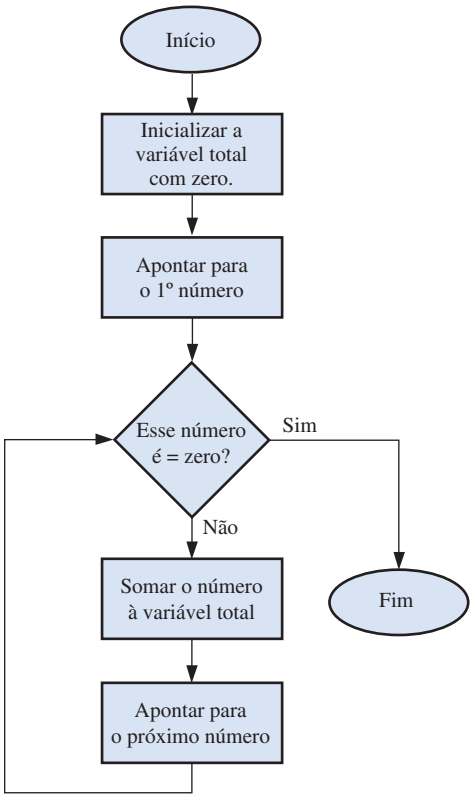
último número na lista para indicar o final da lista. Os passos necessários para realizar essa tarefa são os seguintes:

1. Zerar (limpar) um registro (no microprocessador) para armazenar o total ou soma dos números.
2. Apontar para o primeiro número na memória RAM.
3. Verificar se o número é zero. Em caso afirmativo, todos os números têm que ser somados.
4. Se o número não for zero, somar o número apontado na memória com o registrador que armazena o total.
5. Apontar para o próximo número na memória.
6. Repetir os passos 3, 4 e 5.

Freqüentemente é usado um fluxograma para representar a seqüência de passos de um programa de computador. A Figura 12–18 mostra um fluxograma para o programa representado pelos 6 passos listados acima.

O programa em linguagem assembly implementa o problema da adição mostrado no fluxograma da Figura 12–18. Dois dos registradores do microprocessador usado são denominados de ax e bx. Os comentários após o ponto-e-vírgula não são reconhecidos pelo microprocessador; eles são usados apenas para explicação.

```
      mov ax,0              ;Substitui o conteúdo do registrador ax por zero.
                            ;O registrador ax armazernará o total da adição.
      mov bx,50H            ;Carrega o endereço hexadecimal da memória (50H) no registrador bx.
next: cmp word ptr [bx],0   ;Compara o número armazenado na posição de memória apontada pelo
                            ;registrador bx com zero.
      jz done               ;Se o número nessa posição de memória for zero, salta (jump) para
                            ;"done" (problema solucionado).
      add ax,[bx]           ;Soma o número na posição de memória apontada pelo registrador bx com
                            ;o número no registrador ax e armazena o resultado no registrador ax.
      add bx,02             ;Soma 2 ao endereço no registrador bx. Dois endereços são necessários
                            ;para armazenar cada número que tem extensão de dois bytes.
      jmp next              ;Retorna para o início do loop em "next" e repete o processo.
done: mov [bx],ax           ;Substitui o ultimo número, que é zero, na memória apontada pelo
                            ;registrador bx pelo total da soma que está no registrador ax.
      nop                   ;Não realiza operação, indicando o fim desse programa.
```



◄ **FIGURA 12–18**
Fluxograma para somar uma lista de números.

Dependendo do assemblador, a maioria dos programas em linguagem assembly tem diversas diretivas que são usadas pelo assemblador para realizar uma variedade de tarefas. Essas tarefas incluem definição de segmento, usando o conjunto de instruções apropriado, descrição de tamanhos de dados e a realização de muitas outras funções internas do assemblador. Para simplificar a explicação, apenas uma diretiva (necessária) foi mostrada no programa anterior. A diretiva foi **word ptr**, que é usada para indicar a extensão do dado apontado pelo registrador BX.

Debug Assembler

Com algumas pequenas alterações, podemos executar o programa anterior usando, caso optemos, um assemblador que acompanha os PCs baseados no DOS. Podemos observá-lo e executá-lo passo a passo. Todos os PCs baseados no DOS têm um programa denominado *Debug* (depurador) que inclui um assemblador primitivo. Para usar o *Debug*, digite no *prompt* do DOS **Debug<cr>**. (**<cr>** significa "*carriage return*" (retorno do carro), que é o mesmo que pressionar a tecla ENTER). Veremos um sinal negativo, que é o *prompt* do *Debug*. O *Debug* tem diversos comandos para observar ou inserir dados ou programas. A lista completa dos comandos do *Debug* é mostrada quando digitamos **?** no *prompt* do *Debug*. Antes de escrever e executar um programa em assembly, podemos inserir algum dado digitando a informação mostrada na cor laranja:

```
-a 50
```

Isso diz ao *Debug* para iniciar a montagem das instruções no segmento de dados atual com um *offset* de 50H. Embora esses sejam dados inseridos, é mais simples inserir uma palavra de 16 bits dessa forma. (Tenha em mente que todos os dados no *Debug* são inseridos em HEXA). O *Debug* responde com um endereço de segmento, o qual será indubitavelmente diferente daquele mostra-

do (20D8), mas não importa. O endereço de *offset* (50H) será o mesmo. Digite a informação mostrada na cor laranja (lembre-se de que cada <cr> significa o acionamento da tecla ENTER).

```
20D8:0050 dw 30 <cr>
20D8:0052 dw 15 <cr>
20D8:0054 dw a0 <cr>
20D8:0056 dw 0c <cr>
20D8:0058 dw 00 <cr>
<cr>
```

O comando **dw** é uma diretiva do assemblador. Essa diretiva não armazena por si mesma; ela simplesmente informa ao assemblador que cada dado tem uma extensão de dois bytes. No dado mostrado, apenas um byte é usado, porém o programa salva cada dado em duas posições (endereços) com zeros na posição mais significativa.

Agora podemos inserir o programa na posição 100 como a seguir:

```
          -a 100 <cr>
20D8:0100 mov ax,0 <cr>
20D8:0103 mov bx,50 <cr>
20D8:0106 cmp word ptr [bx],0 <cr>
20D8:0109 jz 112 <cr>
20D8:010B add ax,[bx] <cr>
20D8:010D add bx,2 <cr>
20D8:0110 jmp 106 <cr>
20D8:0112 mov [bx],ax <cr>
20D8:0114 nop <cr>
20D8:0115 <cr>
```

Para confirmar que o programa foi inserido corretamente, temos que digitar **u 100 114** no *prompt* do *Debug*, sendo que o código digitado será mostrado na tela. (Ele será mostrado em letras maiúsculas). Agora digitamos **r** após o *prompt* do *Debug*, sendo que aparecerá uma lista de registradores de 16 bits e a condição dos flags. Observe que o IP deve conter 100, o endereço inicial do código. Abaixo da lista de registradores, a primeira instrução (MOV AX, 0000) será mostrada.

Podemos fazer com que o *Debug* execute essa instrução com o comando **t** (*trace*). Isso colocará na tela a última condição de todos os registradores e mostrará a próxima instrução (MOV BX, 0050). A execução dessa com o comando **t** mostrará que o número 0050 foi movido para o registrador BX. Os passos até esse momento são mostrados na Figura 12–19. Observe que o primeiro dado é mostrado em baixo à direita.

Continuando dessa forma, podemos executar o código completo e observar as mudanças nos registradores conforme o microprocessador executa as instruções, como mostra a Figura 12–20. Esse programa tem uma estrutura de programação comum denominada *loop*. Um *loop* é um grupo de instruções executadas de forma repetitiva até que alguma condição seja encontrada; nesse caso, a condição é encontrar um dado zero. Após isso, a última instrução será executada e a soma armazenada (nesse caso, 00F1 é a soma em hexa) no lugar do zero que indicava o último dado. Podemos observar isso pressionando **d 0050 005F** (apresenta os dados entre os endereços 0050 e 005F) no *prompt* do Bebug quando a última instrução (NOP) é executada, como mostra a Figura 12–20. O resultado parece como o nono e o décimo bytes (a 5ª palavra) na linha que segue a instrução de apresentação. Observe que a parte menos significativa da resposta é mostrada primeiro. Quando executado em "tempo real" pelo microprocessador, esse programa gasta apenas cerca de 1μs para fazer todo esse processo. Se optarmos por repetir o processo, precisamos recarregar o zero na posição 0058 porque ele foi substituído pelo resultado da soma. (Lembre que o programa usa o zero como indicador de "último dado").

```
-u 100 114
20D8:0100 B80000          MOV     AX,0000
20D8:0103 BB5000          MOV     BX,0050
20D8:0106 833F00          CMP     WORD PTR [BX],+00
20D8:0109 7407            JZ      0112
20D8:010B 0307            ADD     AX,[BX]
20D8:010D 83C302          ADD     BX,+02
20D8:0110 EBF4            JMP     0106
20D8:0112 8907            MOV     [BX],AX
20D8:0114 90              NOP
-r
AX=0000   BX=0000   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0100     NV UP EI PL ZR NA PE NC
20D8:0100 B8000           MOV     AX,0000
-t

AX=0000   BX=0000   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0103     NV UP EI PL ZR NA PE NC
20D8:0103 BB5000          MOV     BX,0050
-t

AX=0000   BX=0050   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0106     NV UP EI PL ZR NA PE NC
20D8:0106 833F00          CMP     WORD PTR [BX],+00                    DS:0050=0030
-
```

▲ **FIGURA 12–19**

Passos do início da execução do programa de adição com *Debug*.

```
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0110     NV UP EI PL NZ NA PO NC
20D8:0110 EBF4            JMP     0106
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0106     NV UP EI PL NZ NA PO NC
20D8:0106 833F00          CMP     WORD PTR [BX],+00                    DS:0058=0000
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0109     NV UP EI PL ZR NA PE NC
20D8:0109 7407            JZ      0112
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0112     NV UP EI PL ZR NA PE NC
20D8:0112 8907            MOV     [BX],AX                              DS:0058=0000
-t

AX=00F1   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0114     NV UP EI PL ZR NA PE NC
20D8:0114 90              NOP
-d 0050 005f
20D8:0050   30 00 15 00 A0 00 0C 00-F1 00 00 00 00 20 20 20    0............
-
                  Dados originais          Soma
```

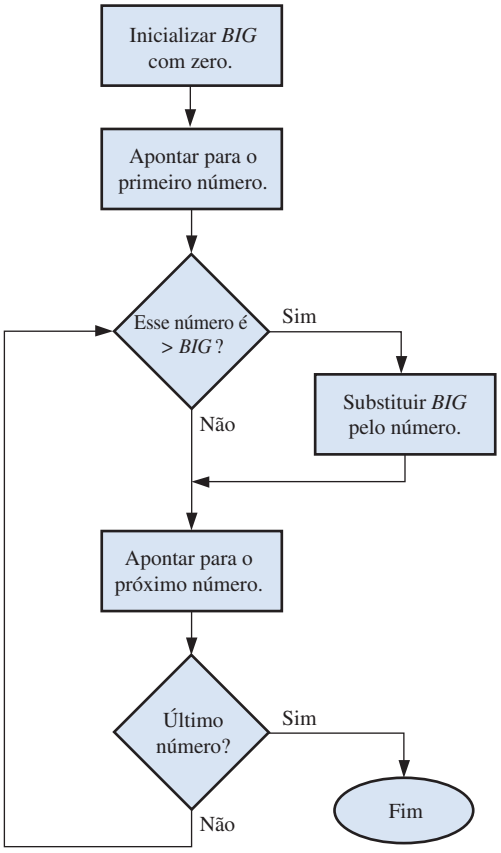
▲ **FIGURA 12–20**

A última parte do programa de adição. A soma 00F1 é mostrada na cor laranja com a parte menos significativa (F1) mostrada primeiro.

EXEMPLO 12–2

Escreva as instruções para um programa em linguagem *assembly* que identifique o maior número não sinalizado num grupo de dados e o coloque na última posição. Considere que o último dado seja indicado por um zero.

Solução O fluxograma é mostrado na Figura 12–21.



▶ **FIGURA 12–21**
Fluxograma. A variável *BIG* representa o maior valor.

Considere que os dados sejam os mesmos de antes. A listagem do programa (com os comentários) é a seguinte:

```
        mov ax,0000          ; o valor inicial de BIG está no registrador ax
        mov bx,0050          ; aponta para uma posição na memória (50H) onde
                             ; começam os dados
repeat: cmp [bx],ax          ; o dado apontado é maior que BIG?
        jbe check            ; se o dado for menor, pule para "check"
        mov ax,[bx]          ; caso contrário, coloque em ax um novo dado apontado
 check: add bx,02            ; aponta para o próximo número na memória (dois bytes
                             ; por palavra)
        cmp word ptr [bx],0  ; verifica se é o último dado
        jnz repeat           ; continua se o dado apontado não for zero
        mov [bx],ax          ; salva BIG na memória
        nop                  ; sem operação
```

A listagem do *Debug* desse programa é mostrada na Figura 12–22. Os dados são inseridos da mesma forma como mostrado antes começando pela posição 0050. Nesse caso os mesmos dados são usados, mas podemos escolher novos dados se preferirmos. É

importante que um zero seja inserido como último dado porque o programa continua até que ele seja identificado nessa posição. O zero é substituído pelo maior dado a cada vez que o programa é executado. O programa é inserido começando o *assembly* na posição 100 inserindo o programa após o comando **a 100** ser emitido.

```
-a 100
20D8:0100 mov ax,0
20D8:0103 mov bx,50
20D8:0106 cmp [bx],ax
20D8:0108 jbe 10c
20D8:010A mov ax, [bx]
20D8:010C add bx,2
20D8:010F cmp word ptr [bx],0
20D8:0112 jnz 106
20D8:0114 mov [bx],ax
20D8:0116 nop
20D8:0117
```

▶ **FIGURA 12–22**

Listagem de parte do *Debug* do programa.

O programa pode ser investigado, um passo de cada vez, para ver a execução. Alternativamente, podemos inserir **g = 100 116** para executar (*go*) as instruções entre 100 e 116. A Figura 12–23 mostra os dados antes e depois da execução. Observe que cada dado apontado é armazenado em dois bytes (embora os dados tenham extensão de apenas um byte) porque os dados foram definidos como *words*. Além disso, note que o byte menos significativo precede o mais significativo na memória. Após o programa ser executado, o último dado (que antes era zero) é visto como sendo igual ao maior valor (A0 nesse exemplo) Esse valor é visto tanto no registrador AX quanto na memória.

```
-a 100
20D8:0100 mov ax,0
20D8:0103 mov bx,50
20D8:0106 cmp [bx],ax
20D8:0108 jbe 10c
20D8:010A mov ax, [bx]
20D8:010C add bx,2
20D8:010F cmp word ptr [bx],0
20D8:0112 jnz 106
20D8:0114 mov [bx],ax
20D8:0116 nop
20D8:0117                          Indicador de fim de dados
-d 50 5f
20D8:0050   30 00 15 00 A0 00 0C 00-00 00  00 00 00 20 20 20   0.............
-g= 100 116
          Dados anteriores à execução do programa

AX=00A0   BX=0058   CX=0000   DX=0000   SP=FFEE   BP=0000   SI=0000   DI=0000
DS=20D8   ES=20D8   SS=20D8   CS=20D8   IP=0116     NV UP EI PL ZR NA PE NC
20D8:0116 90              NOP
-d 50 5f
20D8:0050   30 00 15 00 A0 00 0C 00-A0 00  00 00 00 20 20 20   0.............
-                 Os dados não são alterados
-                                   O fim de dados agora
-                                   tem o maior dado
```

▲ **FIGURA 12–23**

Dados anteriores e posteriores à execução do programa.

Problema relacionado Explique como você alteraria o fluxograma para identificar o menor número da lista em vez do maior.

Tipos de Instruções

Os programas nessa seção mostram apenas algumas das centenas de variações de instruções disponíveis aos programadores. Para simplificar o estudo do conjunto de instruções Intel, elas são divididas em sete categorias. Essas categorias são descritas aqui.

Transferência de Dados A instrução mais básica de transferência de dados, MOV, foi introduzida nos programas exemplo. A instrução MOV, por exemplo, pode ser usada de diversas formas para copiar um byte, uma word (16 bits) ou uma doble word (32 bits) entre várias fontes e destinos tais como registradores, memórias e portas de I/O. (Um mnemônico melhor que MOV seria "COPY" porque é isso que realmente a instrução faz, uma cópia.) Outras instruções de transferência de dados incluem IN (para obter dados de uma porta de entrada), OUT (para enviar dados para uma porta de saída), PUSH (para copiar dados para a pilha, que é uma área de memória separada), POP (para copiar dados a partir da pilha) e XCHG (para efetuar troca).

Aritmética Existem diversas instruções e variações dessas instruções para adição, subtração, multiplicação e divisão. A instrução ADD foi usada nos dois programas exemplo. Outras instruções aritméticas incluindo INC (incremento), DEC (decremento), CMP (comparação), SUB (subtração), MUL (multiplicação) e DIV (divisão). Variações dessas instruções permitem operações com *carry* para aritmética com números sinalizados e não-sinalizados. Essas instruções possibilitam operações com operandos localizados na memória, em registradores, e em portas de I/O.

Manipulação de Bit Esse grupo de instruções incluem aquelas usadas para as três classes de operação: operações lógicas (Booleanas), deslocamento e rotações. As instruções lógicas são NOT, AND, XOR e TEST. Um exemplo de uma instrução de deslocamento é a SAR (deslocamento aritmético à direita). Um exemplo de uma instrução de rotação é a ROL (rotação à esquerda). Quando os bits são deslocados para fora de um operando, eles são perdidos; mas quando os bits são rotacionados (*loop*), eles retornam pela outra extremidade. Essas instruções lógicas, de deslocamento e de rotação podem operar nos bytes ou words nos registradores ou na memória.

Loops e Jumps Essas instruções são projetadas para alterar a seqüência normal (uma após a outra) de instruções. A maioria dessas instruções testa os flags do processador para determinar qual instrução deve ser processada em seguida. No Exemplo 12–2, as instruções JBE e JNZ foram usadas para alterar o percurso da execução. Outras instruções desse grupo incluem JMP (salto incondicional), JA (saltar se acima), JO (salto em caso de *overflow*), LOOP (decrementa o registrador CX e repete se não for zero) e muitas outras.

Strings Uma ***string*** é uma seqüência **contígua** (um após o outro) de bytes ou words. *Strings* são comuns em programas de computadores. Um exemplo simples é uma frase que o programador deseja mostrar na tela. Existem cinco instruções básicas de *strings* que são projetadas para copiar, carregar, armazenar, comparar ou escanear uma *string* – como um byte de cada vez ou uma word de cada vez. Como exemplos de instruções de *string* temos MOVSB (copia uma *string*, sendo um byte de cada vez) e MOVSW (copia uma *string*, sendo uma palavra de cada vez).

Sub-rotina e Interrupções Uma **sub-rotina** é um miniprograma que pode ser usado repetidamente porém programado apenas uma vez. Por exemplo, se um programador necessita converter números ASCII a partir de um teclado para o formato BCD, uma estrutura de programação simples faz uso das instruções necessárias como um processo separado do programa principal e "chama (*call*)" o processo sempre que necessário. As instruções desse grupo incluem CALL (inicia a sub-rotina) e RET (retorna ao programa principal).

Controle do Processador Esse é um pequeno grupo de instruções que permite o controle direto de alguns flags do processador e outras tarefas diversas. Um exemplo é a instrução STC (*setar* o flag de *carry*).

Programação de Alto Nível

Os passos básicos a serem seguidos quando escrevemos um programa de computador em linguagem de alto nível, independentemente da linguagem de programação em particular que usamos, são os seguintes:

1. Determine e especifique o problema a ser solucionado ou a tarefa a ser feita.
2. Gere um algoritmo; ou seja, descreva uma série de passos para realizar a tarefa.
3. Expresse os passos usando uma linguagem de programação em particular inserindo-a por meio do editor de texto do software.
4. Compile (ou assemble) e execute o programa.

Um simples programa, a seguir, mostra um exemplo de programação de alto nível. O programa em C++ a seguir implementa a mesma soma do problema definido pelo fluxograma dado na Figura 12–18 e implementado usando linguagem *assembly*.

```
int total = 0;              //Inicializa a variável total com 0.
while (*number ! = 0×00)    //Fica em loop enquanto o valor não for
                            //encontrado. O asterisco que precede o
                            //número identificador de ponteiro diz
                            //que o conteúdo da posição de memória
                            //apontada pelo número identificador
                            //está sendo avaliado.
{
total = total + *number;    //Soma acumulativa da variável total.
number++;                   //Incrementa o ponteiro para apontar o
                            //próximo número na memória.
}
```

Esse programa C++ é equivalente ao programa em *assembly* que soma uma série de números e produz um valor total.

```
in total = 0;                                   mov ax, 0
while (*number ! = 0×00)                        mov bx, 50H
{                                         next: cmp word ptr [bx], 0
       total = total + *number;                 jz done
       number++;                                add ax, [bx]
}                                               add bx, 02
                                                jmp next
       C++                                done: mov [bx], ax
                                                nop

                                                Assembly
```

Equivalente

SEÇÃO 12–4 REVISÃO

1. Defina programa.
2. O que é um código de operação?
3. O que é uma *string*?

12-5 INTERRUPÇÕES

Nesta seção, o estabelecimento de comunicações entre um periférico e a CPU é apresentado. Três métodos são discutidos: consulta de I/O, interrupção ativada por I/O e interrupções por software.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir a necessidade de interrupções em um sistema de computador ■ Descrever o conceito básico de uma consulta em I/O ■ Descrever o conceito básico de uma interrupção ativada por I/O ■ Discutir uma interrupção por software

Em sistemas microprocessados tal como o computador pessoal, os dispositivos periféricos necessitam de um serviço periódico por parte da CPU. O termo serviço geralmente significa enviar ou receber dados de dispositivos ou realizar algum processo de atualização. Existem três formas em que uma rotina de serviço pode ser iniciada: por *consulta em I/O*, por *interrupção ativada por I/O* ou *interrupções por software*. Lembre-se que uma interrupção é um sinal ou uma instrução que faz com que o processamento atual seja temporariamente paralisado enquanto uma rotina de serviço é executada.

Em geral, os dispositivos periféricos são bem mais lentos em comparação com a CPU. Uma impressora pode em média imprimir apenas alguns caracteres por segundo (um caractere é representado por oito bits), dependendo do tipo de material a ser impresso e do tipo da impressora. A taxa de entrada de caracteres via teclado pode ser de um ou dois caracteres por segundo, dependendo da velocidade do digitador. Assim, entre os momentos em que a CPU é solicitada para servir um periférico, ela pode fazer muito em termos de processamento. Na maioria dos sistemas, esse tempo de processamento tem que ser maximizado usando um método eficiente no serviço de periféricos.

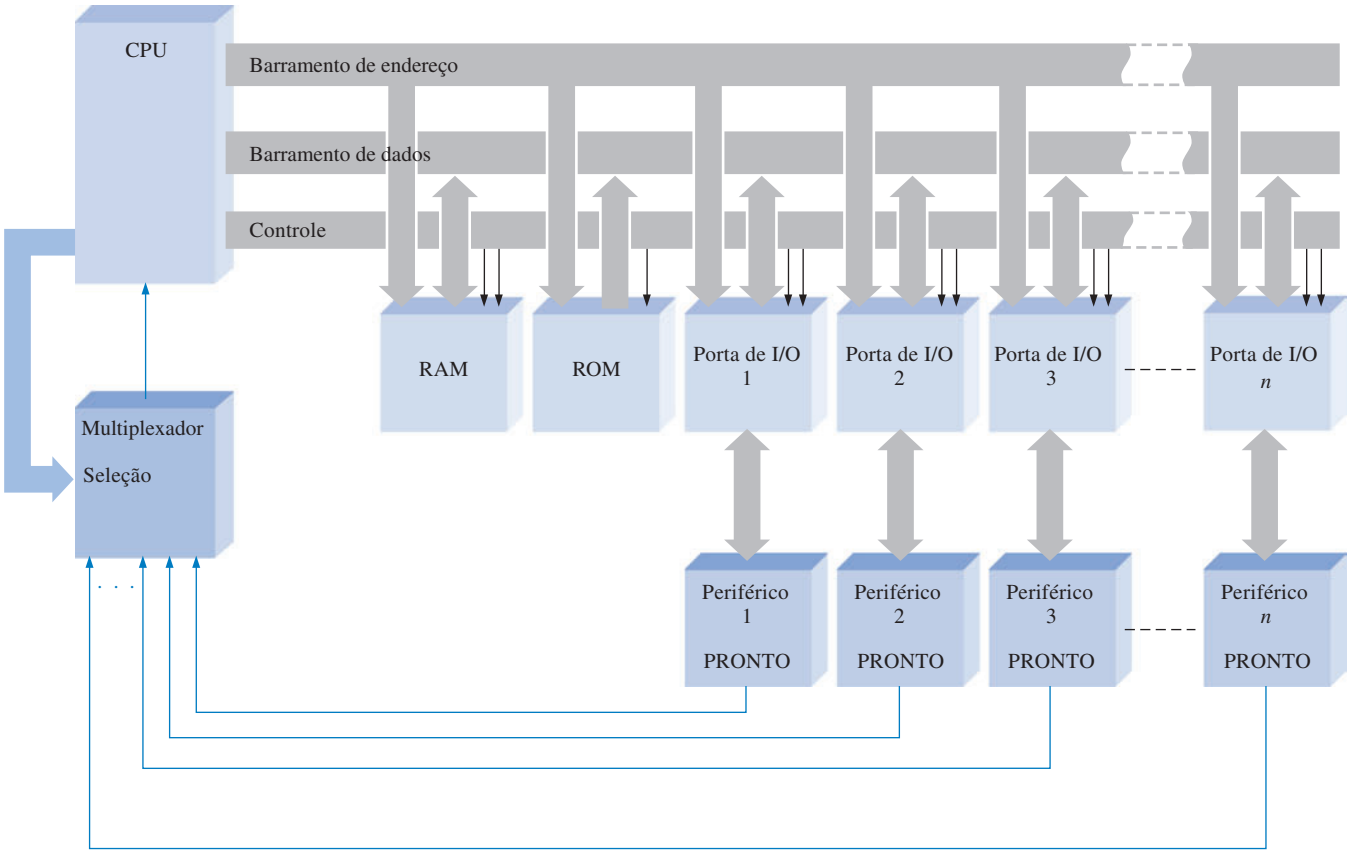
Consulta em I/O

Um método de serviço de periféricos é denominado de **consulta** (*polling*). Nesse método, a CPU tem que testar cada dispositivo periférico numa seqüência em determinados intervalos para verificar se eles precisam ou estão prontos para algum serviço. A Figura 12–24 ilustra o método básico de consulta em I/O.

A CPU seleciona sequencialmente cada dispositivo periférico via multiplexador para "ver" se ele precisa de serviço verificando o estado de sua linha pronto (*ready*). Certamente os periféricos podem precisar de serviço em intervalos irregulares e imprevisíveis, ou seja, mais frequentemente em algumas ocasiões do que em outras. Não obstante, a CPU tem que consultar o dispositivo em uma taxa maior. Por exemplo, digamos que um certo periférico necessita ocasionalmente de serviço a cada 1000μs mas a maior parte do tempo necessita de serviço apenas uma vez a cada 100 ms. Como podemos ver, um tempo precioso de processamento é perdido se a CPU consultar o dispositivo, conforme ele precisa, em sua taxa máxima (a cada 1000μs) porque na maior parte do tempo o dispositivo não precisará de serviço quando for consultado.

Cada vez que a CPU consulta um dispositivo, ela tem que paralisar o programa que está processando, ir até a seqüência de consulta, provê o serviço necessário para então retornar para o ponto onde deixou o processamento.

Um outro problema com a abordagem de consulta seqüencial de I/O é que se dois ou mais dispositivos necessitam de serviço ao mesmo tempo, o primeiro a ser consultado será servido primeiro; os outros dispositivos terão que esperar embora possam estar precisando do serviço com mais urgência que o primeiro dispositivo consultado. Como podemos ver, a consulta é satisfatória apenas para dispositivos que podem ser servidos em intervalos regulares e previsíveis apenas em situações nas quais não existam considerações de prioridade.



▲ **FIGURA 12–24**

Configuração básica de dispositivos I/O consultados.

Interrupção Ativada por I/O

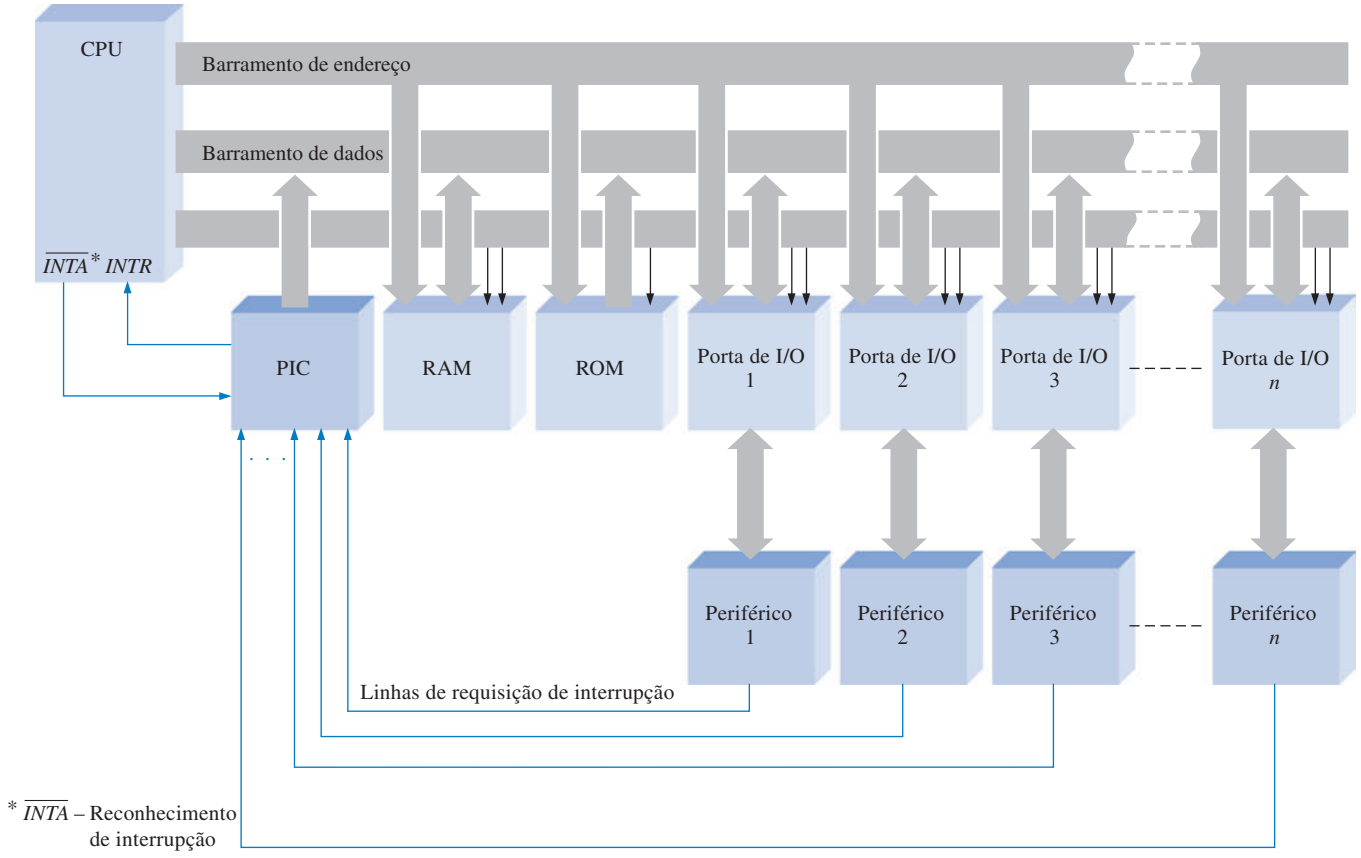
Essa abordagem supera as desvantagens do método de consulta. No método de interrupção ativada por I/O, a CPU responde à necessidade de serviço apenas quando o serviço é solicitado por um dispositivo periférico. Portanto, a CPU pode se concentrar na execução do programa atual sem ter que parar desnecessariamente para verificar se um dispositivo precisa de serviço.

Quando a CPU recebe um sinal de interrupção de I/O, ela paralisa temporariamente o programa atual, reconhece a interrupção e busca um programa especial (rotina de serviço) na memória para o dispositivo em particular que gerou a interrupção. Quando a rotina de serviço é finalizada, a CPU retorna ao ponto do programa que executava antes da interrupção.

Um dispositivo denominado controlador de interrupção programável (**PIC** – *programmable interrupt controller*) opera interrupções baseadas em prioridades. Ele aceita as requisições de serviço a partir dos periféricos. Se dois ou mais dispositivos solicitam serviço ao mesmo tempo, aquele que estiver associado à prioridade mais alta será servido primeiro, em seguida aquele com prioridade logo abaixo da mais alta, e assim por diante. Após emitir um sinal de interrupção (*INTR*) para a CPU, o PIC provê a CPU com informações que indicam (apontam) à CPU o endereço inicial na memória da rotina de serviço apropriada. Esse processo é denominado de *vetoração*. A Figura 12–25 mostra uma configuração básica da interrupção ativada por I/O.

Interrupções por Software

Um outro tipo de interrupção é denominada **interrupção por software**. As interrupções por software são instruções de programa que podem invocar as mesmas rotinas de serviço descritas anteriormente. A diferença é que essas interrupções podem ser invocadas por software em vez do hardware externo. Quando invocada, a rotina de serviço da interrupção executa exatamente como se



▲ **FIGURA 12–25**

Configuração básica de interrupção ativada por I/O.

uma interrupção de hardware tivesse ocorrido. As primeiras cinco interrupções são definidas pela Intel. As outras interrupções são definidas pelos BIOS e pelo DOS para realizar muitas das operações de I/O, tais como leitura e escrita de dados no disco, escrita de dados no display (monitor) e leitura de dados do teclado.

SEÇÃO 12–5 REVISÃO

1. Em que uma interrupção ativada por I/O difere de uma consulta em I/O?
2. Qual é a principal vantagem de uma interrupção ativada por I/O?
3. O que é uma interrupção por software?

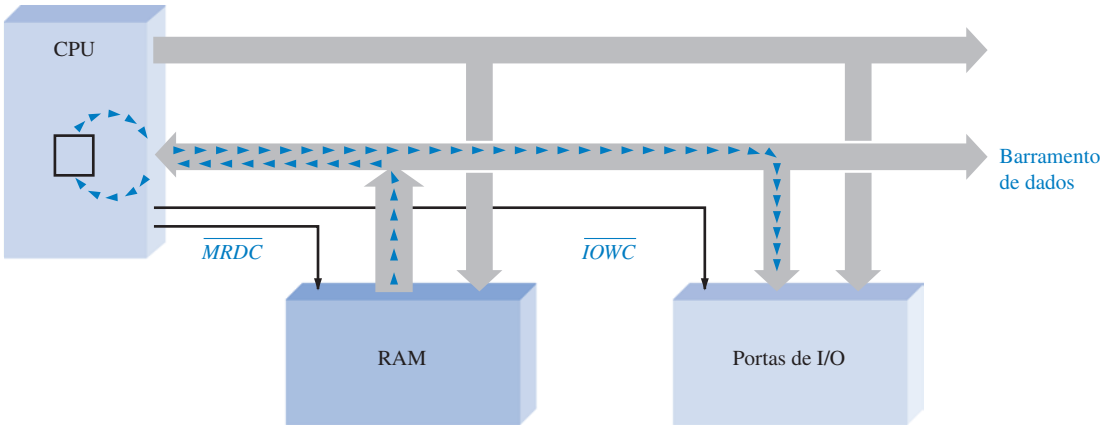
12-6 ACESSO DIRETO À MEMÓRIA (DMA)

Nesta pequena seção é definida a técnica de transferência de dados denominada acesso direto à memória (DMA – *direct memory access*). É apresentada também uma comparação entre a transferência de dados controlada pela CPU e a transferência DMA.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir o termo DMA ■ Comparar uma transferência de dados entre memória e I/O controlada pela CPU com uma transferência DMA

Todas as transferências de dados de I/O discutidas até este momento passaram pela CPU. Por exemplo, quando os dados são transferidos da RAM para um dispositivo periférico, a CPU lê o primeiro byte de dado da memória e o carrega num registrador interno ao microprocessador. Em seguida a CPU escreve o byte de dado na porta de I/O apropriada. Essa operação de leitura/escrita é repetida para cada byte do grupo de dados a ser transferido. A Figura 12–26 ilustra esse processo.

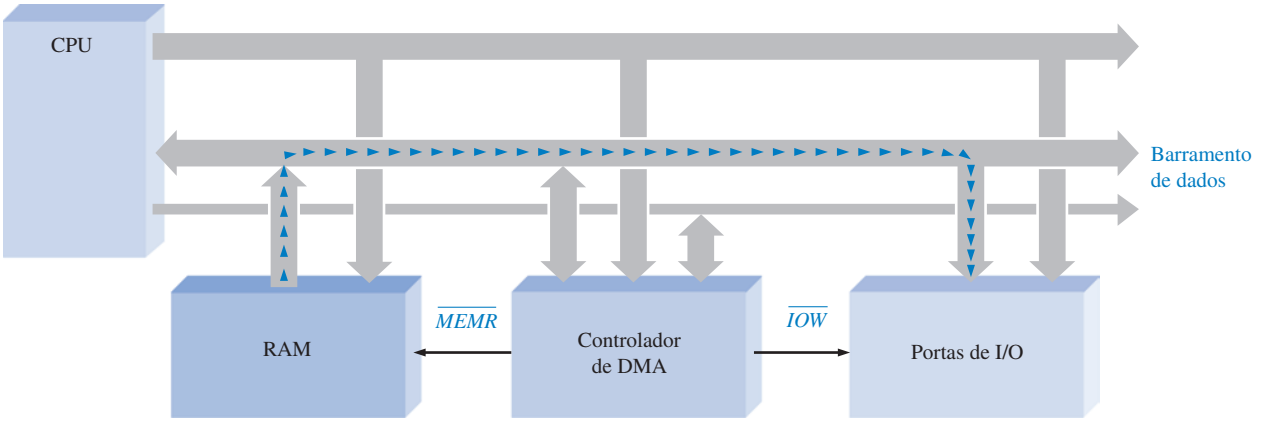


▲ **FIGURA 12–26**

Uma transferência de memória para I/O controlada pela CPU.

Para grandes blocos de dados, paradas intermediárias feitas pelo microprocessador consomem muito tempo. Por essa razão, muitos sistemas usam uma técnica denominada de **DMA** (acesso direto à memória) para aumentar a velocidade na transferência de dados entre a RAM e certos dispositivos periféricos. Basicamente, o DMA realiza um *bypass* (desvio) na CPU para certos tipos de transferência de dados, eliminando portanto o tempo consumido nos ciclos normais de busca e execução necessários para cada operação de leitura ou escrita.

Para transferências diretas com a memória, um dispositivo denominado de controlador de DMA assume o controle dos barramentos do sistema e permite que os dados passem diretamente entre a RAM e o dispositivo periférico, conforme indicado na Figura 12–27. As transferências entre o *drive* de disco e a RAM são particularmente apropriadas para DMA por causa da grande quantidade de dados envolvida e a natureza serial das transferências. O controlador de DMA pode operar transferência de dados várias vezes mais rápido que a CPU.



▲ **FIGURA 12–27**

Uma transferência DMA.

SEÇÃO 12–6 REVISÃO

1. O que significa DMA?
2. Discuta a vantagem do DMA e apresente um exemplo de um tipo de transferência na qual o DMA é freqüentemente usado.

12-7 INTERFACEAMENTO INTERNO

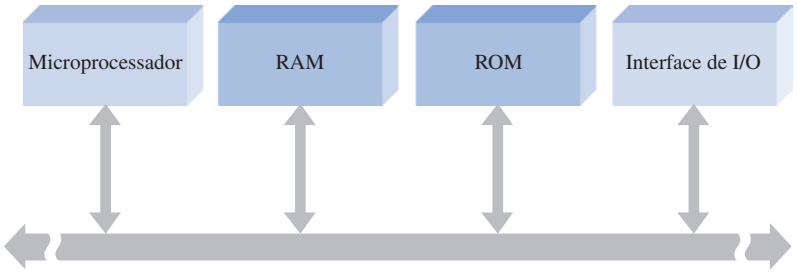
Conforme estudamos, todos os componentes em um computador são interconectados através de barramentos, os quais servem para interconectar dois ou mais componentes funcionais de um sistema ou vários sistemas diferentes. Eletricamente, um barramento é uma coleção de níveis de tensão especificados e/ou níveis de corrente e sinais os quais permitem que vários dispositivos conectados ao barramento trabalhem juntos corretamente.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir o conceito de barramento multiplexado ■ Explicar a razão das saídas tristate

Fundamentos de Barramentos Multiplexados

Em computadores o microprocessador controla e se comunica com as memórias e os dispositivos I/O via estrutura de barramento interno, conforme indicado na Figura 12–38. Um barramento é multiplexado de forma que qualquer dos dispositivos conectados a ele possa enviar ou receber dados de outros dispositivos. Um dispositivo que transmite é freqüentemente denominado de **fonte**, e um dispositivo que recebe é freqüentemente denominado de **aceitador**. Num instante qualquer existe apenas uma fonte ativa. Por exemplo, a RAM pode estar enviando dados para uma interface de I/O sob o controle do microprocessador.



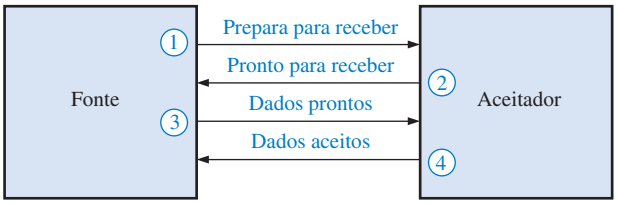
▲ **FIGURA 12–28**

Interconexão dos componentes de um sistema microprocessado através de um barramento multiplexado e bidirecional.

Sinais de Barramento

Em um barramento de controle síncrono, o microprocessador geralmente gera todos os sinais de controle e temporização. Os outros dispositivos sincronizam então suas operações através dos sinais de controle e temporização. Em um barramento de controle assíncrono, os sinais de controle e temporização são gerados em conjunto pela fonte e pelo aceitador. O processo do estabelecimento conjunto de comunicação é denominado ***handshaking***. Um exemplo simples de uma seqüência de *handshaking* é dado na Figura 12–29.

Uma função de controle importante é denominada de **arbitragem de barramento**. A arbitragem evita que duas fontes tentem usar o barramento ao mesmo tempo.

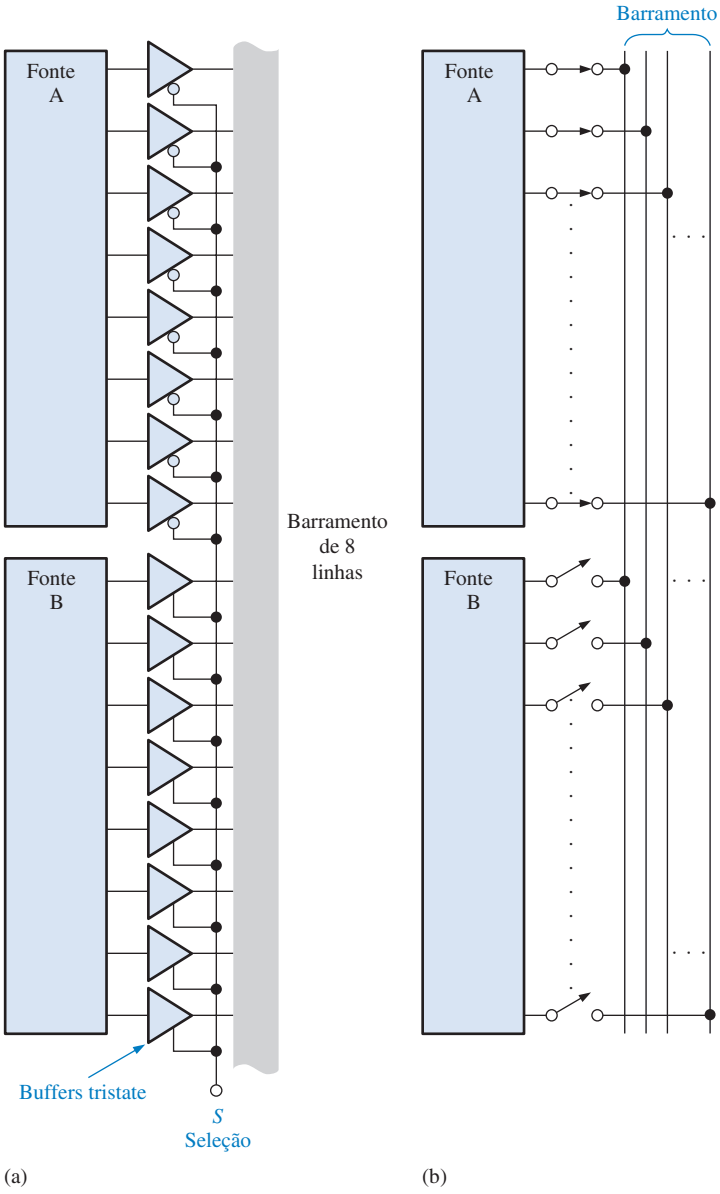


◀ **FIGURA 12–29**
Exemplo de uma seqüência de *handshaking*.

Conexão de Dispositivos a um Barramento

Buffers **tristate** são normalmente usados para interfacear as saídas de um dispositivo fonte com um barramento. Geralmente mais de uma fonte é conectada em um barramento, porém apenas uma pode ter acesso de cada vez. Todas as outras fontes têm que estar desconectadas do barramento para evitar **contenção de barramento** (choque de dados).

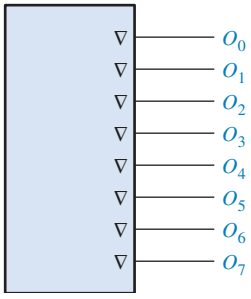
Os circuitos tristate são usados para conectar uma fonte a um barramento ou então desconectá-la, conforme ilustrado na Figura 12–30(a) para o caso de duas fontes. A entrada selecionada é



◀ **FIGURA 12–30**
Interface com um barramento através de um buffer tristate.

usada para conectar a fonte *A* ou a fonte *B* ao barramento, porém não as duas ao mesmo tempo. Quando a entrada de seleção for nível BAIXO, a fonte *A* é conectada e a fonte *B* é desconectada. Quando a entrada de seleção for nível ALTO, a fonte *B* é conectada e a fonte *A* é desconectada. Um circuito equivalente com chaves da ação tristate é mostrado na parte (b) da figura.

Quando a entrada de habilitação do circuito tristate não estiver ativa, o dispositivo estará no estado de alta impedância (**Hi-*Z***) e funciona como uma chave aberta. Muitos CIs digitais possuem buffers tristate internos nas linhas de saída. Uma saída tristate é indicada por pelo símbolo ∇ como mostrado na Figura 12–31.



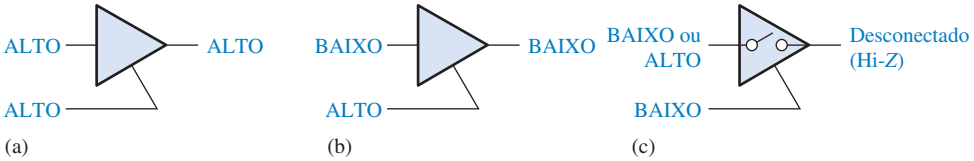
▶ **FIGURA 12–31**
Método de indicação de saídas tristate no símbolo de um CI.

Operação de um Buffer Tristate A Figura 12–32(a) mostra o símbolo lógico para um buffer tristate não-inversor com uma entrada de habilitação ativa em nível ALTO. A parte (b) da figura mostra um buffer com entrada de habilitação ativa em nível BAIXO.

(a) Habilitação ativa em nível ALTO (b) Habilitação ativa em nível BAIXO

▶ **FIGURA 12–32**
Símbolos de buffer tristate.

A operação básica de um buffer tristate pode ser entendida em termos da ação de uma chave como ilustra a Figura 12–33. Quando a entrada de habilitação está ativa, a porta opera como um circuito não-inversor normal. Ou seja, a saída é nível ALTO quando a entrada é nível ALTO e é nível BAIXO quando a entrada é nível BAIXO, como mostra as partes (a) e (b) da figura, respectivamente. Os níveis ALTO e BAIXO representam dois dos estados. O buffer opera no seu terceiro estado quando a entrada de habilitação não estiver ativa. Nesse estado, o circuito funciona como uma chave aberta, sendo que a saída fica completamente desconectada da entrada como mostra a parte (c) da figura. Isso algumas vezes é denominado de estado de *alta impedância* ou *Hi-Z*.



(a) (b) (c)

▲ **FIGURA 12–33**
Operação de um buffer tristate.

Muitos microprocessadores, memórias e outras funções em circuitos integrados têm buffers tristate que servem para interface com barramentos. Tais buffers são necessários quando dois ou mais dispositivos são conectados a um barramento comum. Para evitar que um dispositivo interfira com o outro, os buffers tristate são usados para desconectar todos os dispositivos exceto aqueles que se comunicam num determinado instante.

Contenção de Barramento

A contenção de barramento ocorre quando dois ou mais dispositivos tentam colocar níveis lógicos opostos na mesma linha de um barramento. A forma mais comum da contenção de barramento é quando um dos dispositivos não é totalmente desligado antes que um outro dispositivo conectado ao barramento seja ligado. Isso geralmente ocorre em sistemas de memória quando ocorre uma comutação do modo de LEITURA para o modo de ESCRITA ou vice-versa sendo um problema de temporização.

I/Os Multiplexados

Alguns dispositivos que enviam e recebem dados combinam linhas de entrada e saída, denominadas de portas de I/O, as quais têm que ser multiplexadas e interligadas ao barramento de dados. Buffers tristate bidirecionais fazem a interface desse tipo de dispositivo com o barramento, conforme ilustra a Figura 12–34(a).

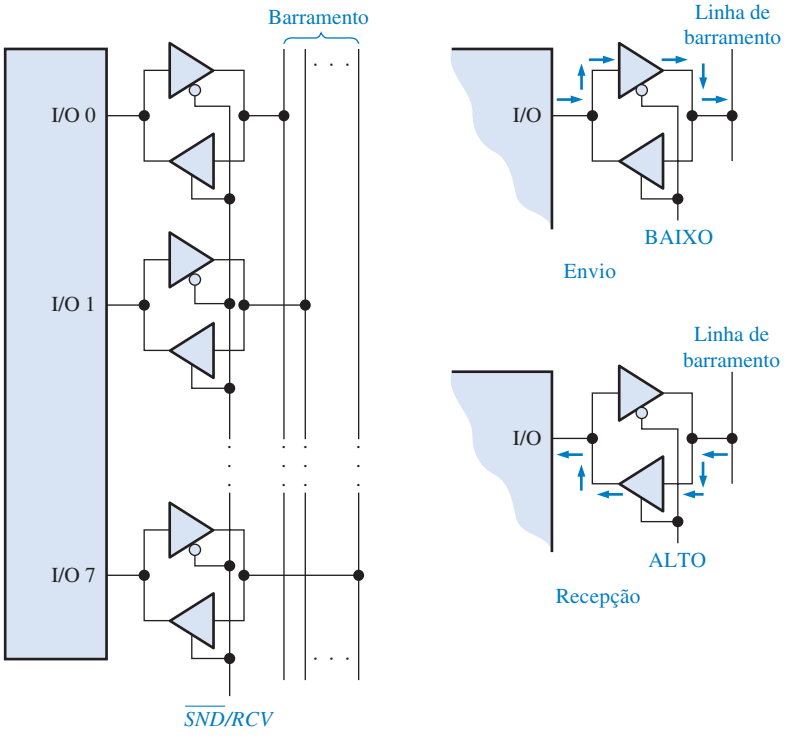


FIGURA 12–34
Operação de I/O multiplexado.

Cada porta de I/O tem um par de buffers tristate. Quando a linha $\overline{SND}/RCV$ ($\overline{\text{Send}}$/Receive), que controla o envio (*send*) e a recepção (*receive*), é nível BAIXO, o buffer tristate superior de cada par é habilitado de forma que o dispositivo funciona como um aceitador recebendo os dados do barramento. Essa operação está ilustrada na Figura 12–34(b). Alguns dispositivos possuem um circuito interno que provê a operação de I/O multiplexado.

SEÇÃO 12–7 REVISÃO

1. Por que os buffers tristate são necessários para interfacear dispositivos digitais com um barramento?
2. Qual é a finalidade de um sistema de barramento?

12-8 BARRAMENTOS PADRÃO

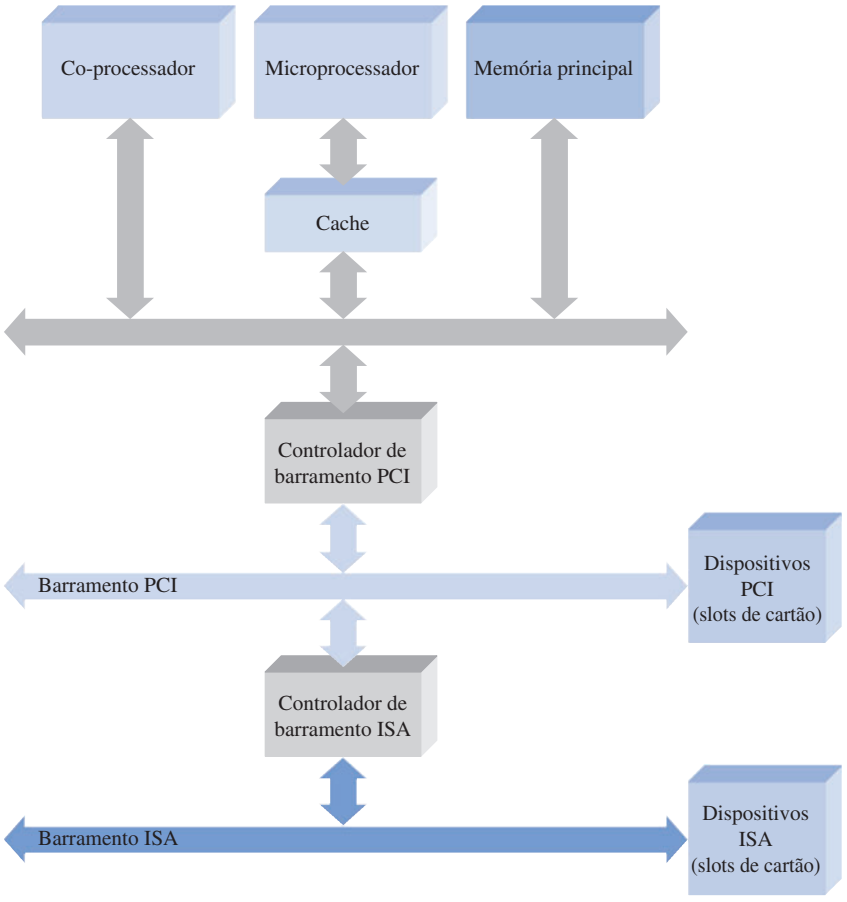
Podemos imaginar um barramento como uma "auto-estrada" para os sinais digitais; ele consiste de um conjunto de conexões físicas (trilhas de circuito impresso ou fios) ao longo dos quais os dados e outras informações são movidos de um lugar para outro. Um barramento também consiste de um conjunto padrão de especificações que determinam as características e tipos de sinais que podem trafegar ao longo de suas vias. Os barramentos internos interconectam os diversos componentes dentro de um sistema de computador: processador, memória, *drive* de disco, controlador e cartões de interface. Os barramentos de I/O ou externos proporcionam a transferência de sinais digitais entre um computador e o "mundo externo" e fazem a interface do computador com equipamentos periféricos (monitor de vídeo, teclado, mouse e impressora) ou com outros equipamentos que sejam controlados por um computador, tais como instrumentos de teste e medição.

Ao final do estudo desta seção você deverá ser capaz de:

■ Discutir os diversos barramentos, serial e paralelo, internos e externos ■ Definir barramento local ■ Descrever os padrões de barramento interno PCI e ISA ■ Descrever o barramento RS-232C ■ Descrever o FireWire ■ Discutir o USB ■ Explicar o GPIB ■ Discutir o SCSI

Barramentos Internos

Os barramentos internos de um computador transportam sinais de endereços, dados e controle entre o microprocessador, memória *cache*, SRAM, DRAM, *drives* de disco, slots de expansão e outros dispositivos internos. Computadores pessoais consistem de três tipos de barramentos internos: o *barramento local*, o *barramento PCI* e o *barramento ISA*. A Figura 12–35 mostra o arranjo básico de um sistema de barramento.



▶ **FIGURA 12–35**
Ilustração simplificada de um sistema de barramento básico em um computador pessoal típico.

Barramento Local Esse barramento conecta diretamente o microprocessador à memória *cache*, memória principal, co-processador e controlador de barramento PCI. O **barramento local** é o único barramento interno que conecta diretamente ao microprocessador. Geralmente, esse barramento inclui o barramento de dados, barramento de endereço e o barramento de controle que permitem o microprocessador comunicar com outros dispositivos. O barramento local pode ser considerado como um barramento *primário* em um sistema de computador. Por exemplo, o barramento local do Pentium consiste de um barramento de endereço contendo 32 linhas de endereço de memória, um barramento de dados contendo 64 linhas e um barramento de controle contendo diversas linhas de controle.

Barramento PCI Esse barramento faz a interface do microprocessador com dispositivos externos via slots de expansão (conectores). O **barramento PCI** (*Peripheral Control Interconnect*) foi desenvolvido pela Intel; e como foi introduzido em 1993, se tornou o padrão de barramento de interface de computadores pessoais, substituindo diversos padrões de barramentos anteriores. O barramento PCI é de 64 bits, embora seja frequentemente implementado como um barramento de 32 bits no qual os barramentos de endereço e dados são multiplexados. Ele pode operar com uma velocidade de clock de 33 MHz ou 66 MHz.

O barramento PCI está isolado do barramento local por uma unidade controladora de barramento que funciona como uma "ponte" entre os dois barramentos. O barramento PCI é considerado *secundário* e opera com clock independente do microprocessador. O PCI pode conectar o microprocessador a dispositivos periféricos como o *drive* de disco rígido, via slots de expansão com cartões adaptadores.

O PCI suporta *plug-and-play*, a capacidade de um computador configurar automaticamente placas de expansão e outros dispositivos. Isso permite que um dispositivo seja conectado a um computador sem a preocupação de posicionar chaves, mudar *jumpers* de fio ou lidar com muitos outros elementos de configuração. Isso é feito com uma memória de 256 bytes que permite ao computador interrogar a interface PCI.

Barramento ISA Esse barramento de expansão foi desenvolvido pela IBM para o seu computador pessoal AT e é o barramento padrão no qual se conecta praticamente todo cartão de circuito impresso feito antes de 1993. O barramento **ISA** (*Industry Standard Architecture*) é atualmente incorporado na maioria dos computadores pessoais modernos como um parceiro do barramento PCI, para fins de compatibilidade com dispositivos anteriores.

O ISA tem um barramento de dados de 8 ou 16 bits e pode operar em 8,33 MHz. Uma versão expandida denominada de EISA provê um barramento de dados de 32 bits, porém ele foi amplamente descontinuado devido à sua baixa velocidade e substituído pelo barramento PCI.

Barramentos Externos

Dispositivos externos são conectados a um computador via interface de entrada/saída (I/O) denominadas de portas de I/O. Existem dois tipos básicos de portas de I/O em computadores, a *porta serial* e a *porta paralela*, sendo que a maioria dos computadores tem uma porta paralela e pelo menos uma porta serial para conexão de modems, impressoras, mice (o plural de mouse) e outros dispositivos periféricos.

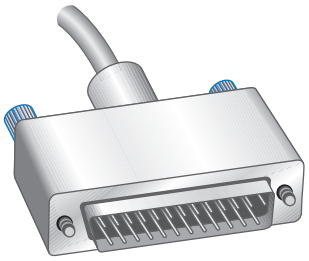
Uma porta serial é usada para comunicação serial de dados, onde apenas 1 bit é transferido de cada vez. Os modems e os mice são exemplos de dispositivos seriais típicos. Além disso, as portas seriais são algumas vezes usadas para fazer a interface de equipamentos de teste e medição com um computador. Uma porta paralela é usada para comunicação paralela de dados, onda pelo menos 1 byte (8 bits) é transferido de cada vez. Existem atualmente vários padrões de barramento em uso para portas serial e paralela. As mais proeminentes são descritas a seguir.

Barramento de Interface de I/O Serial

RS-232C Esse é um dos padrões mais antigos e comuns para interface serial aprovado pela Electronic Industries Association (EIA). O RS-232C também é conhecido como EIA-232. A maioria dos **modems** (*mo*dulador/*dem*odulador) conforme o padrão EIA-232 e a maioria dos computadores pessoais têm uma porta RS-232C. O mouse, algumas telas de display e impressoras seriais — além dos modems — são projetados para serem conectados a uma porta serial RS-232C.

essa porta é normalmente usada para fazer a interface de um equipamento terminal de dados (**DTE**) com um equipamento de comunicação de dados (**DCE**). Por exemplo, um computador é classificado como DTE e um modem como DCE.

O padrão EIA-232 especifica vinte e cinco linhas entre um DTE e um DCE necessitando de um conector de 25 pinos (DB-25), como mostra a Figura 12–36. Em aplicações de computador pessoal, nem todos os sinais RS-232C são necessários. São usados tipicamente um mínimo de três e um máximo de onze sinais. Por essa razão, foi definida uma conexão de 9 pinos (DB-9) pela IBM para sua interface serial.



▶ **FIGURA 12–36**
Conector de 25 pinos da RS-232C.

A Figura 12–37(a) apresenta uma lista de sinais e os pinos associados para um conector RS-232C de 25 pinos e a parte (b) da figura apresenta os sinais e os pinos associados para um conector de 9 pinos. Os onze pinos e sinais mostrados em laranja na parte (a) indicam os sinais tipicamente usados em aplicações de computador pessoal, sendo que os três sinais mínimos são indicados por um asterisco (pinos 2, 3 e 7).

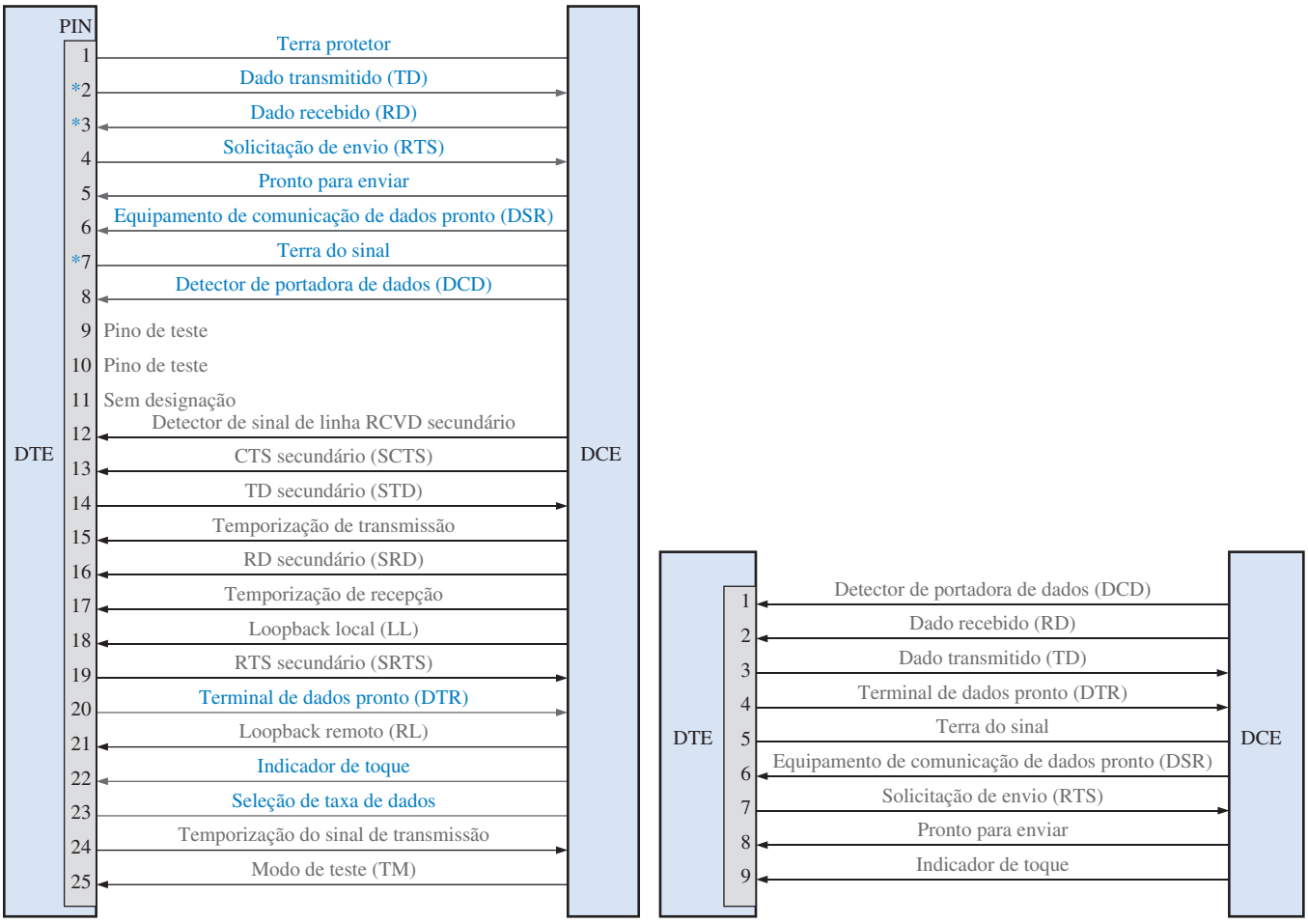
O comprimento do cabo máximo especificado para o RS-232C é 50 pés (cerca de 15,24 m) com uma taxa de transferência de dados de 20 kbaund. Caso seja usado um cabo mais curto, a taxa em baunds pode ser maior. A especificação da taxa de transferência de dados em baunds e bits por segundo (b/s) não é necessariamente sempre igual. Isso porque a taxa em baunds é uma terminologia de modems e é definida como o número de mudanças de sinal por segundo, que é denominado de taxa de modulação. Em modems, uma mudança de sinal algumas vezes transfere vários bits de dados. Em taxas menores, o baud é igual a bits por segundo, porém em taxas maiores o baud pode se menor que bits por segundo.

Para superar as limitações do RS-232C, dois outros padrões, o RS-422 e o RS-423, foram desenvolvidos. Esses padrões mais novos especificam comprimentos de cabos muito mais longos e taxas de dados maiores sob certas condições. Por exemplo, o RS-422 e o RS-423 especificam um comprimento máximo de cabo de 4000 pés (cerca de 1219,2 m). A máxima taxa de transferência de dados para RS-422 é 10 Mbaud para um cabo de 40 pés (cerca de 12,19 m) e 100 kbqud para 4000 pés (cerca de 1219,2 m). Para o RS-423, a taxa de transferência de dados é 100 kbaud para um cabo de 30 pés (cerca de 9,14 m) e 1 kbaud para 4000 pés (cerca de 1219,2 m). O RS-232C ainda permanece o padrão mais comum.

IEEE 1394 Esse barramento serial externo padrão suporta taxas de transferências de dados de até 400 Mb/s e é usado tipicamente para fazer a interface, embora sem limitação, com periféricos de vídeo e gráfico, tais como câmeras digitais. O padrão **IEEE 1394** é freqüentemente denominado de **FireWire**, uma marca registrada pela Apple Computer, que foi a primeira empresa a desenvolvê-lo. Outras companhias usam outros nomes para descrever os produtos IEEE 1394. IEEE significa *Institute of Electrical AND Electronics Engineers*.

Até 63 dispositivos podem ser conectados ao FireWire baseado em um arranjo tipo flor. O cabo FireWire consiste de seis fios, dois pares torcidos para dados e dois para alimentação. Além disso, esse padrão permite "conexão a quente", que é a capacidade de inserir ou remover dispositivos conectados ao computador enquanto ele estiver executando um programa.

USB A **USB** (*Universal Serial Bus*) suporta duas taxas de transferências de dados, sendo uma taxa de alta velocidade de 12 Mb/s e uma taxa de velocidade baixa de 1,5 Mb/s. Uma porta USB pode ser usada para conectar até 127 dispositivos periféricos e permite conexão a quente e *plug-*



(a) (a) Interface RS-232C completa de 25 pinos com a configuração típica para computador pessoal indicada em laranja e os três sinais mínimos marcados com um asterisco (pinos 2, 3 e 7).

(b) Interface RS-232C de 9 pinos.

▲ **FIGURA 12–37**

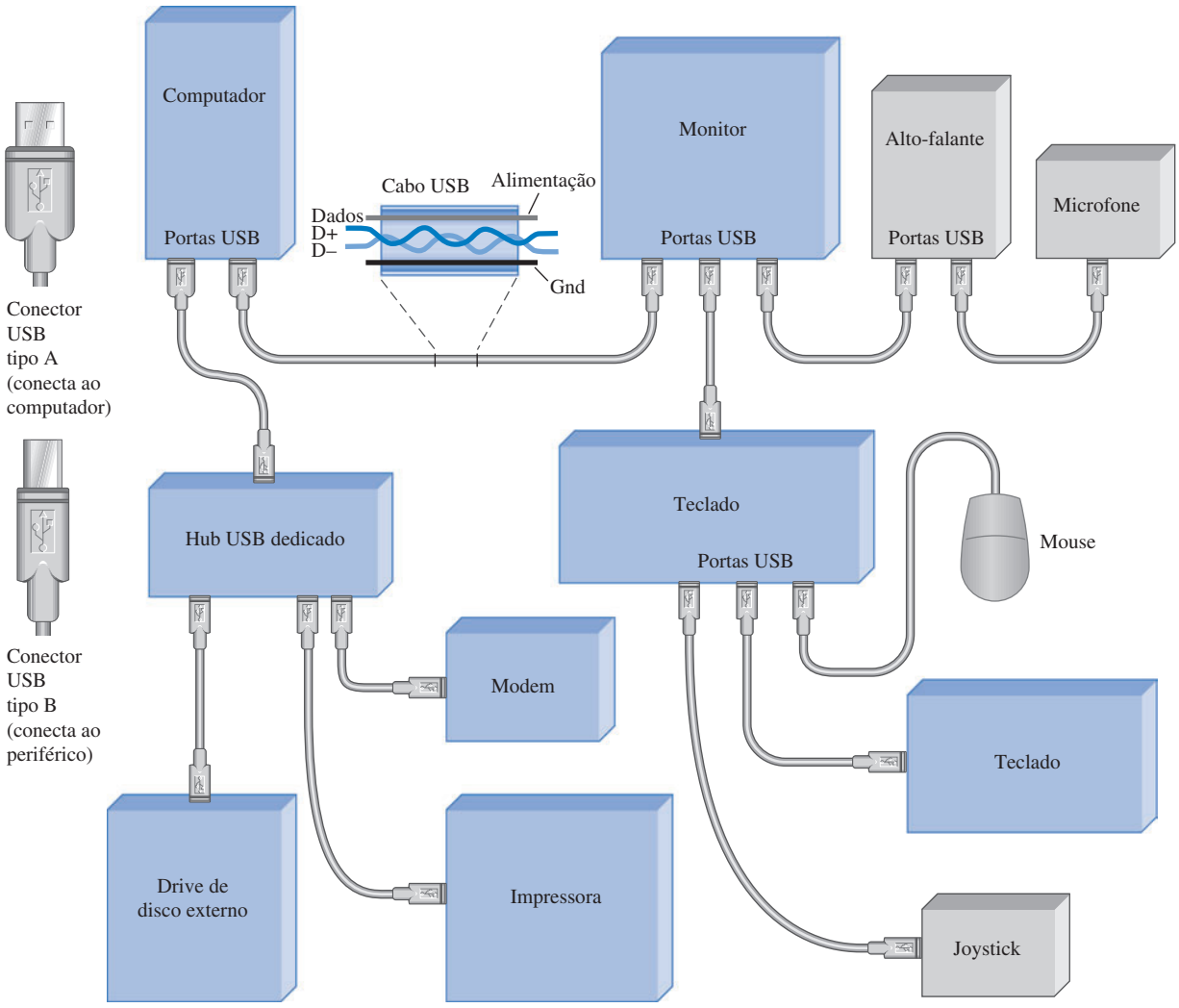
A designação dos pinos e sinais para os dois conectores RS-232C.

and-play. O cabo USB tem quatro fios, dois para dados e dois para alimentação, e conecta o computador a um dispositivo periférico USB, sendo que qualquer um pode funcionar como *hub* para a conexão de outros dispositivos periféricos USB. A Figura 12–38 ilustra um sistema de computador com interface USB.

Barramentos de Interface de I/O Paralelo

IEEE 488 Esse padrão de barramento surgiu há muito tempo e também é conhecido como General-Purpose Interface Bus (**GPIB**). Amplamente usada em aplicações de testes e medições, ela foi desenvolvida pela Hewlett-Packard (HP) nos anos de 1960. O **IEEE 488** especifica 24 linhas que são usadas para transferir oito bits de dado em paralelo de cada vez e provê oito sinais de controle que incluem três linhas de *handshake* e cinco linhas de gerenciamento do barramento. Além disso estão incluídos oito linhas de GND usadas para blindagem e retorno de GND. A taxa de transferência máxima para o padrão IEEE 488 é 1 MB/s. Um padrão superior a esse, denominado de HS488, tem uma taxa de dados máxima de 8 MB/s.

Para conectar um equipamento de teste a um computador usando o barramento IEEE 488, é instalado no computador um cartão de interface o qual transforma o computador em um **contro-**



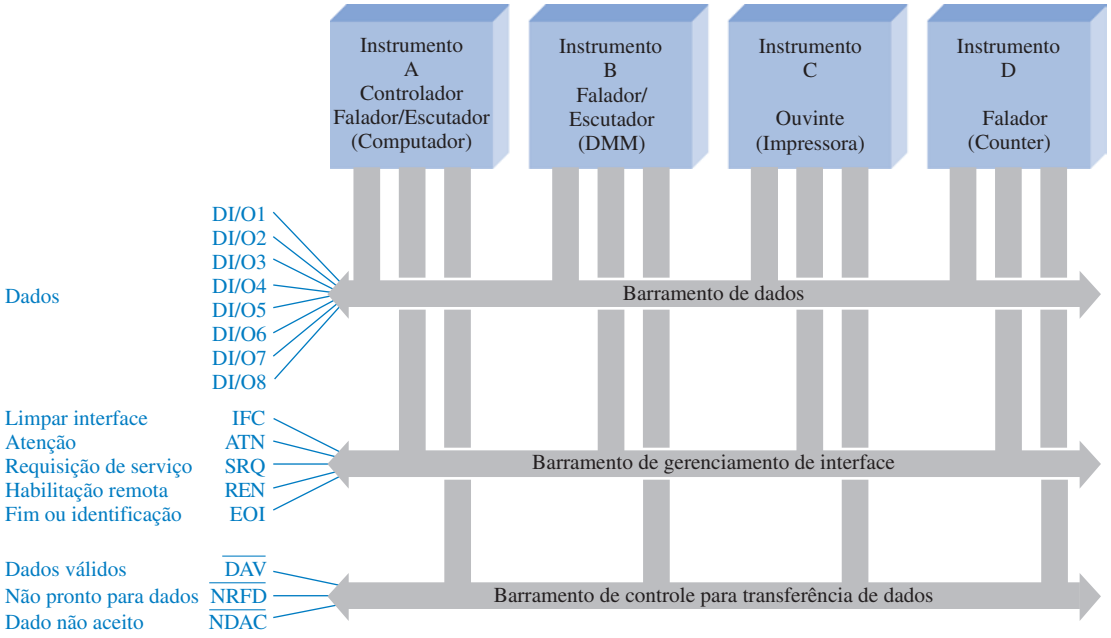
▲ **FIGURA 12–38**

Exemplo de um sistema de computador com interface USB.

lador de sistema. Em uma configuração típica de GPIB, podem ser conectados até 14 dispositivos controlados (instrumentos de teste e medição). Quando o controlador de sistema emite um comando para um dispositivo controlado realizar uma operação especificada, tal como uma medição de freqüência, dizemos que o controlador "fala" e o dispositivo controlado "ouve".

Um "**ouvinte**" é um instrumento capaz de receber dados por meio do GPIB quando endereçado pelo controlador do sistema (computador). Exemplos desse tipo são as impressoras, monitores, fontes de alimentação programáveis e geradores de sinais programáveis. Um "**falante**" é um instrumento capaz de enviar dados por meio do GPIB. São exemplos os DMMs (multímetros digitais) e os freqüencímetros que podem disponibilizar dados em um barramento compatível. Alguns instrumentos podem enviar e receber dados e são denominados "falantes/ouvintes"; são exemplos os computadores, modems e certos instrumentos de medição. O controlador do sistema pode especificar cada um dos outros instrumentos no barramento como "falante" ou "ouvinte" para fins de transferência de dados. O controlador é geralmente um "falante/ouvinte".

Um arranjo GPIB típico é mostrado na Figura 12–39 como exemplo. Os três barramentos básicos com agrupamentos de sinais são mostrados como *barramento de dados*, *barramento de controle para transferência de dados* e *barramento de gerenciamento de interface*.

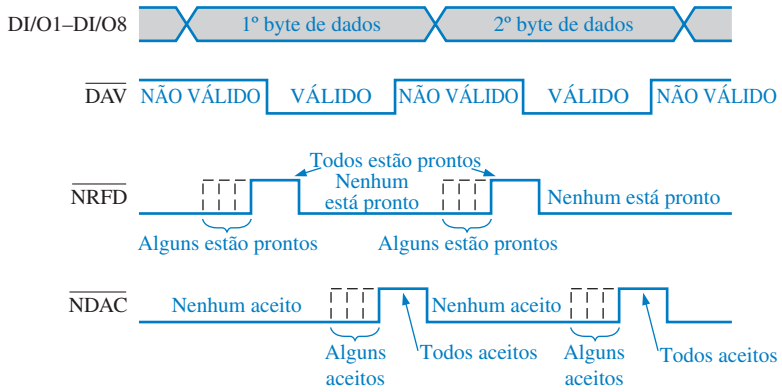


▲ **FIGURA 12–39**

Uma conexão típica IEEE 488 (GPIB).

As linhas de dados em paralelo são designadas por DI/O1 até DI/O8 (entrada/saída de dados). Um byte de dado é transferido na parte bidirecional do barramento. Cada byte que é transferido passa por uma operação de *handshaking* via controle de transferência de dados. As três linhas de *handshaking* ativas em nível BAIXO indicam se os dados são válidos ($\overline{\text{DAV}}$), se o instrumento endereçado não está pronto para os dados ($\overline{\text{NRFD}}$), ou se os dados não foram aceitos ($\overline{\text{NDAC}}$). Mais de um instrumento pode aceitar dados ao mesmo tempo e o instrumento mais lento determina a taxa de transferência. A Figura 12–40 mostra o diagrama de temporização para a seqüência de *handshaking* GPIB e a Tabela 12–2 descreve os sinais de *handshaking*.

Os cinco sinais do barramento de gerenciamento de interface controla ordenadamente o fluxo de dados. A linha ATN (atenção) é monitorada por todos os instrumentos no barramento. Quando ATN está ativa, o controlador do sistema seleciona a operação de interface específica, determina os "falantes" e os "ouvintes", e provê o endereçamento específico para os "ouvintes". Cada ins-



▲ **FIGURA 12–40**

Diagrama de temporização para a seqüência de *handshaking* GPIB.

TABELA 12–2

Os sinais de *handshaking* GPIB

NOME	DESCRIÇÃO
$\overline{\text{DAV}}$	**Dados válidos:** Após o "falante" detectar um nível ALTO na linha $\overline{\text{NRFD}}$, é colocado um nível BAIXO nessa linha () pelo "falante" quando os dados em suas linhas de I/O estão estáveis e válidos.
$\overline{\text{NRFD}}$	**Não está pronto para dados:** O "ouvinte" coloca um nível baixo nessa linha para indicar que não está pronto para dados. Um nível ALTO indica que ele está pronto. A linha $\overline{\text{NRFD}}$ não irá para nível ALTO até que todos os "ouvintes" endereçados estejam prontos para aceitar dados.
$\overline{\text{NDAC}}$	**Dados não aceitos:** O "ouvinte" coloca um nível BAIXO nessa linha para indicar que ele não aceitou os dados. Quando ele aceita os dados a partir das linhas de I/O, ele libera a linha $\overline{\text{NDAC}}$. Essa linha $\overline{\text{NDAC}}$ do "falante" não vai para nível ALTO até que o último "ouvinte" tenha aceito os dados.

TABELA 12–3

As linhas de gerenciamento do GPIB

NOME	DESCRIÇÃO
ATN	**Atenção:** Faz com que todos os dispositivos no barramento interprete os dados, como um comando ou endereço do controlador e ativa a função *handshaking*.
IFC	**Limpa a Interface:** Inicializa o barramento.
SRQ	**Requisição de Service:** Alerta o controlador que um dispositivo solicita comunicação.
REN	**Habilitação Remota:** Habilita dispositivos a responder ao controle do programa remoto.
EOI	**Fim ou identidade:** Indica o último byte de dados a ser transferido.

trumento GPIB tem um endereço de identificação específico que é usado pelo controlador do sistema. A Tabela 12–3 descreve as linhas de gerenciamento da interface GPIB e suas funções.

O GPIB está limitado a um comprimento máximo de cabo de 15 m podendo haver não mais que um instrumento por metro com uma capacitância de carga máxima de 50 pF cada um. A limitação no comprimento do cabo pode ser superada usando extensores de barramento e modems. Um extensor de barramento provê um interfaceamento via cabo de instrumentos que estão separados por uma distância maior que a permitida pelas especificações GPIB ou para comunicações em grandes distâncias via linha telefônica interfaceada com modem. O uso de extensões de barramento e/ou modems está ilustrado na Figura 12–41.

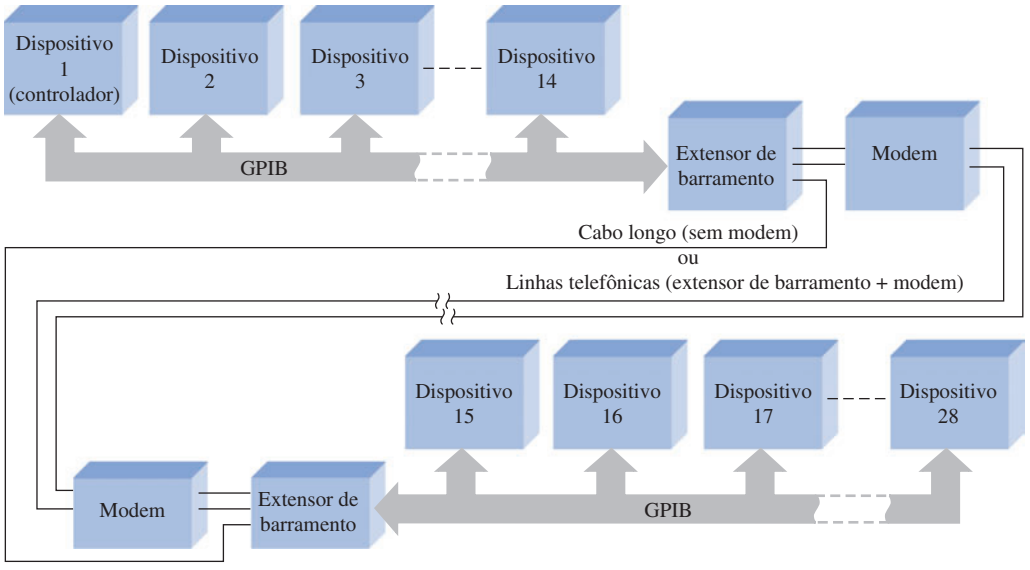


FIGURA 12–41

Um extensor de barramento e um modem podem ser usados para interface de sistemas GPIB remotos.

SCSI Pronunciada "*scâzi*", esse é um padrão amplamente usado na interface de computadores pessoais e periféricos. Embora o **SCSI** (*Small Computer System Interface*) seja um padrão ANSI (*American National Standards Institute*), existem diversas variações e tipos de conectores produzidos por uma variedade de fabricantes. Um tipo de SCSI pode não ser compatível com outro. SCSI-1 é uma versão com conector de 25 pinos que provê um barramento de dados de 8 bits e suporta taxas de transferência de dados de 4 MB/s. Algumas outras versões do padrão de barramento SCSI são mostradas a seguir:

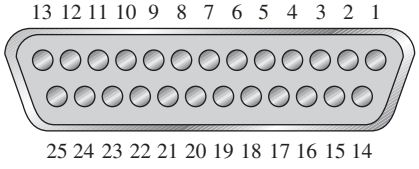
- *SCSI-2* Essa versão é a mesma que a SCSI-1, porém usa um conector de 50 pinos e suporta dispositivos múltiplos.
- *Wide SCSI* Essa usa um conector maior que o SCSI-2 para suportar transferências de dados de 16 bits.
- *Fast SCSI* Essa provê transferência de dados de 8 bits, porém suporta taxas de transferência de 10 MB/s.
- *Fast Wide SCSI* Essa versão permite transferência de dados de 16 bits a 20 MB/s.
- *Ultra SCSI* Essa versão transfere dados de 8 bits a 20 MB/s.
- *SCSI-3* Essa versão tem linhas de dados de 16 bits e opera a 40 MB/s.
- *Ultra SCSI-2* Essa versão transfere transfere 8 bits a 40 MB/s.
- *Wide Ultra SCSI-2* Essa versão provê transferência de dados de 16 bits e opera a 80 MB/s.

As descrições dos sinais para um conector SCSI de 25 pinos são apresentadas na Tabela 12–4 e a configuração de pinos é mostrada na Figura 12–42.

▼ TABELA 12–4

Sinais SCSI

NÚMERO DO PINO	NOME DO SINAL	DESCRIÇÃO DO SINAL	NÚMERO DO PINO	NOME DO SINAL	DESCRIÇÃO DO SINAL
1	REQ/	Requisição	14	GND	Terra de sinal
2	MSG/	Mensagem	15	C/D/	Comando/Dado
3	I/O/	Entrada/Saída	16	GND	Terra de sinal
4	RST/	Resete do barramento SCSI	17	ATN/	Atenção
			18	GND	Terra de sinal
5	ACK/	Reconhecimento	19	SEL/	Seleção
6	BSY/	Ocupado	20	DBP/	Paridade do dado
7	GND	Terra de sinal	21	DB1/	Bit 1 do dado
8	DB0/	Bit 0 do dado	22	DB2/	Bit 2 do dado
9	GND	Terra de sinal	23	DB4/	Bit 4 do dado
10	DB3/	Bit 3 do dado	24	GND	Terra de sinal
11	DB5/	Bit 5 do dado	25	TPWR	Terminal de alimentação
12	DB6/	Bit 6 do dado			
13	DB7/	Bit 7 do dado			



◄ FIGURA 12–42

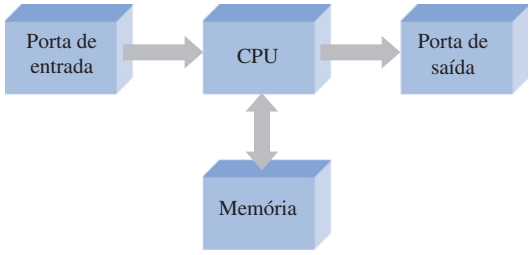
Conector SCSI de 25 pinos.

SEÇÃO 12–8 REVISÃO

1. Cite as duas principais categorias de barramento em termos da forma com que os dados são transferidos.
2. Classifique cada um dos seguintes barramentos como serial ou paralelo:
   (a) SCSI **(b)** RS-232C **(c)** USB **(d)** GPIB
3. Explique as diferenças básicas entre os barramentos serial e paralelo.
4. Quantos dispositivos podem ser conectados no barramento USB?
5. O barramento FireWire é mais rápido do que o USB em termos de transferência de dados?

RESUMO

- As unidades básicas de um computador são mostradas na Figura 12–43.



► **FIGURA 12–43**

- Os três barramentos básicos de um computador são o barramento de endereço, barramento de dados e barramento de controle. O tamanho de cada barramento é especificado pelo número de vias independentes.
- Dispositivos periféricos típicos incluem o teclado, *drives* de disco externos, mouse, impressora, modem e scanner.
- O número de linhas de endereço foi ampliado de 20, para o 8086/8088, para 32, para a família Pentium. O barramento de dados é de 16 bits para o 8086 e 64 bits para a família Pentium.
- Os registradores gerais são um sub-conjunto dos registradores em todos os processadores Intel. Eles incluem

  Acumulador (AX, que inclui AH e AL)
  Base (BX, que inclui BH e BL)
  Contador (CX, que inclui CH e CL)
  Dado (DX, que inclui DH e DL)
  Ponteiro de pilha (SP)
  Ponteiro base (BP)
  Índice destinação (DI)
  Índice fonte (SI)

  A partir do processador 80386, esse conjunto básico foi expandido para o conjunto de registradores estendidos.
- Os registradores de segmento básicos são um subconjunto dos registradores em todos os processadores Intel. Os registradores de segmento são

  Segmento de código (CS)
  Segmento de dados (DS)
  Segmento extra (ES)
  Segmento de pilha (SS)

  A partir do processador 80386, foram acrescentados dois novos registradores de segmento.
- Os registradores de flag são um subconjunto dos registradores em todos os processadores Intel. Eles incluem

  Trap (TF)
  Direção (DF)
  Habilitação de interrupção (IF)

Overflow (OF)
Sinal (SF)
Zero (ZF)
Carry auxiliar (AF)
Paridade (PF)
Carry (CF)

- A "linguagem" básica de um computador é denominada de código de máquina na qual as instruções são apresentadas como uma série de códigos em binário.
- Na linguagem *assembly*, as instruções de máquina são substituídas por mnemônicos em inglês que têm correspondência com os códigos de máquina. A linguagem *assembly* também usa diretivas para permitir ao programador especificar outros parâmetros que não são traduzidos diretamente para código de máquina.
- As portas de I/O são interfaces para dispositivos externos. Elas podem ser configuradas como entrada, saída ou uma combinação de ambas. Elas podem ser acessadas como de forma dedicada ou mapeada na memória e podem ser servidas por consulta, ativação de interrupção ou por software.
- A Tabela 12–5 compara barramentos padrão.

▼ TABELA 12–5

Tabela-verdade para a lógica AND-OR do circuito mostrado na Figura 5–1.

	BARRAMENTOS INTERNOS		BARRAMENTOS EXTERNOS				
	PCI	ISA	RS-232C	IEEE 1394	USB	IEEE 488	SCSI
Tipo	Paralelo	Paralelo	Serial	Serial	Serial	Paralelo	Paralelo
Linhas de dados	32/64	8/16	—	—	—	8	8/16
Taxa de dados	33/66 MHz	8,33 MHz	20 kbaud	400 Mb/s	1,5/12 Mb/s	1 Mb/s	4 Mb/s (1) 10 Mb/s (Fast) 20 Mb/s (Ultra) 40 Mb/s (3) 80 Mb/s (Ultrawide 2)
Número de dispositivos	—	—	1	63	127	14	16

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Barramento de controle Um conjunto de vias condutoras que conectam a CPU a outras partes do computador para coordenar sua operação e se comunicar com dispositivos externos.

Barramento de dados Um conjunto bidirecional de vias condutoras através das quais os dados ou códigos de instrução são transferidos para o microprocessador ou os resultados das operações são enviados para fora do microprocessador.

Barramento de endereço Um grupo de condutores que interconectam o processador e a memória, ou ainda outros dispositivos externos, nos quais os códigos de endereços são enviados.

CPU Unidade central de processamento; o "cérebro" de um computador que processa as instruções do programa.

FireWire O padrão IEEE-1394 de barramento serial.

GPIB Barramento de interface de propósito geral baseada no padrão IEEE-488.

Interrupção Um sinal de computador ou instrução que faz com que o processamento atual seja paralisado temporariamente enquanto uma rotina de serviço é executada.

Linguagem *assembly* Uma linguagem de programação que usa palavras originárias do inglês e que tem uma correspondência individual com a linguagem de máquina.

Linguagem de alto nível Um tipo de linguagem de computador próxima à linguagem humana que está um nível acima da linguagem *assembly*.

Linguagem de máquina Instruções de computador escritas em código binário que são interpretadas por um computador; o nível mais baixo de linguagem de programação.

Microprocessador Um circuito integrado digital de alta densidade (larga escala de integração) que pode ser programado para realizar operações aritméticas, lógicas ou de outra natureza; a CPU de um computador.

Modem Um modulador/demodulador para interface de dispositivos digitais com sistemas de transmissão analógicos tal como a linha telefônica.

Periférico Um dispositivo tal como uma impressora ou modem que provê comunicação com um computador.

Porta de I/O Uma interface física em um computador através da qual os dados são enviados ou recebidos pelos periféricos.

Programa Uma lista de instruções ordenadas que um computador segue para obter um resultado específico.

SCSI Interface para pequenos sistemas de computadores; um padrão de barramento paralelo externo.

Tristate Um tipo de saída existente em determinados circuitos lógicos que apresenta três estados: ALTO, BAIXO e alta impedância (Hi-Z); usado para interface das saídas de um dispositivo fonte com um barramento.

USB Barramento serial universal; um padrão de barramento serial externo.

AUTOTESTE

As respostas estão no final do capítulo.

1. Um computador básico não inclui
(a) uma unidade lógica e aritmética **(b)** uma unidade de controle
(c) unidades de periféricos **(d)** uma unidade de memória

2. Um barramento de endereço de 20 bits suporta
(a) 100.000 endereços de memória **(b)** 1.048.576 endereços de memória
(c) 2.097.152 endereços de memória **(d)** 20.000 endereços de memória

3. O número de bits no barramento de dados no processador Pentium é
(a) 16 **(b)** 24 **(c)** 32 **(d)** 64

4. Um barramento que é usado pelo microprocessador para transmitir e receber informações é o
(a) barramento de endereço **(b)** barramento de dados
(c) ambos os barramentos acima **(d)** nenhuma das alternativas anteriores

5. Um exemplo de uma unidade periférica é
(a) o registrador de endereço **(b)** a MPU
(c) o monitor de vídeo **(d)** o adaptador de interface

6. Os tipos de transferências de memória controlados pela CPU são
(a) direto e interrupção **(b)** leitura e escrita
(c) por barramento e multiplexada **(d)** entrada e saída

7. Na família Intel, o número máximo de dispositivos I/O de 8 bits é
(a) 64 **(b)** 1000 **(c)** 64.000
(d) 1 milhão **(e)** ilimitado

8. A consulta é um método usado para
(a) determinar o estado do microprocessador
(b) estabelecer comunicações entre a CPU e um periférico
(c) estabelecer uma prioridade para comunicação com diversos periféricos
(d) determinar a próxima instrução

9. Das alternativas a seguir, qual é um registrador de 8 bits?
(a) AH **(b)** BX **(c)** SS **(d)** IP

10. Essencialmente, um mnemônico é um(a)
(a) fluxograma **(b)** operando **(c)** código de máquina **(d)** instrução

11. DMA significa
(a) endereço do microprocessador digital **(b)** acesso direto à memória
(c) acesso multiplexado de dados **(d)** endereçamento direto de memória

12. Um programa de computador é uma lista de
(a) endereços de memória que contém dados a serem usados numa operação.
(b) endereços que contêm instruções a serem usadas em uma operação.
(c) instruções organizadas para se obter um resultado específico.

13. Um tipo de instrução *assembly* que altera o curso de um programa é denominado de
(a) *loop* **(b)** *jump*
(c) as duas alternativas acima. **(d)** nenhuma das alternativas acima.

14. Um tipo de interrupção que é solicitada dentro de um programa é denominada de
(a) interrupção por software **(b)** interrupção por consulta
(c) interrupção direta **(d)** interrupção por I/O

15. A maioria dos dispositivos fazem interface com um barramento através de
(a) saídas totem-pole **(b)** buffers tristate
(c) transistores *pnp* **(d)** resistores

16. O barramento PCI consiste em
(a) 8 a 16 linhas de dados **(b)** 32 a 64 linhas de dados **(c)** 1 linha de dados serial

17. Os dispositivos que operam na interface GPIB são denominados
(a) fonte e carga **(b)** falante e ouvinte
(c) transmissor e receptor **(d)** doador e aceitador

18. O RS-232C é
(a) uma interface padrão para dados em paralelo. **(b)** uma interface padrão para dados em série.
(c) um melhoramento da interface IEEE 488. **(d)** o mesmo que SCSI.

19. O barramento FireWire é o mesmo que
(a) o barramento IEEE 488 **(b)** USB **(c)** IEEE 1394
(d) RS-422 **(e)** RS-423

20. O barramento USB pode suportar até
(a) 63 dispositivos **(b)** 14 dispositivos
(c) 100 dispositivos **(d)** 127 dispositivos

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 12–1 O Computador Básico

1. Cite os elementos básicos de um computador.
2. Cite duas categorias de software de computador.
3. O que é um barramento?
4. O que é uma porta de I/O?

SEÇÃO 12–2 Microprocessadores

5. Cite os elementos básicos de um microprocessador.
6. Liste as três operações que um microprocessador realiza.
7. Cite os três barramentos do microprocessador.
8. Quais são os sete grupos básicos do conjunto de instruções do Pentium?

SEÇÃO 12–3 Uma Família Específica de Microprocessador

9. Quais são os três passos básicos que um processador repete de forma cíclica?
10. O que significa *pipelining*?
11. Cite os seis registradores de segmento?
12. Considere que o registrador de segmento de código contém o número hexa 0F05 e que o registrador ponteiro de instrução contém o número 0100. Qual o endereço físico da próxima instrução a ser executada?

13. Explique a diferença entre os registradores AH e AL, bem como AX e EAX.

14. **(a)** O que é um flag?
(b) Quais são as duas finalidades para o uso dos flags?

15. Explique a vantagem do emparelhamento de instruções no processador Pentium.

SEÇÃO 12–4 Programação de um Computador

16. O que é um assemblador?

17. Desenhe um fluxograma para um programa que soma os números de um a dez e salva o resultado numa posição de memória denominada TOTAL.

18. Desenhe um fluxograma mostrando como você contaria o número de bytes numa *string* colocando o resultado numa posição de memória denominada COUNT. Considere que a *string* começa na posição denominada START e tem um 20H (código ASCII para o espaço em branco) sinalizando o fim. Você não deve contar o caractere de espaço.

19. Explique o que acontece quando a instrução **mov ax,[bx]** é executada.

20. O que é um compilador?

SEÇÃO 12–5 Interrupções

21. Compare a consulta em I/O com a interrupção ativada por I/O.

22. Qual é o significado do termo vetoração?

23. Qual é o significado de interrupção por software?

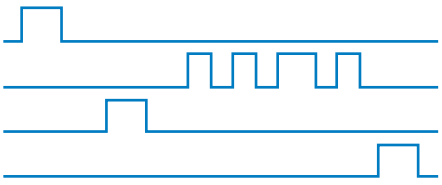
SEÇÃO 12–6 Acesso Direto à Memória (DMA)

24. Explique o que acontece na operação DMA.

25. Como a CPU é usado na operação DMA?

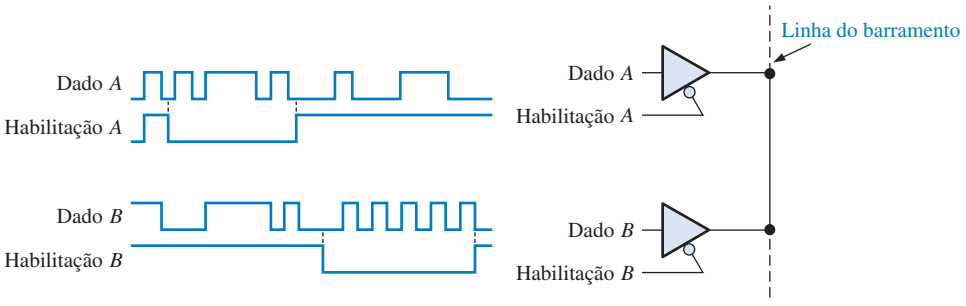
SEÇÃO 12–7 Interfaceamento Interno

26. Em uma transferência serial simples de 8 bits de dados de um dispositivo fonte para um dispositivo aceitador, a seqüência de *handshaking* na Figura 12–44 é observada nas quatro linhas de barramento genérico. Analisando a temporização, identifique a função de cada sinal e indique se ele é emitido pela fonte ou pelo aceitador.



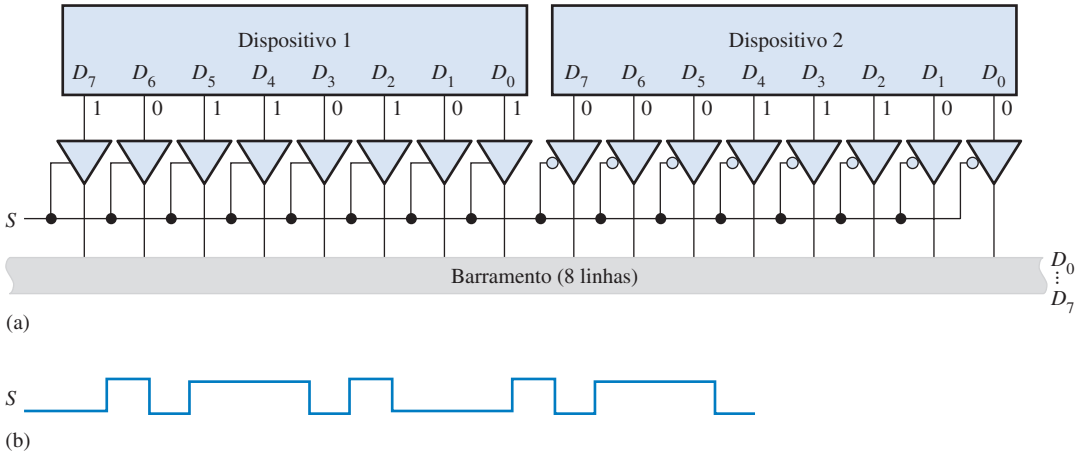
▶ FIGURA 12–44

27. Determine o sinal na linha do barramento mostrado na Figura 12–45 para as formas de onda de entrada de dados e habilitação mostradas.



▶ FIGURA 12–45

28. Na Figura 12–46(a), os dados das duas fontes são colocadas no barramento sob o controle da linha de seleção. A forma de onda de seleção é mostrada na Figura 12–46(b). Determine as formas de onda no barramento de dados para o código de saída do dispositivo indicado.



▲ **FIGURA 12–46**

SEÇÃO 12–8 Barramentos Padrão

29. Explique a diferença básica entre um barramento local e o barramento PCI.

30. Defina "*plug-and-play*".

31. Em que o barramento PCI difere do barramento ISA?

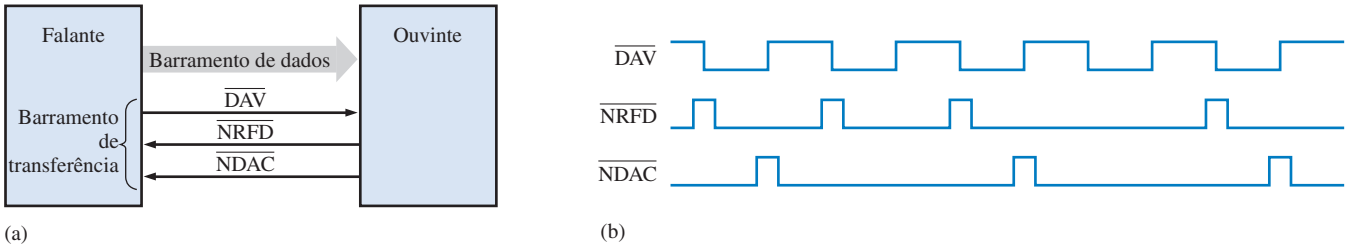
32. Se uma interface RS-232C de comprimento curto for usada, os dados podem ser transmitidos na taxa mais rápida?

33. DCE e DTE são partes de qual especificação de barramento? Explique os acrônimos DCE e DTE.

34. Cite os nomes atribuídos aos fios em um cabo USB.

35. Oito instrumentos compatíveis com GPIB são conectados ao barramento. Quantos instrumentos a mais podem ser acrescentados sem exceder à especificação?

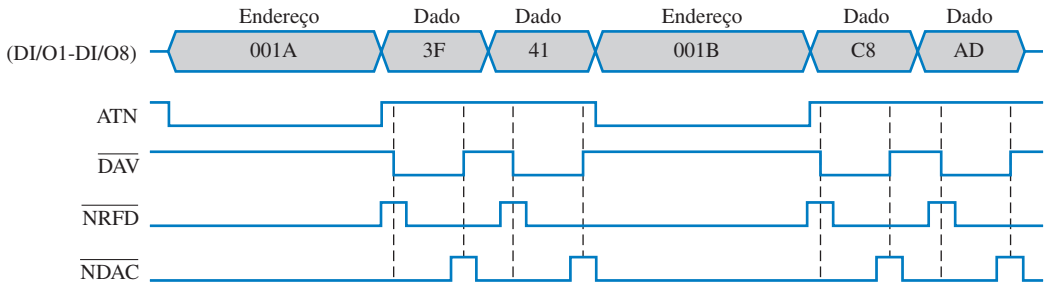
36. Considere a interface GPIB entre um "falante" e um "ouvinte" como mostra a Figura 12–47(a). A partir do diagrama de temporização do *handshaking* na parte (b) da figura, determine como vários bytes de dados são transferidos para o dispositivo "ouvinte".



▲ **FIGURA 12–47**

37. Descreva as operações ilustradas no diagrama de temporização GPIB mostrado na Figura 12–48. Desenvolva um diagrama em bloco básico do sistema envolvido nessa operação.

38. Um dispositivo "falante" envia um byte de dado para um "ouvinte" em um sistema GPIB. Simultaneamente, um DET envia um byte de dado para um DCE via interface RS-232C. Qual sistema o byte de dado completo primeiro? Por quê?



▲ **FIGURA 12–48**

RESPOSTAS

SEÇÃO DE REVISÃO

SEÇÃO 12–1 O Computador Básico

1. Os elementos básicos de um computador são a CPU, memórias, portas de entrada/saída e barramentos.

2. RAM é memória de acesso aleatório e ROM é memória apenas de leitura.

3. Periféricos são dispositivos externos ao computador.

4. Hardware é o microprocessador, memória, disco rígido, etc. Software é o programa executado pelo computador.

SEÇÃO 12–2 Microprocessadores

1. Os elementos de um microprocessador são ALU, decodificador de instruções conjunto de registradores e unidade de controle.

2. Os barramentos do microprocessador são endereço, dados e controle.

3. Um microprocessador funciona como uma CPU.

4. Operações aritméticas/lógicas, movimentação de dados e tomada de decisões.

5. *Pipelining* é o processo de execução de mais de uma instrução de cada vez.

SEÇÃO 12–3 Uma Família Específica de Microprocessador

1. Os registradores de propósito geral são

Acumulador (AX: AH, AL)	Ponteiro de pilha (SP)
Índice de base (BX: BH, BL)	Ponteiro de base (BP)
Contador (CX: CH, CL)	Índice destino (DI)
Dados (DX: DH, DL)	Índice fonte (SI)

2. A BIU provê o endereçamento e a interface de dados.

3. Não, a EU não faz interface com os barramentos.

4. A fila de instruções armazena as instruções pré-buscadas pela EU para aumentar o processamento.

5. Os códigos podem ser facilmente realocados dentro da memória.

6. O emparelhamento de instruções é o processo de combinação de instruções independentes de forma que elas possam ser executadas simultaneamente por duas unidades de execução no Pentium.

SEÇÃO 12–4 Programação de um Computador

1. Um programa é uma lista de instruções de computador organizadas de forma a alcançar um resultado específico.
2. Um código de operação é o código para uma instrução.
3. Uma *string* é uma seqüência contínua de bytes ou words.

SEÇÃO 12–5 Interrupções

1. Para uma interrupção ativada por I/O, a CPU provê um serviço para um periférico apenas quando solicitado pelo periférico; para uma consulta em I/O, a CPU verifica periodicamente o periférico para identificar a necessidade do serviço.
2. As interrupções ativadas por I/O fazem com que a CPU economize tempo.
3. Uma interrupção por software é uma instrução que invoca uma rotina de serviço de interrupção.

SEÇÃO 12–6 Acesso Direto à Memória (DMA)

1. DMA é acesso direto à memória.
2. Uma transferência DMA de dados de memória para I/O ou vice-versa economiza tempo da CPU. O acesso direto à memória é usado freqüentemente na transferência de dados entre a RAM e o *drive* de disco.

SEÇÃO 12–7 Interfaceamento Interno

1. Os buffers tristate permitem que dispositivos sejam completamente desconectados do barramento quando não estiver em uso, evitando assim interferência com outros dispositivos.
2. Um barramento interconecta todos os dispositivos de um sistema tornando a comunicação entre eles possível.

SEÇÃO 12–8 Barramentos Padrão

1. Transferência serial e paralela de dados
2. **(a)** paralela **(b)** serial **(c)** serial **(d)** paralela
3. Serial – um bit de cada vez; Paralela – 8 bits, ou mais, de cada vez
4. 127 dispositivos USB
5. FireWire é mais rápido que USB.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

12–1. $6C4C2_{16}$

12–2. Altere o primeiro bloco (bloco de inicialização) para “BIG = FFFF”; esse é o maior número não-sinalizado possível. Altere a primeira pergunta para “O número é < BIG?”

AUTOTESTE

1. (c) **2.** (b) **3.** (d) **4.** (b) **5.** (c) **6.** (b) **7.** (c) **8.** (b)
9. (a) **10.** (d) **11.** (b) **12.** (c) **13.** (c) **14.** (a) **15.** (b) **16.** (b)
17. (b) **18.** (b) **19.** (c) **20.** (d)

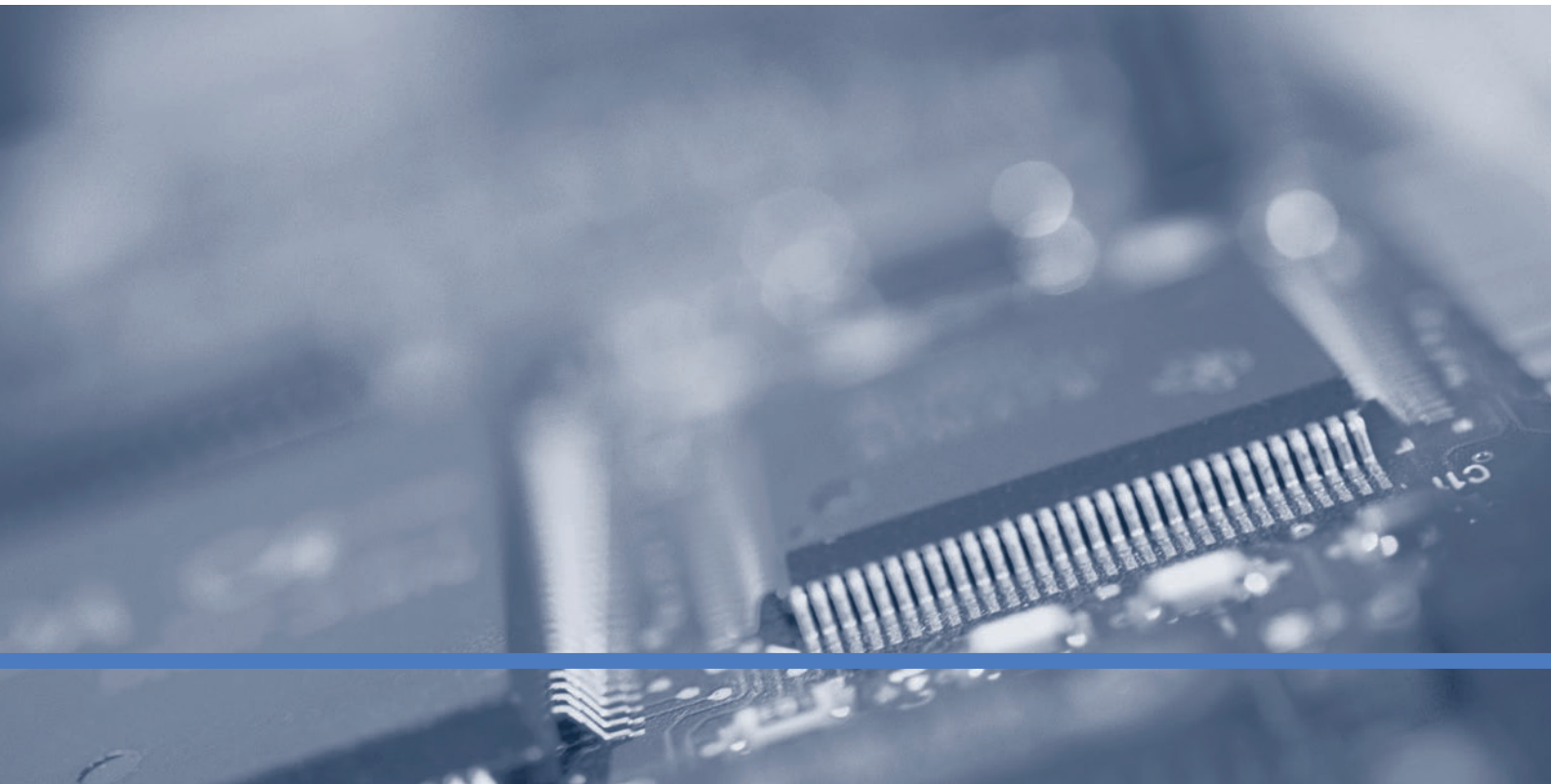
13

INTRODUÇÃO AO PROCESSAMENTO DE SINAIS DIGITAIS

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Listar os elementos essenciais em um sistema de processamento de sinais digitais
- Explicar como os sinais analógicos são convertidos para a forma digital
- Discutir a finalidade da filtragem
- Descrever o processo de amostragem
- Citar a finalidade da conversão analógico-digital



- Explicar como operam diversos tipos de ADCs
- Explicar os conceitos básicos de um processador digital de sinais (DSP)
- Descrever a arquitetura básica de um DSP
- Citar algumas das funções realizadas por um DSP
- Citar a finalidade da conversão analógico-digital
- Explicar como operam os DACs

TERMOS IMPORTANTES

- Conversor analógico-digital (ADC)
- DSP
- Conversor digital-analógico (DAC)
- Amostragem
- Freqüência de Nyquist
- *Aliasing*
- Quantização
- Núcleo de DSP
- MIPS
- MFLOPS
- MMACS
- *Pipeline*
- Busca
- Decodificação
- Execução

INTRODUÇÃO

O processamento de sinais digitais é uma tecnologia poderosa amplamente usada em muitas aplicações tais como automotiva, equipamentos de consumo, gráfica/imagem, industrial, instrumentação, médica, militar, telecomunicações e aplicações de som/voz. O processamento de sinais digitais incorpora matemática, programação (software) e hardware para processamento de sinais analógicos. Por exemplo, o processamento de sinais digitais pode ser usado para melhorar imagens, comprimir dados para transmissão e armazenamento, reconhecimento e geração de voz e redução de ruído e melhoria de áudio deteriorado.

Esse capítulo fornece uma abordagem resumida do processamento de sinais digitais. Uma abordagem completa do tópico em uma profundidade necessária para uma compreensão detalhada, necessitaria muito mais do que um único capítulo. Podemos encontrar livros completos sobre esse assunto; uma referência bibliográfica é disponibilizada no final desse capítulo. Mais informações, incluindo folhas de dados, sobre a família TMS320 de DSPs está disponível no site da Texas Instruments (www.ti.com). Informações relativas a outros DSPs podem ser encontradas no site da Motorola (www.motorola.com) e da Analog Devices (www.analogdevices.com).

DISPOSITIVOS LÓGICOS DE FUNÇÕES FIXAS

ADC0804

PROCESSADORES DE SINAIS DIGITAIS

TMS320C62xx TMS320C64xx TMS320C67xx

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

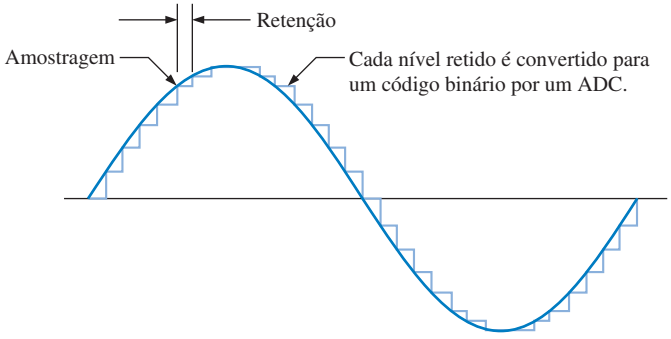
13-1 FUNDAMENTOS DE PROCESSAMENTO DE SINAIS DIGITAIS

O processamento de sinais digitais converte sinais que estejam originalmente na forma analógica tais como som, vídeo e informações de sensores, para a forma digital e usa técnicas digitais para melhorar e modificar os dados do sinal analógico em diversas aplicações.

Ao final do estudo desta seção você deverá ser capaz de:

■ Definir *ADC* ■ Definir *DSP* ■ Definir *DAC* ■ Desenhar um diagrama em bloco básico de um sistema de processamento de sinais digitais

Um sistema de processamento de sinais digitais traduz primeiro um sinal analógico que varia continuamente em uma série de níveis discretos. Essa série de níveis segue a variação do sinal analógico lembrando o formato de uma escada, conforme ilustrado para o caso de uma onda senoidal na Figura 13–1. O processo de transformação do sinal analógico original para a aproximação de uma "escada" é realizado por um circuito de amostragem e retenção (S/H – *sample and hold*).

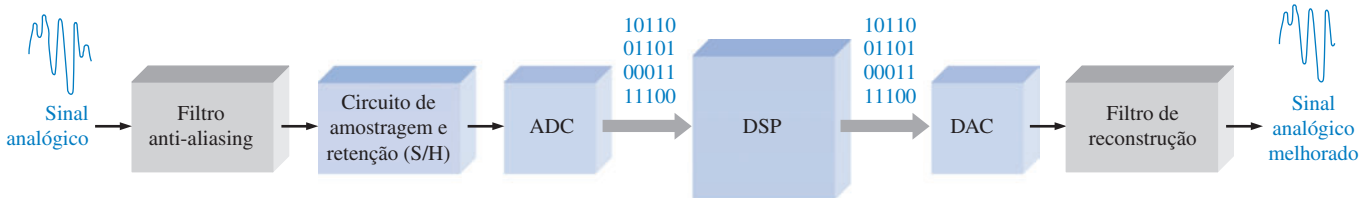


▶ **FIGURA 13–1**
Sinal analógico original (senoidal) e a sua aproximação na forma de "escada".

Em seguida, a aproximação na forma de "escada" é quantizada em códigos binários que representam cada degrau discreto da "escada" por um processo denominado conversão analógico-digital (A/D). O circuito que realiza a conversão A/D é um **conversor analógico-digital (ADC)**.

Uma vez que o sinal analógico foi convertido para o formato codificado em binário, ele é aplicado em um **DSP** (processador de sinais digitais). O DSP pode realizar diversas operações nos dados recebidos, tais como remoção de interferências indesejadas, aumento da amplitude de algumas freqüências e redução de outras, codificação de dados para transmissões seguras (sigilo) e detecção e correção de erros em códigos transmitidos. Os DSPs são capazes de, entre outras coisas, "limpar" sons ao serem gravados, remover ecos de linhas de comunicações, melhorar as imagens em equipamentos de tomografia computadorizados para um melhor diagnóstico médico e embaralhar (criptografar) a conversação telefônica celular para fins de privacidade.

Após o DSP processar um sinal, este pode ser convertido de volta para o formato analógico original numa versão bastante melhorada. Isso é realizado por um **conversor digital-analógico (DAC)**. A Figura 13–2 mostra um diagrama em bloco básico de um sistema de processamento de sinais digitais.



▲ **FIGURA 13–2**
Diagrama em bloco básico de um sistema de processamento de sinais digitais.

DSPs são na realidade um tipo especializado de microprocessador, porém são diferentes dos processadores de propósito geral em dois aspectos importantes. Tipicamente, os microprocessadores são projetados para funções de propósitos gerais e operam com vários pacotes de software. DSPs são usados em aplicações de propósitos específicos; eles são processadores numéricos muito rápidos que têm que trabalhar em tempo real processando informações à medida que são geradas usando algoritmos especializados (programas). O conversor analógico-digital (ADC) em um sistema tem que receber as amostras do sinal analógico de entrada numa freqüência suficiente para capturar todas as flutuações relevantes na amplitude do sinal e o DSP tem que manter o sincronismo com a taxa de amostragem do ADC fazendo os cálculos tão rápido quanto as amostras são recebidas. Uma vez que os dados digitais sejam processados pelo DSP, eles vão para um conversor digital-analógico (DAC) para converter de vota para o formato analógico.

SEÇÃO 13–1 REVISÃO

1. O que significa DSP?
2. O que significa ADC?
3. O que significa DAC?
4. Um sinal analógico é transformado para a forma codificada em binário por qual circuito?
5. Um sinal codificado em binário é transformado para a forma analógica por qual circuito?

13-2 CONVERSÃO DE SINAL ANALÓGICO PARA DIGITAL

Para processar sinais usando técnicas digitais, o sinal analógico de entrada tem que ser convertido para o formato digital.

Ao final do estudo desta seção você deverá ser capaz de:

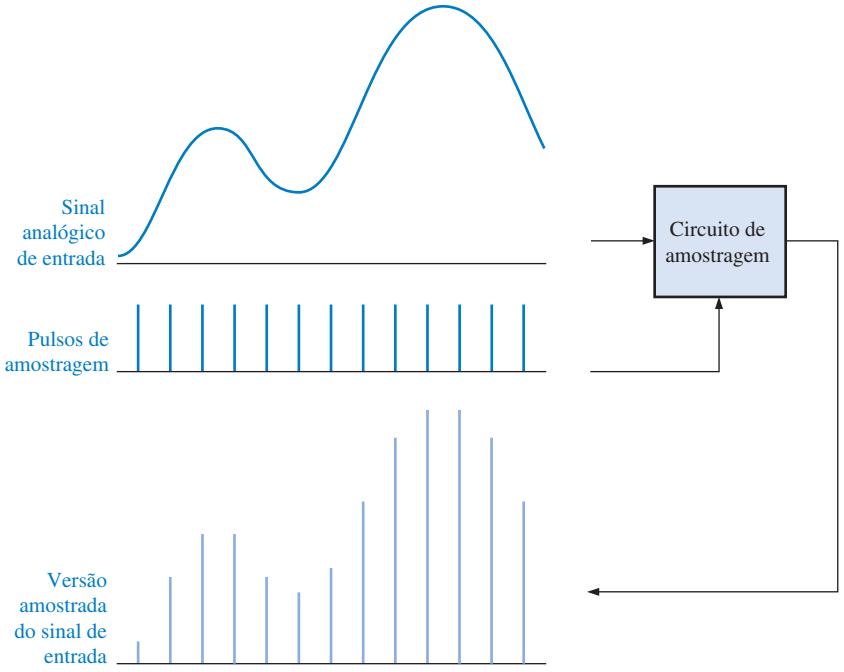
■ Explicar o processo básico de conversão de um sinal analógico para digital ■ Descrever a finalidade da função de amostragem e retenção ■ Definir a freqüência de Nyquist ■ Definir a razão do *aliasing* e discutir como ele pode ser eliminado ■ Descrever a finalidade de um ADC

Amostragem e Filtragem

Os dois primeiros blocos no diagrama do sistema mostrado na Figura 13–2 são o filtro *anti-aliasing* e o circuito de amostragem e retenção. A função de amostragem e retenção faz duas operações sendo que a primeira é a amostragem. A **amostragem** é o processo de aquisição de um número suficiente de valores discretos em pontos da forma de onda que definem o formato da onda. Quanto mais amostras são obtidas, com mais precisão a forma de onda pode ser definida. A amostragem converte um sinal analógico em uma série de impulsos, cada um representando a amplitude do sinal num dado instante do tempo. A Figura 13–3 ilustra o processo de amostragem.

Quando um sinal analógico tem que ser amostrado, existem certos critérios que têm que ser observados para que o sinal original seja representado com a precisão necessária. Todos os sinais analógicos (exceto uma onda senoidal pura) contêm um espectro de componentes freqüências denominadas de *harmônicas*. As harmônicas de um sinal analógico são ondas senoidais de diferentes freqüências e amplitudes. Quando as harmônicas de uma dada forma de onda periódica são somadas, o resultado é o sinal original. Antes de o sinal ser amostrado, ele tem que passar por um filtro passa-baixas (filtro *anti-aliasing*) para eliminar freqüências harmônicas acima de um certo valor conforme determinado pela freqüência de Nyquist.

O Teorema da Amostragem Observe na Figura 13–3 que existem duas formas de onda de entrada. Uma delas é o sinal analógico e a outra é a forma de onda dos pulsos de amostragem. O teorema da amostragem diz que, para representar um sinal analógico, a freqüência de amostragem, $f_{(\text{amostragem})}$, tem que ser pelo menos duas vezes a componente de maior freqüência, $f_{a(\text{máx})}$, do sinal



► **FIGURA 13–3**
Ilustração do processo de amostragem.

analógico. Uma outra forma de dizer isso é que a maior freqüência analógica não pode ser maior que metade da freqüência de amostragem. A freqüência é conhecida como **freqüência de Nyquist** e é expressa pela Equação 13–1. Na prática, a freqüência de amostragem deve ser maior que duas vezes a maior freqüência analógica.

Equação 13–1

$$f_{\text{amostragem}} \geq 2f_{\text{a(máx)}}$$

Para entender intuitivamente o teorema da amostragem, uma simples analogia com uma "bola saltando" pode ser útil. Embora não seja o caso de uma representação perfeita da amostragem de sinais elétricos, ela serve para ilustrar a idéia básica. Se uma bola for fotografada (amostrada) em um instante durante um único salto, como ilustra a Figura 13–4(a), não podemos dizer nada a respeito da trajetória da bola exceto que ela está acima do piso. Não podemos dizer se ela está subindo ou descendo ou ainda a distância do salto. Se tomarmos duas fotos em instantes espaçados igualmente durante um salto, conforme mostra a parte (b) da figura, podemos obter um mínimo de informação sobre seu movimento e nada sobre a distância do seu salto. Neste caso em particular, sabemos apenas que a bola estava no ar no momento em que as duas fotos foram tiradas e que a altura máxima do salto é pelo menos igual a altura mostrada em cada foto. Se tirarmos quatro fotos, como mostra a parte (c) da figura, então a trajetória da bola durante um salto começa a surgir. Quanto mais fotos (amostras) tirarmos, com mais precisão poderemos determinar a trajetória da bola em seus saltos.

(a) Uma amostra do movimento de uma bola

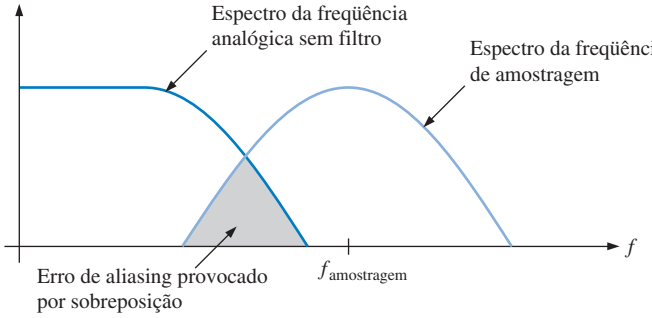
(b) Duas amostras do movimento de uma bola durante um único salto.

(c) Quatro amostras do movimento de uma bola durante um único salto formam um esboço da trajetória da bola.

▲ **FIGURA 13–4**
Analogia do teorema da amostragem com uma bola saltando.

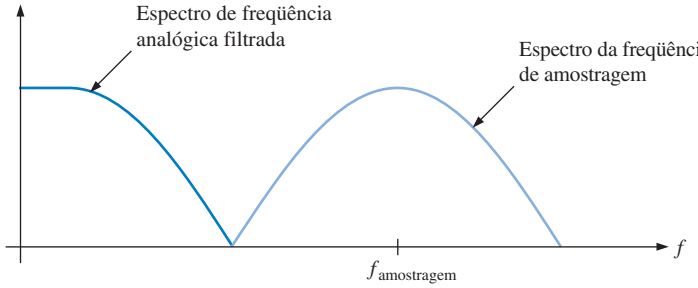
A Necessidade da Filtragem A filtragem por um filtro passa-baixas é necessária para remover todas as componentes de freqüência (harmônicas) do sinal analógico que excedem à freqüência de Nyquist. Se existirem quaisquer componentes de freqüência no sinal analógico que excedam à freqüência de Nyquist, uma condição indesejada conhecida como ***aliasing*** (falseamento ou sobreposição) ocorrerá. Um *alias* é um sinal produzido quando a freqüência de amostragem não é pelo menos duas vezes a freqüência do sinal. Um sinal *alias* tem uma freqüência que é menor que a maior freqüência no sinal analógico amostrado e, portanto, esta situado dentro do espectro ou banda de freqüência do sinal analógico de entrada provocando distorção. Tal sinal "aparenta" ser parte do sinal analógico quando na realidade não é. Por isso o termo *alias* (falso).

Uma outra forma de ver o *aliasing* é considerando que os pulsos de amostragem produzem um espectro de freqüências harmônicas acima e abaixo da freqüência de amostragem, como mostra a Figura 13–5. Se um sinal analógico contém freqüências acima da freqüência de Nyquist, essas freqüências se sobrepõem ao espectro da forma de onda amostrada conforme visto na figura, ocorrendo uma interferência. As componentes de freqüência menores da forma de onda de amostragem se misturam com o espectro de freqüência do sinal analógico, resultando em um erro de *aliasing* (falseamento ou sobreposição).



◀ **FIGURA 13–5**
Uma ilustração básica da condição $f_{amostragem} < 2f_{a(máx)}$.

Um filtro passa-baixas *anti-aliasing* tem que ser usado para limitar o espectro de freqüência do sinal analógico para uma dada freqüência de amostragem. Para evitar um erro de *aliasing*, o filtro tem que pelo menos eliminar todas as freqüências analógicas acima da freqüência mínima no espectro de amostragem, conforme ilustrado na Figura 13–6. O *aliasing* também pode ser evitado aumentando suficientemente a freqüência de amostragem. Entretanto, a freqüência de amostragem máxima é geralmente limitada pelo desempenho do conversor analógico-digital (ADC) que vem em seguida ao filtro.



◀ **FIGURA 13–6**
Após o filtro passa-baixas, o espectro de freqüência dos sinais analógico e de amostragem não se sobrepõem, eliminando assim o erro de *aliasing*.

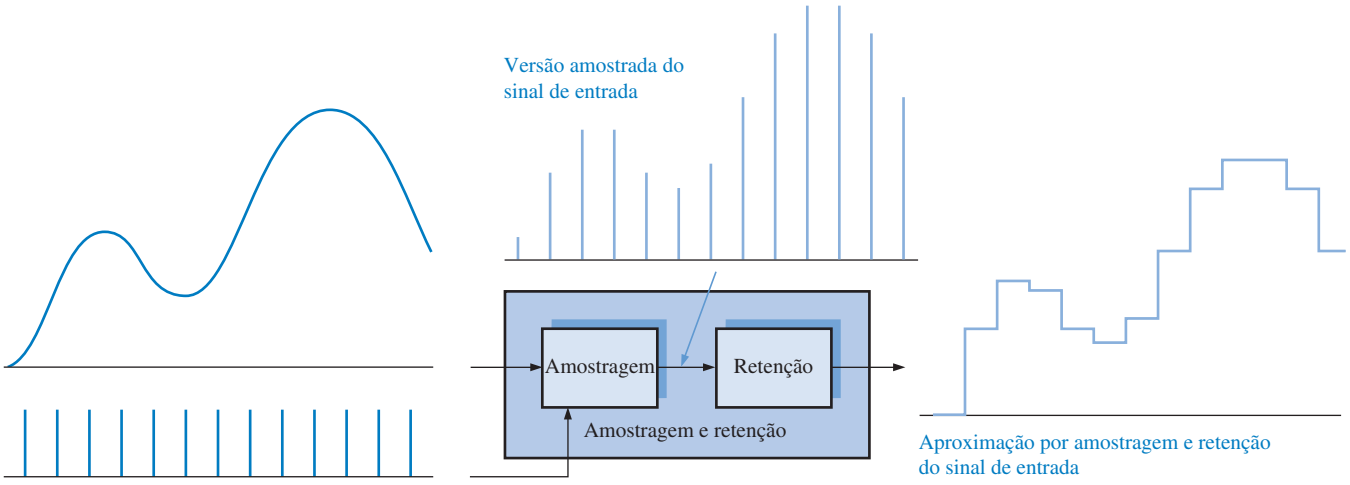
Uma Aplicação Um exemplo da aplicação da amostragem está nos equipamentos de áudio digital. As taxas de amostragens usadas são 32 kHz, 44,1 kHz ou 48 kHz (o número de amostras por segundo). A taxa de 48 kHz é a mais comum, mas a taxa de 44,1 kHz é usada para CDs de áudio e fitas pré-gravadas. De acordo com a taxa de Nyquist, a freqüência de amostragem tem que ser pelo menos duas vezes a freqüência do sinal de áudio. Portanto, a taxa de amostragem de 44,1 kHz para um CD captura as freqüências de até aproximadamente 22 kHz, o que excede à especificação de 20 kHz que é comum para a maioria dos equipamentos de áudio.

Muitas aplicações não necessitam de uma faixa de freqüência ampla para obter uma reprodução aceitável do som. Por exemplo, a voz humana contém algumas freqüências próximas de 10 kHz

e, portanto, requer uma taxa de amostragem de pelo menos 20 kHz. Entretanto, se apenas as freqüências até 4 kHz (necessitando idealmente de uma taxa de amostragem mínima de 8 kHz) forem reproduzidas, a voz fica bastante inteligível. Por outro lado, se um sinal sonoro não for amostrado numa taxa alta o suficiente, o efeito do *aliasing* será notado como ruído de fundo e distorção.

Retenção do Valor Amostrado

A operação de retenção é parte do bloco de amostragem e retenção mostrado na Figura 13–2. Após a filtragem e amostragem, o nível amostrado tem que ser mantido constante até que aconteça a próxima amostra. Isso é necessário para o ADC ter tempo de processar o valor amostrado. Essa operação de amostragem e retenção resulta numa forma de onda no formato de uma "escada" a qual se aproxima da forma de onda analógica de entrada, conforme mostra a Figura 13–7.

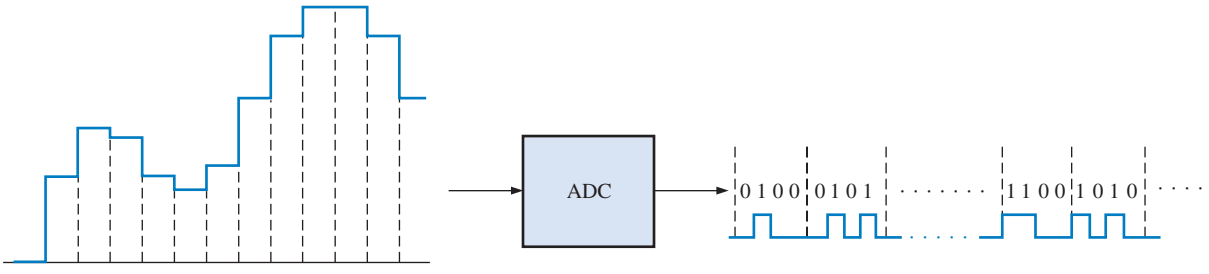


▲ **FIGURA 13–7**

Ilustração de uma operação de amostragem e retenção.

Conversão Analógico-Digital

A conversão analógico-digital é o processo de conversão da saída do circuito de amostragem e retenção em uma série de códigos binários que representam a amplitude do sinal de entrada analógico em cada instante amostrado. O processo de amostragem e retenção mantém a amplitude do sinal de entrada analógico constante entre os pulsos de amostragem; portanto, a conversão analógico-digital pode ser feita usando um valor constante em vez de um sinal analógico que varia durante o intervalo de conversão, o qual corresponde ao tempo entre os pulsos de amostragem. A Figura 13–8 ilustra a função básica de um conversor analógico-digital (ADC). Os intervalos de amostragem são indicados por linhas tracejadas.

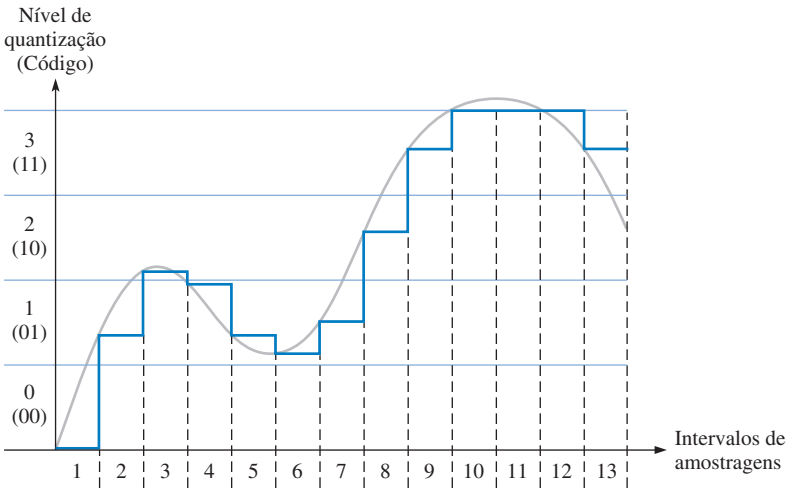


▲ **FIGURA 13–8**

Função básica de um conversor analógico-digital (ADC) (os códigos binários e o número de bits foram escolhidos arbitrariamente apenas por questão de ilustração). A forma de onda de saída do ADC que representa os códigos binários também é mostrada.

Quantização O processo de conversão de um valor analógico para um código é denominado de **quantização**. Durante o processo de quantização, o ADC converte cada valor amostrado do sinal analógico em código binário. Quanto mais bits forem usados para representar um valor amostrado, mais precisa será a representação.

Para ilustrar, vamos quantizar uma produção da forma de onda analógica em quatro níveis (0–3). Conforme mostra a Figura 13–9, são necessários dois bits. Observe que cada nível de quantização é representado por um código de 2 bits no eixo vertical e cada intervalo de amostragem é numerado ao longo do eixo horizontal. O processo de quantização é resumido na Tabela 13–1.



◀ **FIGURA 13–9**

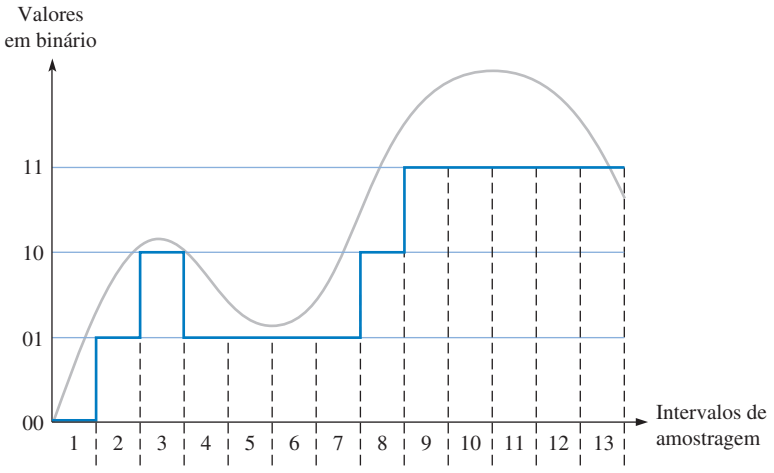
Forma de onda de saída do circuito de amostragem e retenção com quatro níveis de quantização. A forma de onda analógica original é mostrada na cor cinza para referência.

◀ **TABELA 13–1**

Quantização de dois bits para a forma de onda vista na Figura 13–9

INTERVALO DE AMOSTRAGEM	NÍVEL DE QUANTIZAÇÃO	CÓDIGO
1	0	00
2	1	01
3	2	10
4	1	01
5	1	01
6	1	01
7	1	01
8	2	10
9	3	11
10	3	11
11	3	11
12	3	11
13	3	11

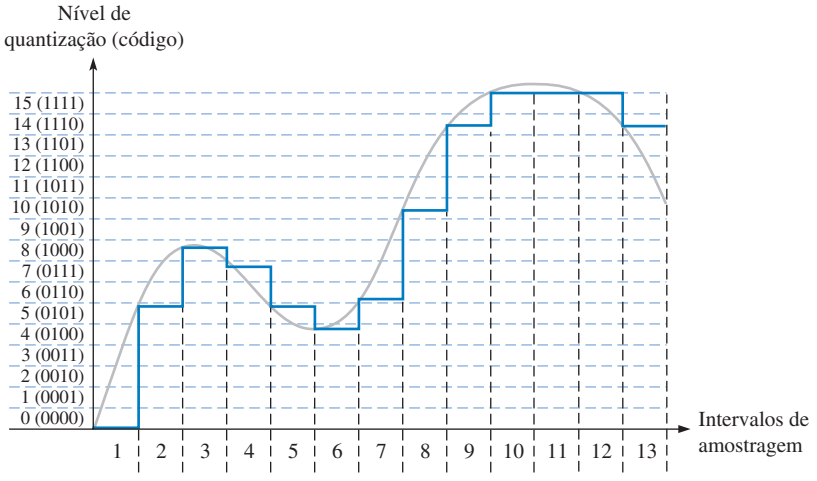
Se os códigos digitais de 2 bits resultantes forem usados para reconstruir a forma de onda original, o que é feito pelos conversores digital-analógico (DACs), obteremos a forma de onda mostrada na Figura 13–10. Conforme podemos ver, é perdido totalmente um bit de precisão usando apenas dois bits para representar os valores amostrados.



► FIGURA 13–10

A forma de onda reconstruída na Figura 13–9 usando quatro níveis de quantização (2 bits). A forma de onda analógica original é mostrada na cor cinza para referência.

Agora vamos ver como mais bits melhoram a precisão. A Figura 13–11 mostra a mesma forma de onda com dezesseis níveis (4 bits) de quantização. O processo de quantização de 4 bits é resumido na Tabela 13–2.



► FIGURA 13–11

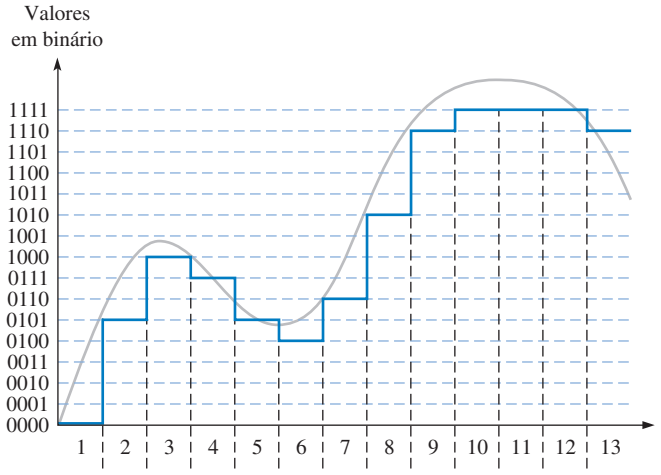
Forma de onda de saída do circuito de amostragem e retenção com dezesseis níveis de quantização. A forma de onda analógica original é mostrada na cor cinza para referência.

► TABELA 13–2

Quantização de 4 bits para a forma de onda mostrada na Figura 13–11

INTERVALO DE AMOSTRAGEM	NÍVEL DE QUANTIZAÇÃO	CÓDIGO
1	0	0000
2	5	0101
3	8	1000
4	7	0111
5	5	0101
6	4	0100
7	6	0110
8	10	1010
9	14	1110
10	15	1111
11	15	1111
12	15	1111
13	14	1110

Se os códigos digitais de 4 bits resultantes forem usados para reconstruir a forma de onda original, obteremos a forma de onda mostrada na Figura 13–12. Conforme podemos ver, o resultado é muito mais parecido com a forma de onda original do que no caso em que os nível de quantização eram quatro (Figura 13–10). Isso mostra que uma precisão maior é alcançada com mais bits de quantização. A maioria dos circuitos integrados ADCs usam de 8 a 24 bits, sendo que a função de amostragem e retenção está algumas vezes contida no chip ADC. Vários tipos de ADCs são apresentados na próxima seção.



◀ **FIGURA 13–12**
Forma de onda reconstruída a partir da Figura 13–11 usando dezesseis níveis de quantização (4 bits). A forma de onda analógica original é mostrada na cor cinza para referência.

SEÇÃO 13–2 REVISÃO

1. O que significa a amostragem?
2. Porque devemos fazer a retenção de um valor amostrado?
3. Se a componente de maior freqüência em um sinal analógico é 20 kHz, qual é a mínima freqüência de amostragem?
4. O que significa a quantização?
5. O que determina a precisão no processo de quantização?

13-3 MÉTODOS DE CONVERSÃO ANALÓGICO-DIGITAL

Conforme já estudamos, a conversão analógico-digital é o processo pelo qual uma grandeza analógica é convertida para a forma digital. Quando se mede grandezas, é necessário que estas estejam na forma digital para processamento, apresentação ou armazenamento. Alguns tipos comuns de conversores analógico-digitais (ADCs) são agora examinados. Dois parâmetros importantes dos ADCs são a *resolução*, que é o número de bits, e a *capacidade de processamento*, que é a taxa de amostragem que um ADC pode operar medida em amostras por segundo (sps – *samples per second*).

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar basicamente o que é um amplificador operacional ■ Mostrar como o amplificador operacional pode ser usado como um amplificador inversor ou um comparador ■ Explicar como um ADC flash funciona ■ Discutir os ADCs de dupla rampa ■ Descrever a operação de um ADC de aproximação sucessiva ■ Descrever um ADC delta-sigma ■ Discutir o teste de ADCs para um código ausente, um código incorreto e um *offset*

Estudo Resumido de um Amplificador Operacional

Antes de iniciar o estudo dos conversores analógico-digitais (ADCs), vamos fazer um estudo resumido de um elemento de circuito que é comum à maioria dos ADCs e DACs. Esse elemento é o amplificador operacional ou, na forma reduzida, é denominado de amp-op.

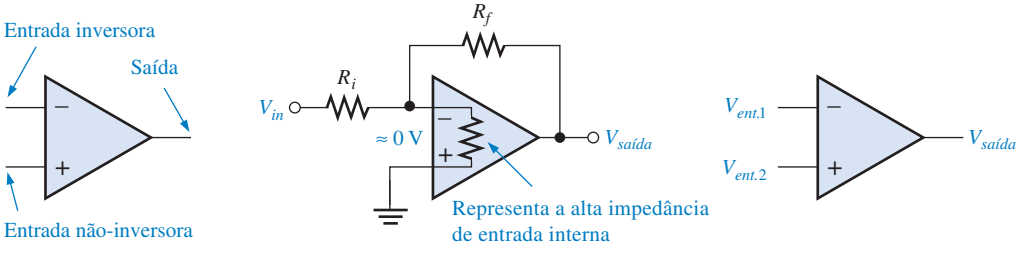
Um **amp-op** é um amplificador linear que tem duas entradas (inversora e não-inversora) e uma saída. Ele tem um ganho de tensão muito alto e uma impedância de entrada também alta, bem como uma impedância de saída muito baixa. O símbolo do amp-op é mostrado na Figura 13–13(a). Quando usado como um amplificador inversor, o amp-op é configurado como mostra a parte (b) da figura. O resistor de realimentação, R_f, e o resistor de entrada, R_i, controlam o ganho de tensão de acordo com a fórmula dada pela Equação 13–2, onde $V_{saída}/V_{ent.}$ é o ganho de tensão em malha fechada (malha fechada se refere à realimentação da saída para a entrada proporcionada por R_f). O sinal negativo indica inversão.

Equação 13–2

$$\frac{V_{saída}}{V_{ent.}} = -\frac{R_f}{R_i}$$

Na configuração de amplificador inversor, a entrada inversora do amp-op está aproximadamente no potencial GND (0 V) por causa da realimentação e do ganho de tensão em malha aberta ser extremamente alto, o que torna a tensão diferencial entre as duas entradas extremamente baixa. Como a entrada não-inversora está aterrada, a entrada inversora está aproximadamente no potencial 0 V, que é denominado de *terra virtual*.

Quando o amp-op é usado como um comparador, como mostra a Figura 13–13(c), duas tensões são aplicadas nas entradas. Quando essas tensões de entrada apresentam uma diferença muito pequena, a saída do amp-op é levada para um dos seus dois estados de saturação (nível ALTO ou BAIXO), dependendo de qual tensão de entrada é maior.



(a) Símbolo do amp-op (b) Amp-op como um amplificador inversor com ganho de R_f/R_i (c) Amp-op como um comparador

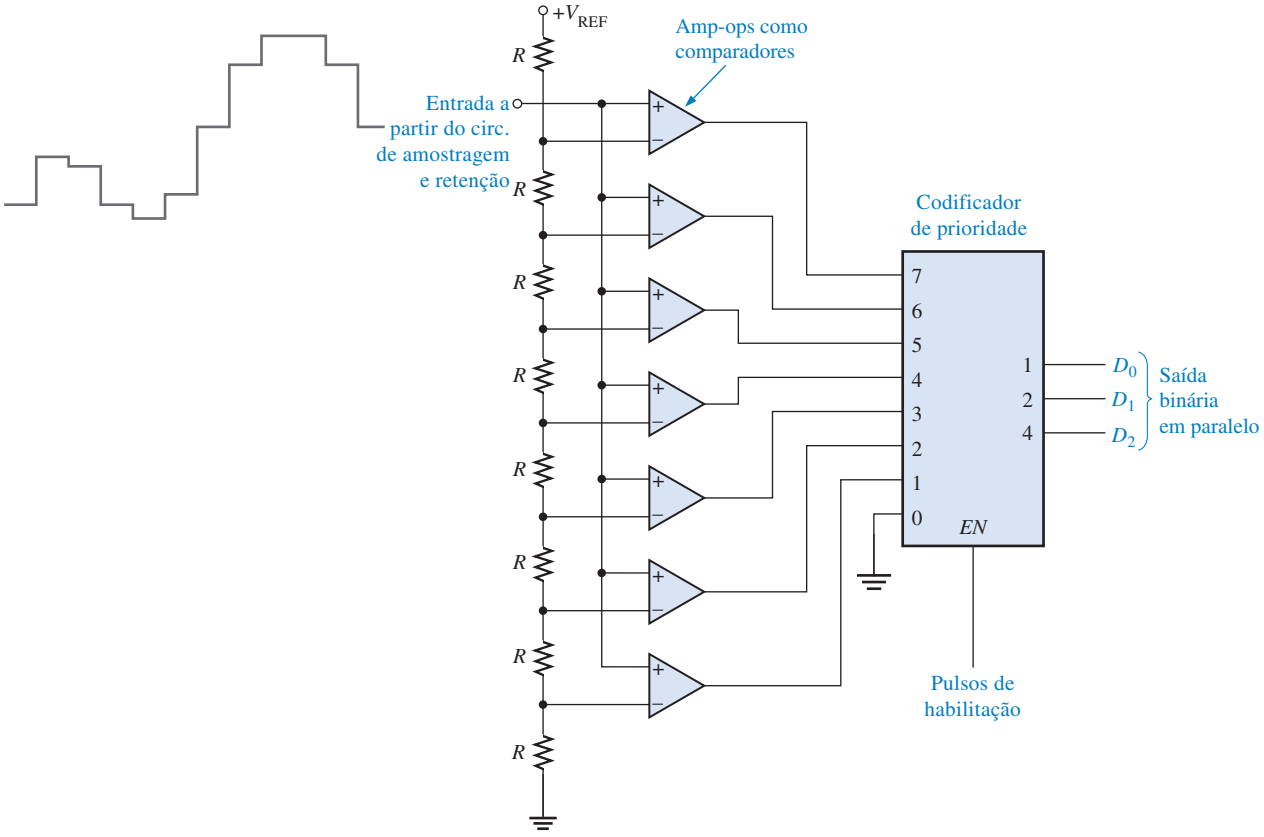
▲ **FIGURA 13–13**

O amplificador operacional (amp-op).

Conversor Analógico-Digital Flash (Simultâneo)

O método flash utiliza comparadores que comparam tensões de referência com a tensão de entrada analógica. Quando a tensão de entrada excede a tensão de referência para um dado comparador, um nível ALTO é gerado na saída. A Figura 13–14 mostra um conversor de 3 bits que usa sete circuitos comparadores; não é necessário um comparador para a condição em que todos os bits são 0. Um conversor desse tipo de 4 bits necessita de quinze comparadores. Em geral, são necessários $2^n - 1$ comparadores para a conversão de n bits em código binário. O número de bits usados em um ADC é a sua **resolução**. O grande número de comparadores necessário para um número binário de tamanho razoável é uma das desvantagens do **ADC flash**. A principal vantagem desse conversor é a rapidez da conversão por causa da sua alta *capacidade de processamento*, medido em amostras por segundo (sps).

A tensão de referência para cada comparador é definida por um circuito divisor de tensão. A saída de cada comparador é conectada à entrada de um codificador de prioridade. O codificador é habilitado por um pulso na entrada EN e um código de 3 bits representa o valor da entrada que aparece na saída do codificador. O código binário é determinado pela entrada de ordem mais alta que estiver em nível ALTO.



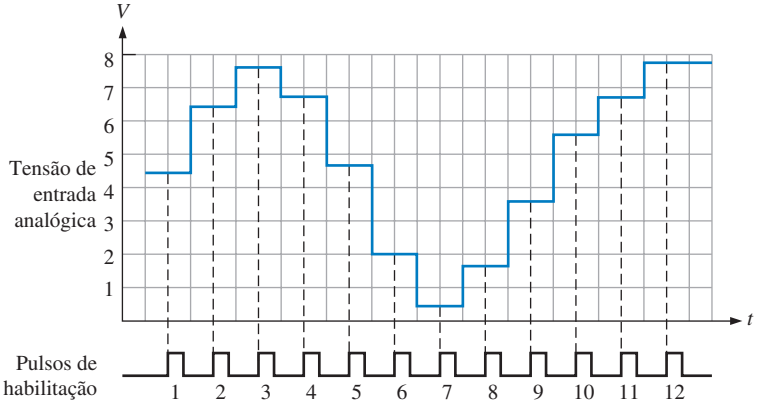
▲ **FIGURA 13–14**

ADC flash de 3 bits.

A freqüência dos pulsos de habilitação e o número de bits no código binário determinam a precisão com a qual a seqüência de códigos binários representam a entrada do ADC. Deve haver um pulso de habilitação para cada nível amostrado do sinal de entrada.

EXEMPLO 13–1

Determine o código binário de saída do ADC flash visto na Figura 13–14 para o sinal de entrada mostrado na Figura 13–15 juntamente com os pulsos de habilitação do codificador. Para esse exemplo, V_{REF} = +8 V.

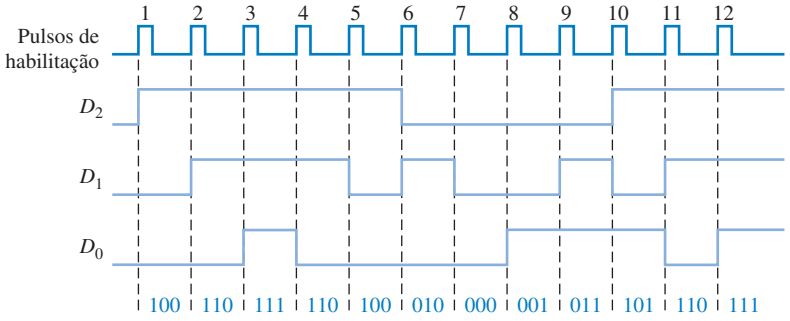


▶ **FIGURA 13–15**

Amostragem de valores de uma forma de onda para a conversão para código binário.

Solução A seqüência de saída digital resultante é mostrada a seguir e é mostrada também como forma de onda num diagrama na Figura 13–16 em relação aos pulsos de entrada:

100, 110, 111, 110, 100, 010, 000, 001, 011, 101, 110, 111



▲ **FIGURA 13–16**

Saídas digitais resultantes relativa aos valores de saída do circuito de amostragem e retenção. A saída D_0 é o LSB do código binário de 3 bits.

Problema relacionado* Se a freqüência dos pulsos de habilitação na Figura 13–15 for reduzida à metade, determine os números binários representados pela seqüência de saída digital resultante para 6 pulsos. Alguma informação é perdida?

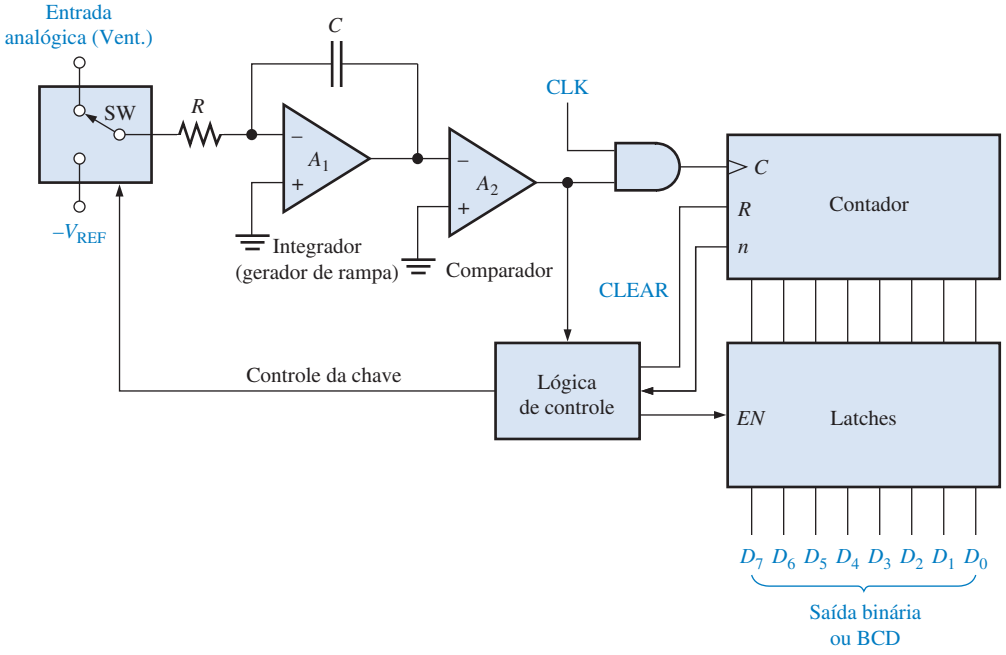
* As respostas estão no final do capítulo.

Conversor Analógico-Digital de Dupla Rampa

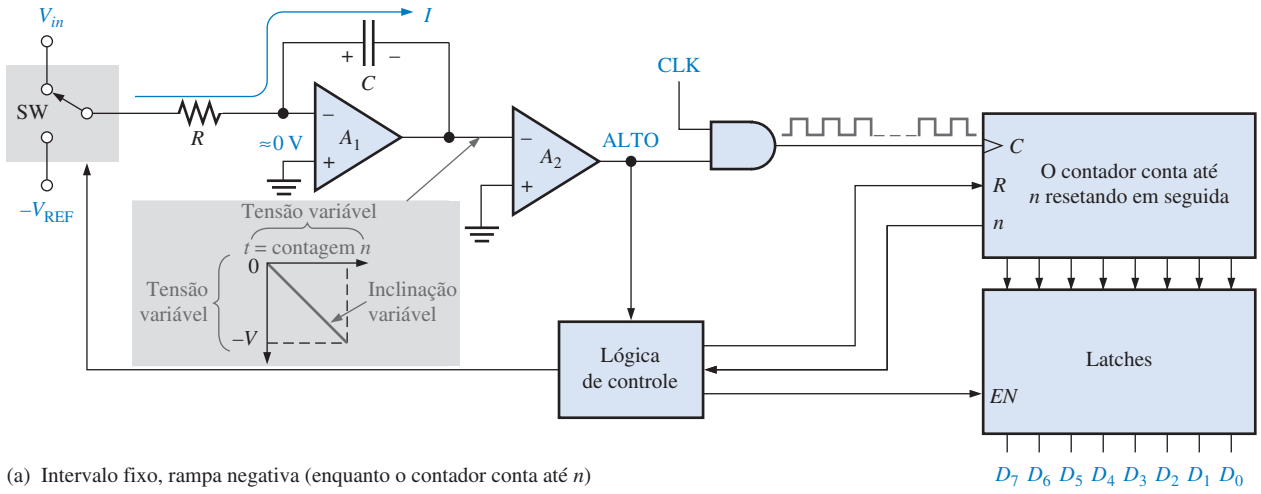
Um ADC de dupla rampa é comum em voltímetros digitais e outros tipos de instrumentos de medição. Um gerador de rampa (integrador) é usado para produzir a característica de dupla rampa. Um diagrama em bloco de um ADC de dupla rampa é mostrado na Figura 13–17.

▶ **FIGURA 13–17**

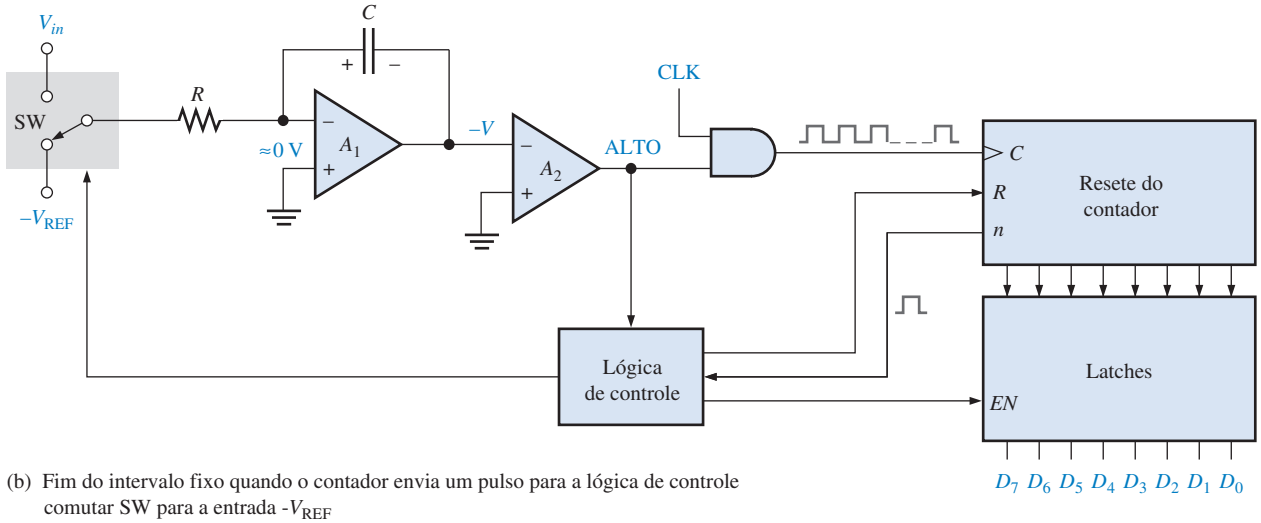
ADC de dupla rampa básico.



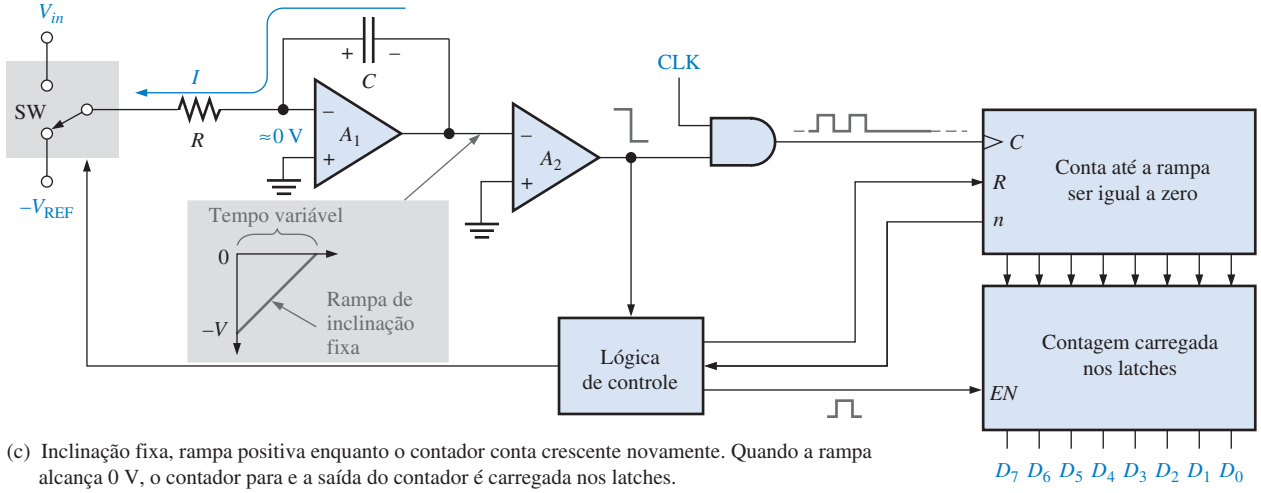
A Figura 13–18 ilustra a conversão de dupla rampa. Comece considerando que o contador é resetado e que a saída do integrador seja zero. Agora considere que uma tensão de entrada positiva seja aplicada na entrada através da chave (SW) conforme selecionada pela lógica de con-



(a) Intervalo fixo, rampa negativa (enquanto o contador conta até n)



(b) Fim do intervalo fixo quando o contador envia um pulso para a lógica de controle comutar SW para a entrada -V_{REF}



(c) Inclinação fixa, rampa positiva enquanto o contador conta crescente novamente. Quando a rampa alcança 0 V, o contador para e a saída do contador é carregada nos latches.

▲ FIGURA 13–18

Ilustração da conversão de dupla rampa.

trole. Como a entrada inversora de A_1 está no terra virtual e considerando que $V_{ent.}$ é constante por um período de tempo, a corrente através do resistor de entrada R será constante bem como no capacitor C. Esse capacitor se carrega linearmente em função da corrente ser constante e, como resultado, a rampa de tensão linear será negativa na saída de A_1, conforme ilustra a Figura 13–18(a).

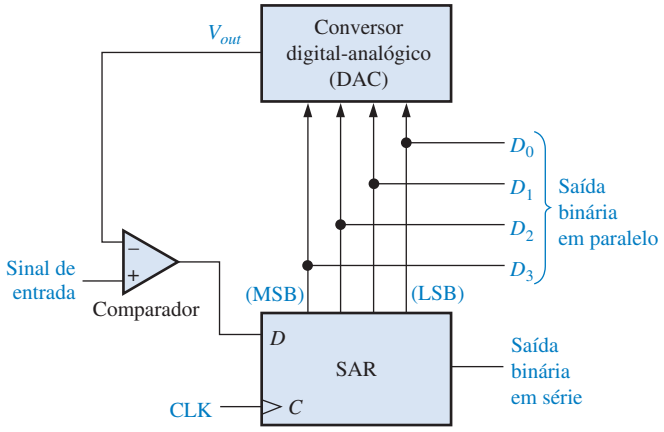
Quando o contador alcança uma contagem específica, ele é resetado e a lógica de controle comuta a chave para a tensão de referência negativa ($-V_{REF}$) para a entrada de A_1, conforme mostra a Figura 13–18(b). Nesse ponto o capacitor está carregado com uma tensão negativa ($-V$) proporcional à tensão analógica de entrada.

Agora o capacitor se descarrega linearmente, porque a corrente é constante, a partir de $-V_{REF}$, como mostra a Figura 13–18(c). Essa descarga linear produz uma rampa positiva na saída de A_1, começando em $-V$ e tendo uma inclinação constante que é independente da tensão de carga. À medida que o capacitor se descarrega, o contador avança a partir do estado de resete. O tempo que o capacitor leva para se descarregar até zero depende da tensão inicial $-V$ (proporcional à $V_{ent.}$) porque a taxa de descarga (inclinação) é constante. Quando a tensão de saída do integrador (A_1) chega em zero, o comparador (A_2) comuta para o estado BAIXO e desabilita o clock para o contador. A contagem binária é armazenada nos latches, completando assim um ciclo de conversão. A contagem binária é proporcional a $V_{ent.}$ porque o tempo que o capacitor leva para se descarregar depende apenas de $-V$ e o contador guarda esse intervalo de tempo.

Conversor Analógico-Digital de Aproximação Sucessiva

Um dos métodos mais usados de conversão analógico-digital é a aproximação sucessiva. Esse método apresenta uma conversão muito mais rápida que a conversão de dupla rampa, porem é mais lento que o método flash. Ele também apresenta um tempo de conversão fixo que é o mesmo para qualquer valor de tensão analógica de entrada.

A Figura 13–19 mostra um diagrama em bloco básico de um ADC de aproximação sucessiva de 4 bits. Ele consiste de um DAC (DACs são abordados na Seção 13–5), um registrador de aproximação sucessiva (SAR) e um comparador. A operação básica é a seguinte: Os bits de entrada do DAC são habilitados (feitos iguais a 1) um de cada vez, começando com o bit mais significativo (MSB). Conforme cada bit é habilitado, o comparador produz uma saída que indica se a tensão do sinal de entrada é maior ou menor que a saída do DAC. Se a saída do DAC for maior que o sinal de entrada, a saída do comparador será nível BAIXO, fazendo com que o bit no registrador seja resetado. Se a saída for menor que o sinal de entrada, o bit 1 é mantido no registrador. O sistema faz isso primeiro com o MSB, e em seguida com o próximo bit mais significativo, depois o próxi-

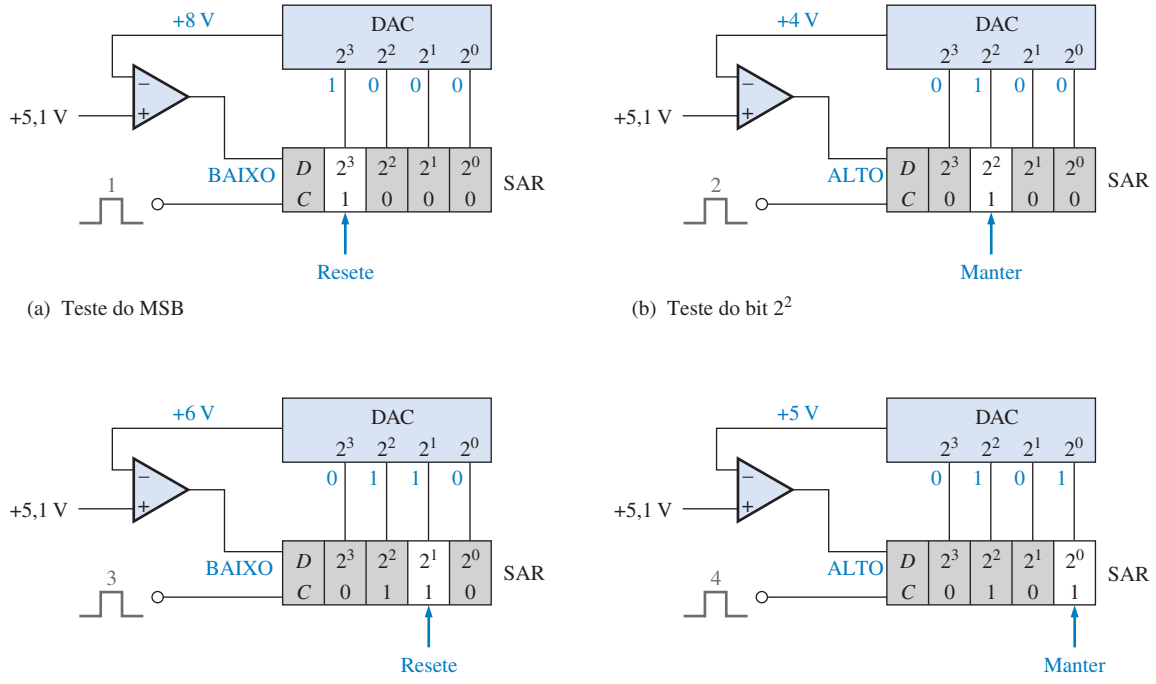


▲ **FIGURA 13–19**

ADC de aproximação sucessiva.

mo e assim por diante. Após todos os bits do DAC serem testados, o ciclo de conversão estará completo.

Para entender melhor a operação do ADC de aproximação sucessiva, vamos usar um exemplo específico de uma conversão de 4 bits. A Figura 13–20 ilustra a conversão passo a passo de uma tensão de entrada constante (5,1 V nesse caso). Vamos considerar que o DAC tem a seguinte característica de saída: $V_{saída}$ = 8 V para o bit 2^3 (MSB), $V_{saída}$ = 4 V para o bit 2^2, $V_{saída}$ = 2 V para o bit 2^1 e $V_{saída}$ = 1 V para o bit 2^0 (LSB).



▲ FIGURA 13–20

Ilustração de um processo de conversão com aproximação sucessiva.

A Figura 13–20(a) mostra o primeiro passo no ciclo de conversão com o MSB = 1. A saída do DAC é 8 V. Como esse valor é maior que a entrada de 5,1 V, a saída do comparador será nível BAIXO, fazendo com que o MSB no SAR seja resetado para 0.

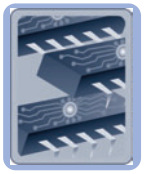
A Figura 13–20(b) mostra o segundo passo no ciclo de conversão com o bit 2^2 igual a 1. A saída do DAC é 4 V. Como esse valor é menor que a entrada de 5,1 V, a saída do comparador comuta para o nível ALTO fazendo com que o bit seja mantido no SAR.

A Figura 13–20(c) mostra o terceiro passo no ciclo de conversão com o bit 2^1 igual a 1. A saída do DAC é 6 V porque existe um nível 1 no bit 2^2 e um nível 1 no bit 2^1; 4 V + 2 V = 6 V. Como esse valor é maior que a entrada de 5,1 V, a saída do comparador comuta para nível BAIXO, fazendo com que o bit seja resetado para 0.

A Figura 13–20(d) mostra o quarto passo no ciclo de conversão com o bit 2^0 igual a 1. A saída do DAC é 5 V porque existe um nível 1 no bit 2^2 e no bit 2^0; 4 V + 1 V = 5 V.

Todos os quatro bits foram testados, completando assim o ciclo de conversão. Nesse ponto o código binário no registrador é 0101, que é aproximadamente o valor binário equivalente à entrada de 5,1 V, Bits adicionais produzirão um resultado com precisão ainda maior. Agora um outro ciclo de conversão inicia, e o processo básico é repetido. O SAR é limpo (resetado) no início de cada ciclo.

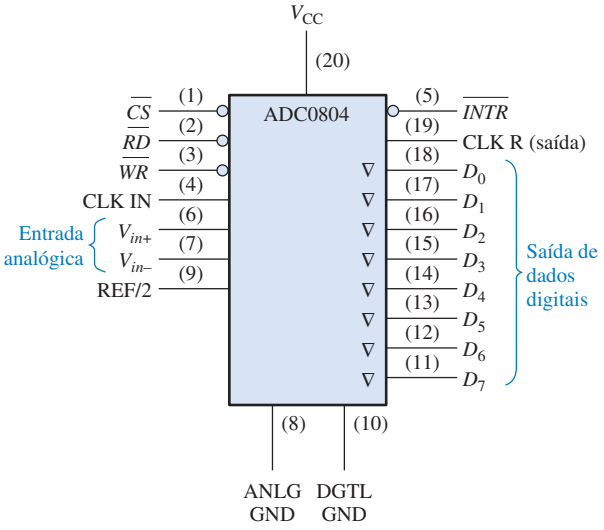
CONVERSOR ANALÓGICO-DIGITAL (ADC0804)



O CI ADC0804 é um exemplo de um ADC de aproximação sucessiva. Um diagrama em bloco típico é mostrado na Figura 13–21. Esse dispositivo opera a partir de uma fonte de +5 V e tem uma resolução de 8 bits com um tempo de conversão de 100?s. Além disso, ele tem um gerador de clock no chip. As saídas de dados são tristate, assim ele pode ser interfaceado com um sistema de barramento de um microprocessador.

► **FIGURA 13–21**

CI conversor analógico-digital ADC0804.



A operação básica do dispositivo é a seguinte: O ADC0804 contém o equivalente a um DAC com uma rede de 256 resistores. A lógica de aproximação sucessiva estabelece uma seqüência de ativação da rede para igualar à tensão de entrada diferencial analógica ($V_{in+} - V_{in-}$) com a tensão de saída da rede de resistores. O MSB é testado primeiro. Após oito comparações (sessenta e quatro períodos do clock), um código binário de 8 bits é transferido para os latches de saída e a saída de interrupção ($\overline{INTR}$) passa para nível BAIXO. O dispositivo pode ser operado no modo de conversões sucessivas conectando a saída $\overline{INTR}$ na entrada de escrita ($\overline{WS}$) e mantendo o inicio de conversão ($\overline{CS}$) em nível BAIXO. Para garantir uma partida sob todas as condições, uma entrada $\overline{WR}$ em nível BAIXO é necessária durante o ciclo de energização. Estando $\overline{CS}$ em nível BAIXO em qualquer momento após isso, interromperá o processo de conversão.

Quando a entrada $\overline{WR}$ vai para nível BAIXO, o registrador de aproximação sucessiva (SAR) interno e o registrador de deslocamento de 8 bits são resetados. Enquanto $\overline{CS}$ e $\overline{WR}$ permanecerem em nível BAIXO, o ADC permanece no estado de RESET. Uma conversão inicia oito períodos do clock após $\overline{CS}$ ou $\overline{WR}$ faz uma transição de nível BAIXO para ALTO.

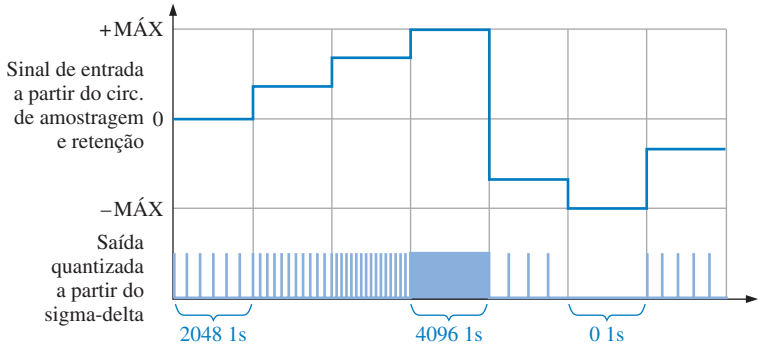
Quando um nível BAIXO estiver nas entradas $\overline{CS}$ e $\overline{RD}$, o latch de saída é habilitado e o código de saída é aplicado nas linhas D_0–D_7. Quando a entrada $\overline{CS}$ ou a entrada $\overline{RD}$ retornar para o nível ALTO, as saídas são desabilitadas.

Conversor Analógico-Digital Sigma-Delta

Sigma-delta é um método amplamente usado de conversão analógico-digital, particularmente em telecomunicações usando sinais de áudio. O método é baseado na **modulação delta** onde a diferença entre duas amostras sucessivas (crescente ou decrescente) é quantizada; os outros métodos ADC se baseiam no valor absoluto da amostra. A modulação delta é um método de quantização de 1 bit.

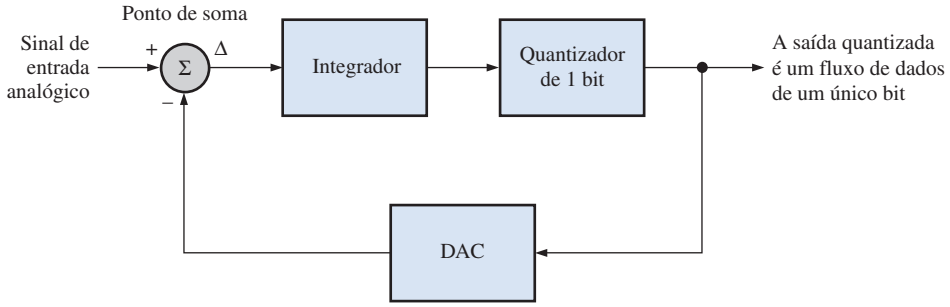
A saída de um modulador delta é uma seqüência de dados de um único bit onde o número relativo de 1s e 0s indicam o nível ou a amplitude do sinal de entrada. O número de 1s ao longo de um dado número de ciclos de clocks estabelece a amplitude do sinal durante o intervalo. Um número máximo de 1s corresponde à máxima tensão de entrada positiva. Um número de 1s igual à

metade do máximo corresponde a uma tensão de entrada zero. Nenhum 1 (todos os bits 0) corresponde à máxima tensão de entrada negativa. Isso está ilustrado de uma forma simplificada na Figura 13–22. Por exemplo, considere que 4096 1s ocorrem durante o intervalo quando o sinal de entrada estiver no máximo positivo. Como zero é o ponto médio da faixa dinâmica do sinal de entrada, 2048 1s ocorrem durante o intervalo quando o sinal de entrada é zero. Não existem 1s durante o intervalo quando o sinal de entrada está no máximo negativo. Para os níveis de sinal intermediários, o numero de 1s é proporcional ao nível.



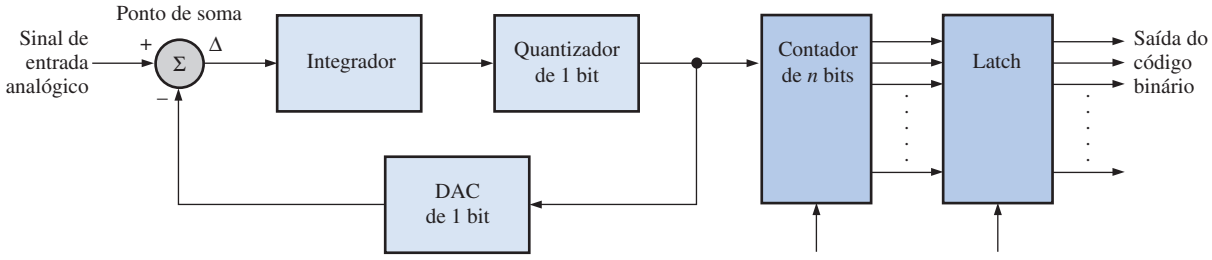
◀ **FIGURA 13–22**
Uma ilustração simplificada da conversão analógico-digital sigma-delta.

Diagrama em Bloco Funcional de um ADC Sigma-Delta O diagrama em bloco na Figura 13–23 realiza a conversão ilustrada na Figura 13–22. O sinal de entrada analógico e o sinal analógico a partir da seqüência de bit quantizada convertida do DAC no loop de realimentação são aplicados ao ponto de soma (∑). A diferença (Δ) do sinal que sai de ∑ é integrado, e o ADC de 1 bit aumenta ou diminui o número de 1s dependendo do sinal da diferença. Essa ação tenta manter o sinal quantizado realimentado igual ao sinal analógico de entrada. O quantizador de 1 bit é essencialmente um comparador seguido por um latch.



◀ **FIGURA 13–23**
Diagrama em bloco funcional parcial de um ADC delta-sigma.

Para completar o processo de conversão sigma-delta usando uma abordagem particular, o fluxo de dados de um único bit é convertido em uma série de códigos binários, como mostra a Figura 13–24. O contador conta os 1s no fluxo de dados quantizado por intervalos sucessivos. O códi-



▲ **FIGURA 13–24**
Um tipo de ADC sigma-delta.

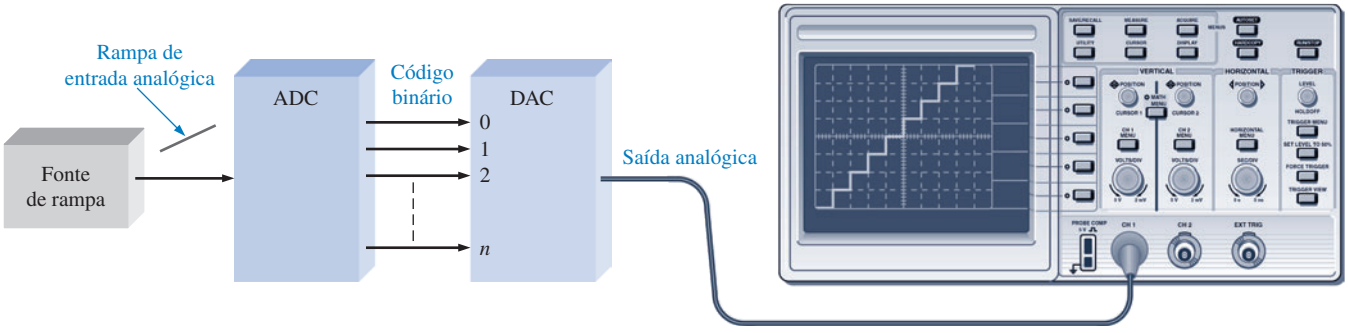
go no contador representa a amplitude do sinal analógico de entrada para cada intervalo. Esses códigos são deslocados para um latch para armazenamento temporário. O que vai parra o latch é uma série de códigos de *n* bits, os quais representam completamente o sinal analógico.

Uma outra abordagem usa um filtro de decimação digital para produzir a saída em vez de um contador e um latch. Esse assunto está além do escopo da nossa abordagem.

Teste de Conversores Analógico-Digital

Um método para teste de ADCs é mostrado na Figura 13–25. Um DAC é usado como parte do teste para converter a saída do ADC de volta para a forma analógica para comparação com a entrada de teste.

Uma entrada de teste na forma de uma rampa linear é aplicada na entrada do ADC. A saída binária resultante é então aplicada na unidade de teste DAC e convertida para uma rampa na forma de escada. As rampas de entrada e saída são comparadas atentando para qualquer desvio.



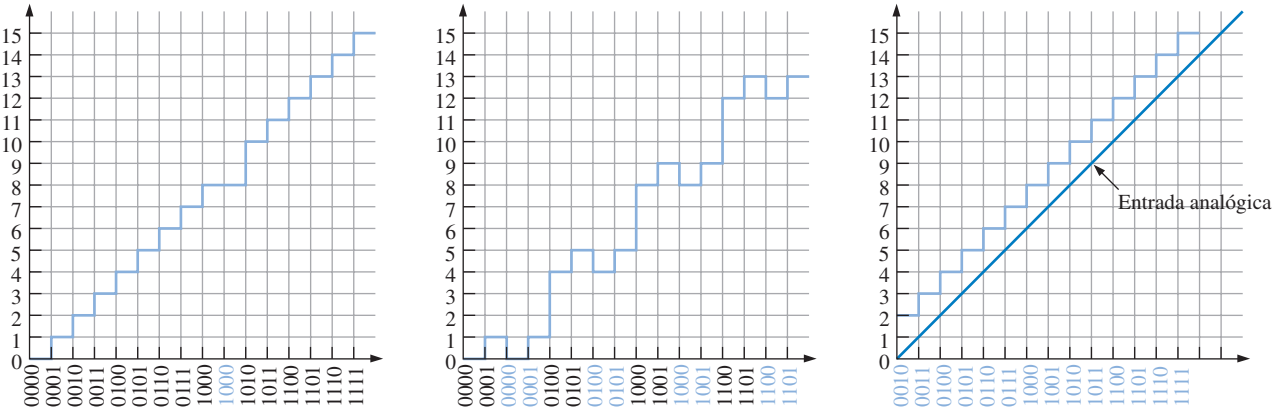
▲ **FIGURA 13–25**

Um método para teste de ADCs.

Erros na Conversão Analógico-Digital

Utilizamos novamente uma conversão de 4 bits para ilustrar os princípios. Vamos considerar que a entrada de teste seja uma rampa linear ideal.

Ausência de Código A saída na forma de escada mostrada na Figura 13–26(a) indica que o código binário 1001 não aparece na saída do ADC. Observe que o valor 1000 permanece por dois intervalos e então a saída salta para o valor 1010.



(a) Ausência de código (b) Códigos incorretos (c) Offset

▲ **FIGURA 13–26**

Ilustração de erros na conversão analógico-digital.

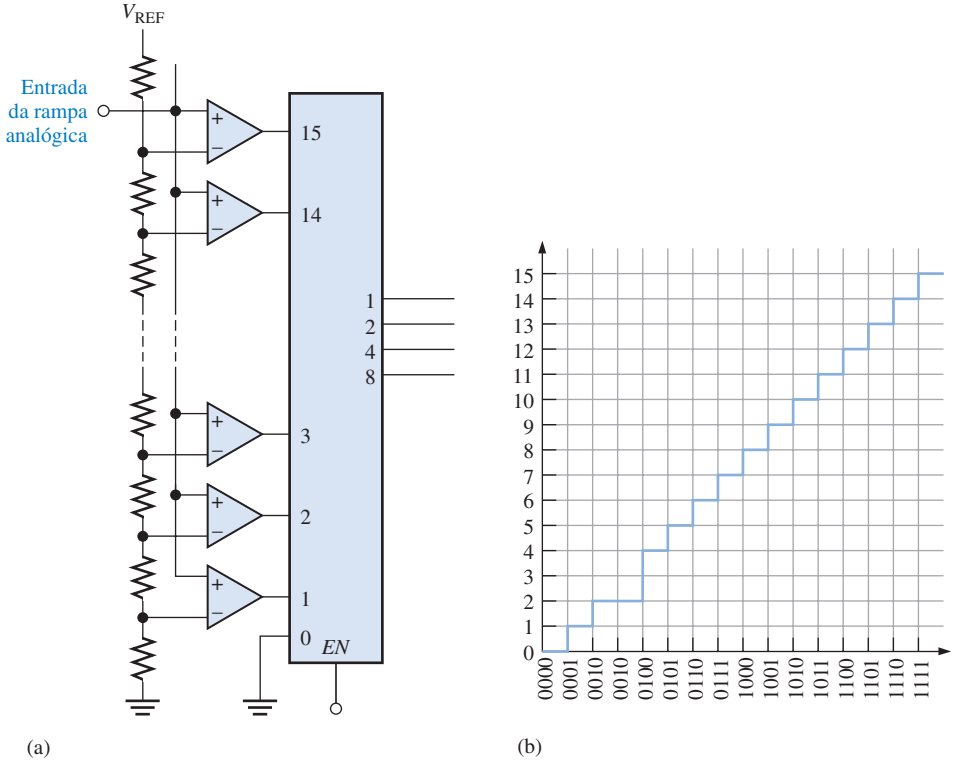
Em um ADC flash, por exemplo, uma falha de um dos comparadores (amp-op) pode causar um erro de ausência de código.

Códigos Incorretos A saída na forma de escada vista na Figura 13–26(b) indica que várias das palavras de código binário que saem do ADC estão incorretas. Uma análise indica que a linha do bit 2^1 está fixa em nível BAIXO (0) nesse caso em particular.

Offset As condições de *offset* são mostradas em 13–26(c). Nessa situação o ADC interpreta a tensão de entrada analógica como sendo maior que o valor real.

EXEMPLO 13–2

Um ADC flash de 4 bits é mostrado na Figura 13–27(a). Ele é testado com um sistema semelhante ao mostrado na Figura 13–25. A saída analógica reconstruída resultante é mostrada na Figura 13–27(b). Identifique o problema e o defeito mais provável.



(a) (b)

▲ **FIGURA 13–27**

Solução O código binário 0011 está ausente na saída do ADC, conforme indicado pelo degrau ausente. O mais provável é que a saída do comparador 3 esteja travada em seu estado inativo (BAIXO).

Problema relacionado Reconstrua a saída analógica em um teste semelhante ao da Figura 13–25 se o ADC dado na Figura 13–27(a) tiver o comparador 8 com a saída travada no estado ALTO.

SEÇÃO 13–3 REVISÃO

1. Qual o método de conversão analógico-digital mais rápido?
2. Qual o método de conversão analógico-digital que produz um fluxo de dados de um único bit?
3. O conversor de aproximação sucessiva tem um tempo de conversão fixo?
4. Cite os dois tipos de erro de saída de um ADC.

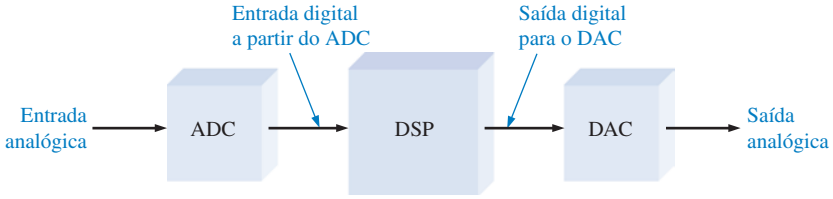
13-4 PROCESSADOR DE SINAIS DIGITAIS (DSP)

Essencialmente, um processador de sinais digitais (DSP) é um tipo especial de microprocessador que processa dados em tempo real. Suas aplicações se concentram no processamento de dados digitais que representam sinais analógicos. Um DSP, da mesma forma que um microprocessador, tem uma unidade central de processamento (CPU) e unidades de memória além de muitas funções de interface. Todas as vezes que usamos o telefone celular, estamos usando um DSP, e esse é apenas um exemplo das diversas aplicações.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar os conceitos básicos de um DSP ■ Listar algumas das aplicações de DSPs ■ Descrever as funções básicas de um DSP em um telefone celular ■ Discutir a série TMS320C6000 de DSP

O processador de sinais digitais (DSP) é o coração de uma sistema de processamento de sinais digitais. Ele obtém suas entradas a partir de um ADC e produz uma saída que vai para um DAC, conforme mostra a Figura 13–28. Conforme estudamos, o ADC muda uma forma de onda analógica para o formato digital como uma série de códigos binários que são então aplicados no DSP para processamento. Após isso, o dado vai para um DAC para ser convertido de volta para a forma analógica.



▶ **FIGURA 13–28**
O DSP tem uma entrada digital e produz uma saída digital.

Programação DSP

Os DSPs são tipicamente programados em linguagem assembly ou C. Como os programas escritos em linguagem assembly podem geralmente ser executados mais rápido e como a velocidade é crítica na maioria das aplicações de DSP, a linguagem assembly é muito mais usada em DSPs do que em microprocessadores de propósito geral. Além disso, os programas de DSP são geralmente muito menores que os programas tradicionais de microprocessadores porque eles são aplicações muito especializadas onde é utilizada muita redundância. Em geral, os conjuntos de instruções de DSPs tendem a ser menores que no caso de microprocessadores.

Aplicações de DSP

O DSP, diferentemente dos microprocessadores de propósito geral, tem que processar dados em *tempo real*; ou seja conforme eles são gerados. Muitas aplicações nas quais os DSPs são usados não toleram quaisquer atrasos notáveis, necessitando que o DSP seja extremamente rápido. Além de telefones celulares, os processadores de sinais digitais (DSPs) são usados em computadores com multimídia, gravadores de vídeo, aparelhos de CD, drives de disco rígido, modems de rádio digital e outras aplicações que melhoram a qualidade de um sinal. Além disso, os DSPs estão se tornando cada vez mais comuns em aplicações de televisão.

Uma aplicação importante de DSPs é na compressão e descompressão de sinais. Em sistemas de CD, por exemplo, a música no CD está em formato comprimido de forma que ele não usa muito espaço para armazenamento. Os dados têm que ser descomprimidos para serem reproduzidos. Além disso, a compressão de sinais é usada em telefones celulares para permitir um maior número de chamadas manipuladas por uma célula local. Algumas das outras áreas em que ele tem um impacto importante são discutidas a seguir.

Telecomunicações O campo das telecomunicações envolve a transferência de todos os tipos de informações de um local para outro, incluindo conversação telefônica, sinais de televisão e dados digitais. Dentre outras funções, o DSP facilita a multiplexação de muitos sinais em um canal de transmissão porque a informação no formato digital é relativamente fácil para multiplexar e demultiplexar.

No estágio final da transmissão em um sistema de telecomunicações, os DSPs são usados para comprimir sinais de voz digitalizados para conservação da largura de banda. A compressão é o processo de redução da taxa de dados. Geralmente, um sinal de voz é convertido para a forma digital a 8000 amostras por segundo (sps), baseado na freqüência de Nyquist de 4 kHz. Se são usados 8 bits para codificação de cada amostra, a taxa de dados é de 64 kbits/s. Em geral, a redução (compressão) da taxa de dados de 64 kbits/s para 32 kbits/s resulta em uma perda da qualidade do som. Quando os dados são comprimidos para 8 kbits/s, a qualidade do som é reduzida sensivelmente. Quando comprimido para o mínimo de 2kbits/s, o som é bastante distorcido mas ainda utilizável em algumas aplicações onde apenas o reconhecimento das palavras é importante, e não a qualidade. No receptor de um sistema de telecomunicações, o DSP descomprime os dados para recuperar o sinal para o seu formato original.

O eco, um problema na maioria das comunicações telefônicas de longa distância, ocorre quando uma parte do sinal de voz retorna com um atraso. Para distâncias curtas, esse atraso é pouco perceptível; mas à medida que a distância entre o transmissor e o receptor aumenta, aumenta também o atraso de tempo do eco. Os DSPs são usados para cancelar efetivamente o incômodo eco, o que resulta em um sinal de voz sem distúrbio e limpo.

Processamento de Música O DSP é usado na indústria da música para prover filtragem, adição e subtração de sinais e edição do sinal na preparação e gravação de músicas. Alem disso, uma outra aplicação de DSP é na inserção de eco e reverberação artificiais, os quais são geralmente minimizados pela acústica de um estúdio de som para simular ambientes de som ideais de grandes locais de concertos a pequenas salas.

Geração e Reconhecimento de Voz Os DSPs são usados na geração e reconhecimento de voz para melhorar a qualidade da comunicação homem/máquina. O método mais comum usado para produzir voz gerada por computador é a gravação digital. Na gravação digital, a voz humana é digitalizada e armazenada, geralmente na forma comprimida. Durante a reprodução da voz armazenada os dados são descomprimidos e convertidos de volta para o formato analógico original. Aproximadamente uma hora de voz pode ser armazenada usando cerca de 3 MB de memória.

O reconhecimento de voz é muito mais difícil para realizar que a geração de voz. Mesmo com os computadores atuais, o reconhecimento de voz é muito limitado e, com poucas exceções, os resultados são apenas de sucesso moderado. O DSP é usado para isolar e analisar cada palavra do sinal de voz de entrada. Certos parâmetros são identificados em cada palavra e comparados com exemplos prévios da pronúncia das palavras para criar uma correspondência de proximidade. A maioria dos sistemas são limitados a algumas centenas de palavras na melhor das hipóteses. Além disso, são necessários pausas significativas entre palavras e o sistema tem que ser "treinado" para uma voz específica. O reconhecimento de voz é uma área de grandes esforços de pesquisa que eventualmente terão muitas aplicações comerciais.

Radar Em aplicações de detecção e rastreamento via rádio (***ra****dio* ***d****etection* ***a****nd* ***r****anging*), os DSPs provêem mais precisão na determinação de distâncias usando técnicas de compressão, diminuição do ruído usando técnicas de filtragem, aumentando assim o alcance e otimizando a capacidade dos sistemas de radar de identificar tipos de objetos específicos. Os DSPs são também são usados de forma similar em sistemas de sonar.

Processamento de Imagem O DSP é usado em aplicações de processamento de imagem tal como a tomografia computadorizada (CT) e a imagem por ressonância magnética (MRI), as quais

NOTA: COMPUTAÇÃO

Os cartões de som usados em computadores usam um ADC para converter o som de um microfone, de um aparelho de CD de áudio, ou de uma outra fonte em um sinal digital. O ADC envia o sinal digital para um processador de sinais digitais (DSP). Baseado nas instruções armazenadas em uma ROM, uma função do DSP é a compressão do sinal digital de forma que ele use menos espaço de armazenamento. O DSP envia os dados comprimidos para o processador do computador que, por sua vez, envia os dados para o drive de disco rígido ou para um CD-ROM para serem armazenados. Para executar um som gravado, os dados armazenados são recuperados pelo processador e enviados ao DSP onde são descomprimidos e enviados a um DAC. A saída do DAC, que é a reprodução do sinal sonoro original, é aplicada nos alto-falantes.

são amplamente usadas em áreas médicas para observar o corpo humano por dentro. Na tomografia computadorizada, os raios X passam através de uma seção do corpo em diferentes direções. Os sinais resultantes são convertidos para a forma digital e armazenados. Essa informação armazenada é usada para produzir imagens calculadas que mostram fatias do corpo humano com muitos detalhes permitindo um melhor diagnóstico.

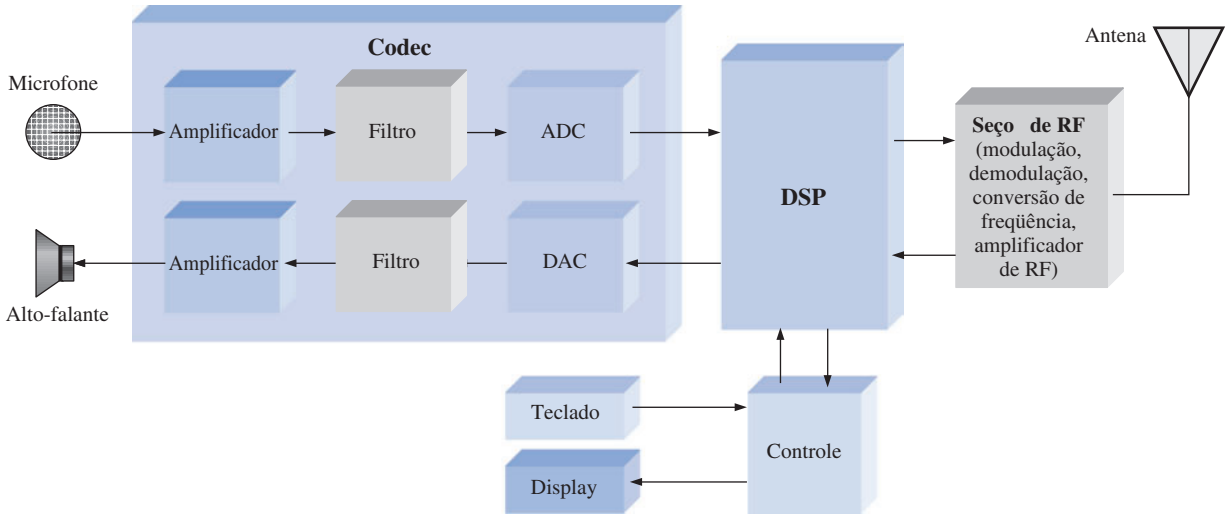
Em vez de raios X, a imagem por ressonância magnética usa campos magnéticos em conjunto com ondas de rádio para pesquisar o interior do corpo humano. O MRI produz imagens, assim como a tomografia computadorizada, e provê excelente discriminação entre diferentes tipos de tecidos bem como informações tais como o fluxo sangüíneo através das artérias. A imagem por ressonância magnética depende completamente dos métodos de processamento de sinais digitais.

Em aplicações como vídeo-fone, televisão digital e outros meios que provêem imagens em movimento, o DSP usa compressão de imagem para reduzir o número de bits necessários, tornando esses sistemas comercialmente possíveis.

Filtragem Os DSPs são normalmente usados para implementar filtros digitais com a finalidade de separar sinais que foram combinados com outros sinais ou com interferência e ruído e para recuperação de sinais distorcidos. Embora os filtros analógicos sejam bastante adequados para algumas aplicações, o filtro digital é geralmente muito superior em termos do desempenho que pode ser alcançado. Uma desvantagem dos filtros digitais é que o tempo de execução necessário produz um atraso a partir do instante que o sinal analógico é aplicado até o momento que aparece na saída. Os filtros analógicos não apresentam problema de atraso porque logo que o sinal é colocado na entrada, a resposta aparece na saída. Os filtros analógicos também são mais baratos que os digitais. Independente disso, o desempenho geral dos filtros digitais é muito superior em muitas aplicações.

DSP em um Telefone Celular

O telefone celular digital é um exemplo de como um DSP pode ser usado. A Figura 13–29 mostra um diagrama em bloco simplificado de um telefone celular digital. O **codec** (abreviação de codificador/decodificador) de voz contém, dentre outras funções, o ADC e o DAC são necessários para a conversão entre o sinal analógico de voz e o formato digital da mesma. A conversão sigma-delta é usada normalmente na maioria das aplicações de celulares. Para a transmissão, o sinal de voz do microfone é convertido para a forma digital pelo ADC no codec e então vai para o DSP para processamento. A partir do DSP, o sinal digital vai para a seção de RF (rádio freqüência) onde é modulado e preparado para ser transmitido em rádio freqüência. Um sinal de RF, que contém as informações de voz, é capturado pela antena, demodulado e convertido em um sinal digital. Este é então aplicado no DSP para processamento, após o qual o sinal digital vai para o codec para ser convertido de volta para o formato original por um DAC. Em seguida é amplificado e enviado para os alto-falantes.



▲ **FIGURA 13–29**

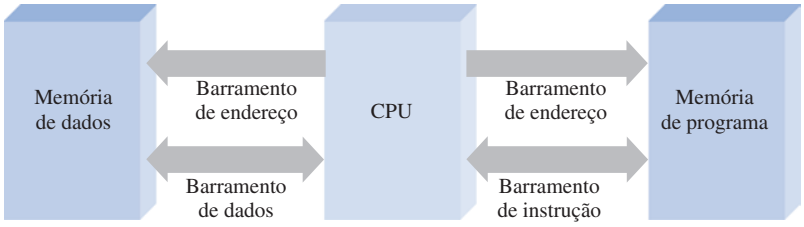
Diagrama em bloco simplificado de um telefone celular.

Funções Realizadas por um DSP Em um celular, o DSP realiza muitas funções para melhorar e facilitar a recepção e a transmissão do sinal de voz. Algumas dessas funções do DSP são as seguintes:

- *Compressão de voz.* A taxa do sinal de voz é reduzida significativamente para a transmissão para se enquadrar nos requisitos da largura de banda.
- *Descompressão de voz.* A taxa do sinal de voz digital recebido retorna para a sua taxa original para reproduzir de forma adequada o sinal de voz analógico.
- *Operação com Protocolo.* O telefone celular se comunica com a estação base mais próxima para estabelecer a localização do celular, alocar *slots* de tempo e freqüência e estabelecer um *handover* com uma outra estação base à medida que o celular se move para uma outra célula.
- *Detecção e correção de erros.* Durante a transmissão, os códigos de detecção e correção de erros são gerados e, durante a recepção, detecta e corrige os erros induzidos no canal RF por ruído ou interferência.
- *Criptografia.* Converte o sinal de voz digital para uma forma de transmissão segura e converte de volta para a forma original durante a recepção.

Arquitetura Básica de um DSP

Conforme mencionado antes, um DSP é basicamente um microprocessador especializado e otimizado em termos de velocidade para processar dados em tempo real. Muitos DSPs são baseados no que ficou conhecido como *arquitetura Harvard*, a qual consiste de uma unidade central de processamento (CPU) e duas memórias, uma para dados e outra para programa, como mostra o diagrama em bloco da Figura 13–30.



◀ **FIGURA 13–30**
Muitos DSPs a arquitetura Harvard (duas memórias).

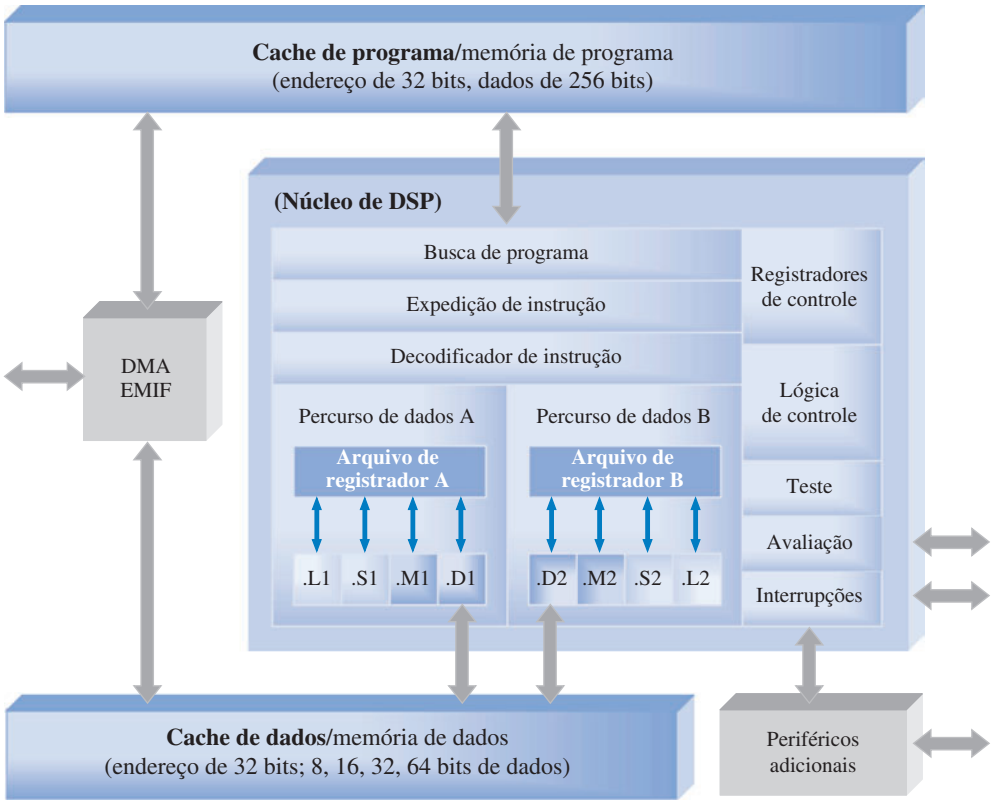
DSPs Específicos — Série TMS320C6000

Os DSPs são fabricados por diversas companhias incluindo a Texas Instruments, Motorola e Analog Devices. Os DSPs encontrados realizam processamento de ponto fixo e ponto flutuante. Lembre-se do Capítulo 2 que esses dois métodos diferem na forma com que os números são armazenados e manipulados. Todos os DSPs de ponto flutuante podem também manusear números no formato de ponto fixo. Os DSPs de formato de ponto fixo são mais baratos que os de ponto flutuante e, geralmente, são mais rápidos. Os detalhes da arquitetura DSP podem variar significativamente, mesmo dentro da mesma família. Vamos estudar de forma geral uma série DSP particular como exemplo de como um DSP é geralmente organizado.

Exemplos de DSPs comercializados na série TMS320C6000 incluem o TMS320C62xx, TMS320C64xx e TMS320C67xx os quais são parte da família TMS320 de dispositivos. Um diagrama em bloco geral para esses dispositivos é mostrado na Figura 13–31.

Os DSPs têm uma unidade central de processamento (CPU), também conhecida como **núcleo de DSP**, a qual contém 64 registradores de 32 bits na C64xx e 32 registradores de propósito geral de 32 bits na C62xx e C67xx. A C67xx pode realizar operações de ponto flutuante, ao passo que a C62xx e C64xx são dispositivos de ponto fixo.

Cada DSP tem oito unidades funcionais que contêm dois multiplicadores de 16 bits e seis unidades lógicas e aritméticas (ALUs). O desempenho desses três DSPs da série C6000 em termos



▲ **FIGURA 13–31**

Diagrama em bloco geral do DSP da série TMS320C6000.

de **MIPS** (Milhões de Instruções Por Segundo), **MFLOPS** (Milhões de Operações de Ponto Flutuante Por Segundo) e **MMACS** (Milhões de Multiplicações/Acumulações por Segundo) é mostrado na Tabela 13–3.

▶ **TABELA 13–3**

Desempenho de processamento de dados do DSP da série TMS320C6000.

DSP	TIPO	APLICAÇÃO	VELOCIDADE DE PROCESSAMENTO	VELOCIDADE DE MULTIPLICAÇÃO/ ACUMULAÇÃO
C62xx	Ponto fixo	Propósito geral	1200–2400 MIPS	300–600 MMACS
C64xx	Ponto fixo	Propósito especial	3200–4800 MIPS	1600–2400 MMACS
C67xx	Ponto flutuante	Propósito geral	600–1000 MFLOPS	200–333 MMACS

Percurso dos Dados na CPU Na CPU, as seções de busca do programa, expedição de instruções e decodificação de instruções podem fornecer oito instruções de 32 bits para as unidades funcionais durante cada ciclo do clock. A CPU é dividida em dois percursos de dados e o processamento de instruções ocorre nos percursos A e B. Cada percurso de dado contém metade dos registradores de propósito geral (16 no C62xx e C67xx ou 32 no C64xx) e quatro unidades funcionais. O registrador de controle e a lógica de controle são usados para configurar e controlar as diversas operações do processador.

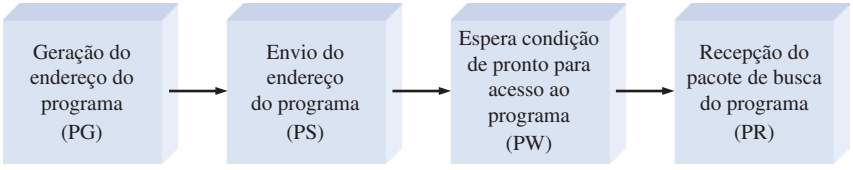
Unidades Funcionais Cada percurso de dados tem quatro unidades funcionais. As unidades M (indicadas por .M1 e .M2 na Figura 13–31) são multiplicadores dedicados. As unidades L (indicadas por .L1 e. L2) realizam operações lógicas, aritméticas e mistas. As unidades S (indicadas

por .S1 e .S2) realizam comparação, deslocamento e operações aritméticas mistas. As unidades D (indicadas por .D1 e .D2) realizam carga, armazenamento e operações mistas.

Pipeline Um **pipeline** permite que instruções múltiplas sejam processadas simultaneamente. Uma operação *pipeline* consiste de três estágios através dos quais todas as instruções passam: busca, decodificação e execução. Oito instruções de cada vez são buscadas primeiro a partir da memória de programa; então elas são decodificadas e finalmente executadas.

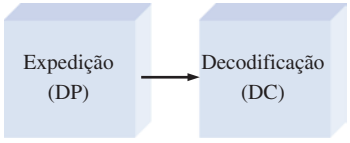
Durante a **busca**, as oito instruções (denominadas de pacote) são obtidas da memória em quatro fases, como mostra a Figura 13–32.

- *Geração do endereço do programa (PG)* O endereço do programa é gerado pela CPU.
- *Envio do endereço do programa (PS)* O endereço do programa é enviado para a memória.
- *Espera condição de pronto para acesso ao programa (PW)* Ocorre uma operação de leitura na memória.
- *Recepção do pacote de busca do programa (PR).* A CPU recebe o pacote de instruções.



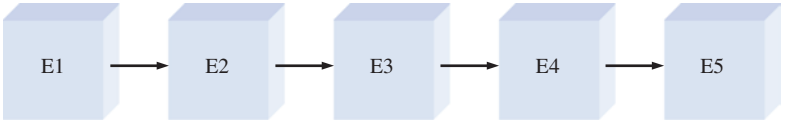
◀ FIGURA 13–32
As quatro fases da busca na operação *pipeline*.

Duas fases formam o estágio de **decodificação** da instrução da operação *pipeline*, como mostra a Figura 13–33. A fase de expedição de instrução (DP) é onde os pacotes de instruções são divididos em pacotes de execução e associados às unidades funcionais apropriadas. A fase de decodificação da instrução (DC) é onde as instruções são decodificadas.



◀ FIGURA 13–33
As duas fases de decodificação da operação *pipeline*.

O estágio de **execução** da operação *pipeline* é onde as instruções do estágio de decodificação são executadas. O estágio de execução tem um máximo de cinco fases. O número de fases usadas durante a execução depende do tipo de instrução. Parte da execução de uma instrução necessita obter dados a partir da memória de dados.

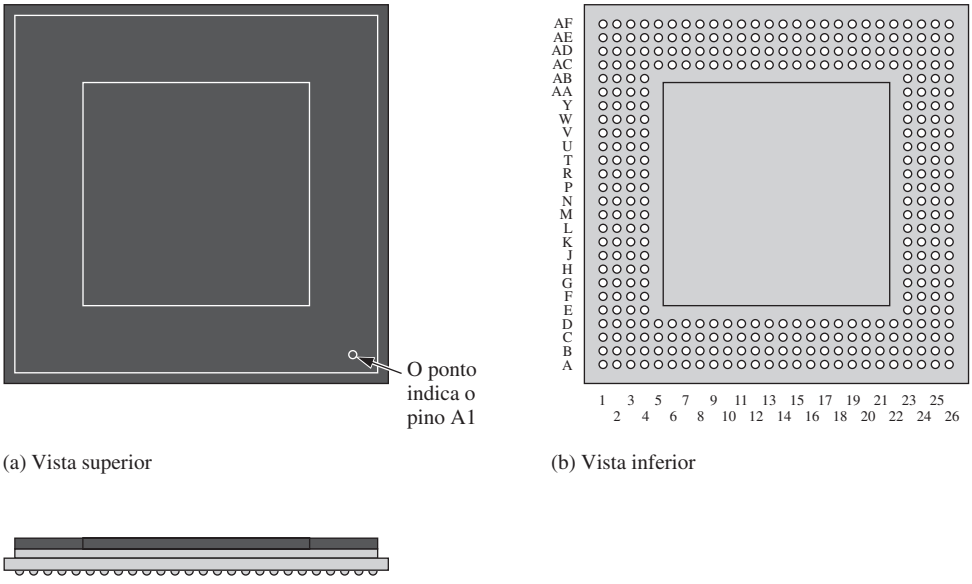


◀ FIGURA 13–34
As cinco fases de execução da operação *pipeline*.

Memória DSP Interna e Interfaces Como podemos ver na Figura 13–31, existem duas memórias internas, uma para dados e outra para programas. A memória de programa é organizada em pacotes de 256 bits (oito instruções de 32 bits) e existem 64 kB de capacidade. A memória de dados também tem uma capacidade de 64 kB e pode ser acessada em comprimentos de palavras de 8, 16, 32 ou 64 bits, dependendo do dispositivo específico da série. As duas memórias internas são acessadas com endereços de 32 bits. O DMA (acesso direto à memória) é usado para transferir dados sem passar pela CPU. A EMIF (Interface de Memória Externa) é usada para suportar memórias externas quando necessário em aplicações. Uma interface adicional é fornecida para portas de I/O seriais e outros dispositivos externos.

Temporizadores Existem dois temporizadores de propósito geral no DSP que podem ser usados para eventos temporizados, contagem, geração de pulsos, interrupção da CPU entre outras finalidades.

Encapsulamento Esses processadores em particular são comercializados em encapsulamentos BGA (*ball grid array*) de 352 pinos, conforme mostra a Figura 13–35 e são implementados com tecnologia CMOS.



(a) Vista superior

(b) Vista inferior

(c) Vista lateral

▲ **FIGURA 13–35**

Um encapsulamento BGA de 352 pinos.

SEÇÃO 13–4 REVISÃO

1. O que significa arquitetura Harvard?
2. O que é um núcleo de DSP?
3. Cite as duas categorias de DSP de acordo com o tipo de números manuseados.
4. Quais são os dois tipos de memória interna?
5. Defina (a) MIPS (b) MFLOPS (c) MMACS.
6. Basicamente, o que faz o *pipeline*?
7. Cite os três estágios de operação *pipeline*?
8. O que acontece durante a fase de busca?

13-5 MÉTODOS DE CONVERSÃO DIGITAL-ANALÓGICO

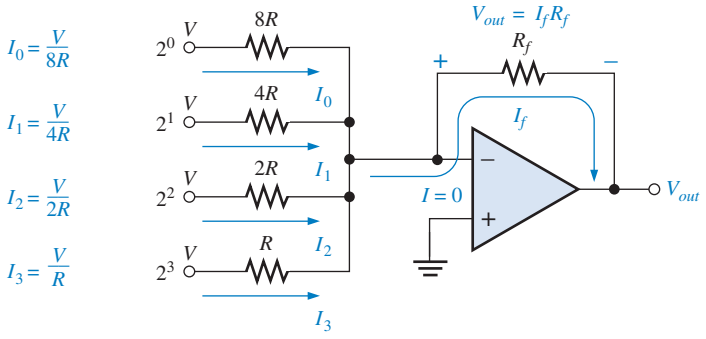
A conversão digital-analógico é uma parte importante do sistema de processamento digital. Uma vez que os dados digitais foram processados pelo DSP, eles são convertidos de volta para a forma analógica. Nessa seção, examinaremos a teoria de operação dos dois tipos básicos de conversores digital-analógico (DACs) e aprenderemos sobre as características de desempenho deles.

Ao final do estudo desta seção você deverá ser capaz de:

■ Explicar a operação de um DAC com entrada ponderada binária ■ Explicar a operação de um DAC com escada R/2R ■ Discutir resolução, precisão, linearidade, monotonicidade e tempo de ajuste em um DAC ■ Discutir o teste de DACs para não-monotonicidade, não-linearidade diferencial, ganho baixo ou alto e erro de *offset*

Conversor Digital-Analógico com Entrada Ponderada em Binário

Um método de conversão digital-analógico usa um circuito de resistores com valores de resistências que representam os pesos binários dos bits de entrada do código digital. A Figura 13–36 mostra um DAC de 4 bits desse tipo. Cada um dos resistores de entrada terá ou não corrente dependendo do nível de tensão na entrada. Se a tensão de entrada for nível ALTO (binário 1), a intensidade da corrente depende do valor do resistor de entrada e é diferente para cada resistor de entrada, conforme indicado na figura.



◀ **FIGURA 13–36**
Um DAC de 4 bits com entradas ponderadas em binário.

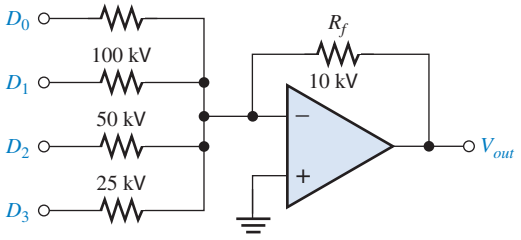
Como praticamente não existe corrente na entrada inversora (–) do amp-op, todas as correntes de entrada se somam e passam no resistor R_f. Como a entrada inversora é 0 V (terra virtual), a queda de tensão em R_f é igual à tensão de saída, assim $V_{saída} = I_f R_f$.

Os valores dos resistores de entrada são escolhidos para serem inversamente proporcionais aos pesos binários dos bits de entrada correspondentes. O resistor de menor valor (R) corresponde à maior entrada ponderada (2^3). Os outros resistores são múltiplos de R (ou seja, $2R$, $4R$ e $8R$) e correspondem aos pesos binários 2^2, 2^1 e 2^0, respectivamente. As correntes de entrada também são proporcionais aos pesos binários. Portanto, a tensão de saída é proporcional à soma dos pesos binários porque a soma das correntes de entrada passam por R_f.

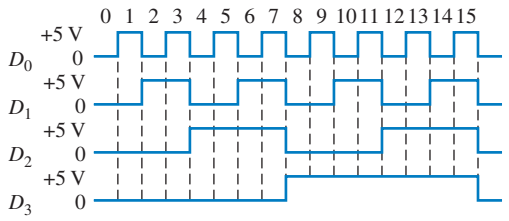
As desvantagens desse tipo de DAC são o número de valores de resistores diferentes e o fato de que os níveis de tensão têm que ser exatamente o mesmo para todas as entradas. Por exemplo, um conversor de 8 bits necessita de oito resistores, com valores que variam de R a $128R$ em degraus ponderados em binário. Essa faixa de resistores requer tolerâncias de uma parte em 255 (menos de 0,5%) para converter com precisão a entrada, o que torna esse tipo de DAC muito difícil de produzir em escala.

EXEMPLO 13–3

Determine a saída do DAC mostrado na Figura 13–37(a) se as formas de onda que representam a seqüência de números de 4 bits, mostradas na Figura 13–37(b), são aplicadas às entradas. A entrada D_0 é o bit menos significativo (LSB).



(a)



(b)

▲ **FIGURA 13–37**

Solução Primeiro, determine a corrente para cada uma das entradas ponderadas. Como a entrada inversora (–) do amp-op está em 0 V (terra virtual) e um binário 1 corresponde a +5 V, a corrente através de qualquer um dos resistores de entrada é 5 V dividido pelo valor da resistência.

$$I_0 = \frac{5\text{ V}}{200\text{ k}\Omega} = 0{,}025\text{ mA}$$

$$I_1 = \frac{5\text{ V}}{100\text{ k}\Omega} = 0{,}05\text{ mA}$$

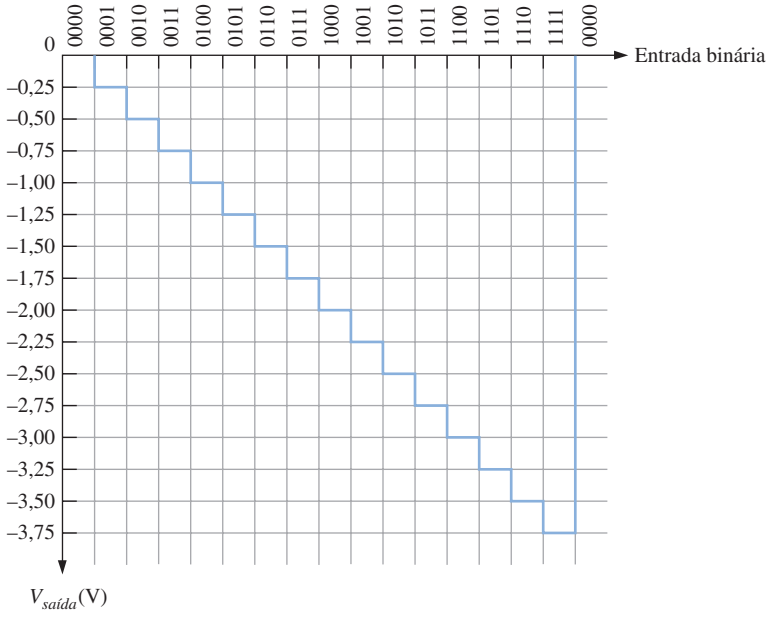
$$I_2 = \frac{5\text{ V}}{50\text{ k}\Omega} = 0{,}1\text{ mA}$$

$$I_3 = \frac{5\text{ V}}{25\text{ k}\Omega} = 0{,}2\text{ mA}$$

Quase nenhuma corrente vai para a entrada inversora do amp-op por causa da sua impedância extremamente alta. Portanto, considere que toda a corrente passa pelo resistor de realimentação R_f. Como uma extremidade de R_f está em 0 V (terra virtual), a queda de tensão em R_f é igual a tensão de saída, a qual é negativa em relação ao terra virtual.

$$V_{saída(D0)} = (10\text{ k}\Omega)(-0{,}025\text{ mA}) = -0{,}25\text{ V}$$
$$V_{saída(D1)} = (10\text{ k}\Omega)(-0{,}05\text{ mA}) = -0{,}5\text{ V}$$
$$V_{saída(D2)} = (10\text{ k}\Omega)(-0{,}1\text{ mA}) = -1\text{ V}$$
$$V_{saída(D3)} = (10\text{ k}\Omega)(-0{,}2\text{ mA}) = -2\text{ V}$$

A partir da Figura 13–37(b), o primeiro código binário de entrada é 0000, o qual produz uma tensão de saída de 0 V. O próximo código de entrada é 0001, o qual produz uma tensão de saída de –0,25 V. Por isso a tensão de saída é –0,25 V. O próxima código é 0011, o qual produz uma tensão de saída de –0,25 V + –0,5 V = –0,75 V. Cada código binário sucessivo aumenta a tensão de saída em –0,25 V, assim para essa seqüência binária direta nas entradas, a saída apresenta uma forma de onda em escada indo de 0 V a –3,75 V em degraus de –0,25 V. Isso é mostrado na Figura 13–38.

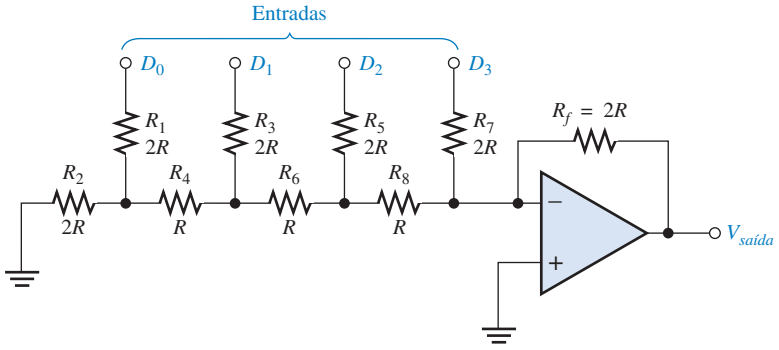


► **FIGURA 13-38**
Saída do DAC mostrado na Figura 13–37.

Problema relacionado Inverta as formas de onda de entrada do DAC mostradas na Figura 13–37 (D_3 com D_0 e D_2 com D_1) e determine a saída.

Conversor Digital-Analógico com Escada R/2R

Um outro método de conversão digital-analógico é a escada *R*/2*R*, como mostra a Figura 13–39 para quatro bits. Esse tipo de circuito supera o problema do DAC com entrada binária ponderada, pois requer apenas dois valores de resistores.



◀ **FIGURA 13–39**
DAC com escada *R*/2*R*.

Comece considerando que a entrada D_3 seja nível ALTO (+5 V) e as outras entradas sejam nível BAIXO (0 V, ou GND). Essa condição representa o número binário 1000. Uma análise do circuito mostra que ele se reduz à forma equivalente mostrada na Figura 13–40(a). Essencialmente nenhuma corrente percorre a resistência equivalente 2*R* porque a entrada inversora está no terra virtual. Portanto, toda a corrente (I = 5V/2*R*) que passa em R_7 também passa em R_f e a tensão de saída é –5 V. o amplificador operacional mantém a entrada inversora (–) próximo de zero volts (≈0 V) devido à realimentação negativa. Portanto, toda a corrente passa em R_f em vez de ir para a entrada inversora.

A Figura 13–40(b) mostra o circuito equivalente quando a entrada D_2 está em +5 V e as outras entradas estão em GND. Essa condição representa 0100. Se "thevenizarmos"* olhando a partir de R_8 teremos 1,25 V em série com *R* como mostrado. Isso resulta em uma corrente através de R_f de I = 2,5 V/2*R*, a qual proporciona uma tensão de saída de –2,5 V. Tenha em mente que não existe corrente na entrada inversora do amp-op e que não existe corrente através da resistência equivalente para GND porque a tensão nessa resistência é 0 V, devido ao terra virtual.

A Figura 13–40(c) mostra o circuito equivalente quando a entrada D_1 está em +5 V e as outras estão em GND. Essa condição representa 0010. Thevenizando novamente olhando a partir de R_8, temos 1,25 V em série com *R* como mostrado. Isso resulta em uma corrente através de R_f de I = 1,25 V/2*R*, a qual proporciona uma tensão de saída de –1,25 V.

Na parte (d) da Figura 13–40, o circuito equivalente que representa o caso onde D_0 está em +5 V e as outras entradas estão em GND é mostrado. Essa condição representa 0001. Thevenizando a partir de R_8 obtemos uma tensão de 0,625 V em série com uma resistência *R* como mostrado. A corrente resultante através de R_f é I = 0,625 V/2*R*, o que resulta em uma tensão de saída de –0,625 V.

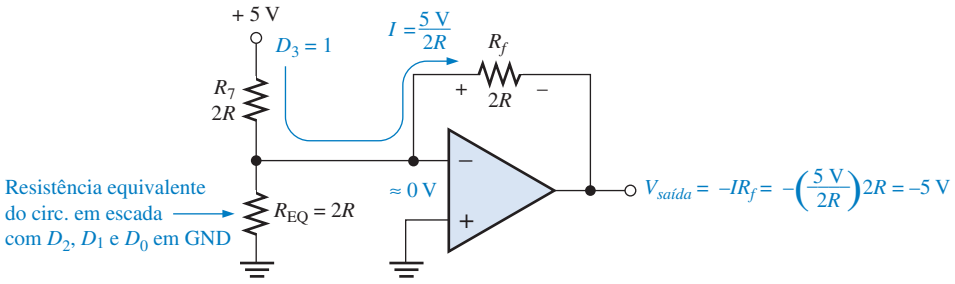
Observe que cada entrada ponderada com um peso sucessivamente menor produz uma tensão de saída que metade da anterior, assim a tensão de saída é proporcional ao peso binário dos bits de entrada.

Características de Desempenho de Conversores Digital-Analógico

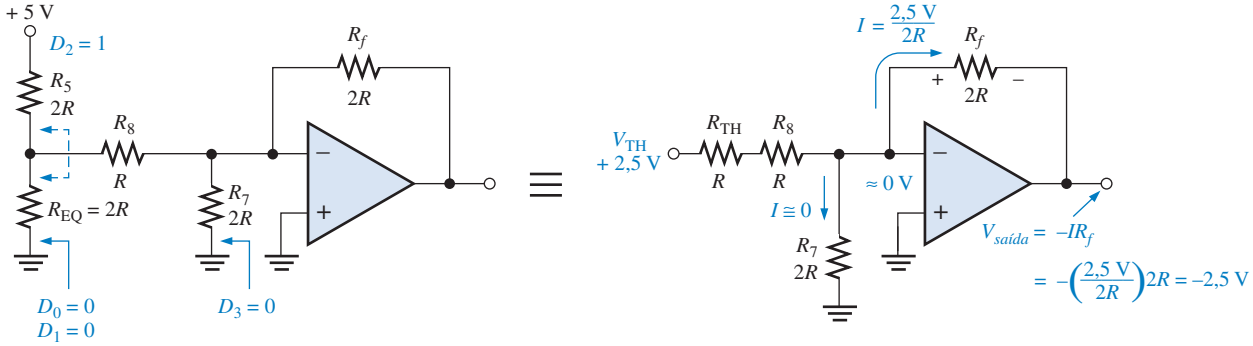
As características de desempenho de um DAC incluem resolução, precisão, linearidade, monotonicidade e tempo de ajuste, sendo cada um dos quais discutidos na seguinte lista:

- *Resolução*. A resolução de um DAC é o inverso do número de degraus discretos na saída. É claro que isso é independente do número de bits de entrada. Por exemplo, um DAC de 4 bits tem uma resolução de uma parte em $2^4 - 1$ (uma parte em quinze). Expresso como porcentagem temos (1/15)100 = 6,67%. O número total de degraus discretos é igual a $2^n - 1$, onde *n* é o numero de bits. A resolução também pode ser expressa como o número de bits da conversão.

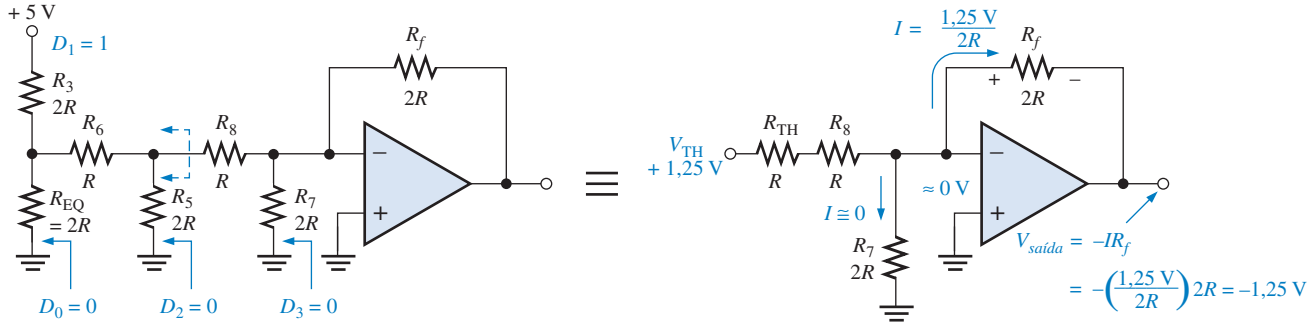
* O teorema de Thévenin diz que qualquer circuito pode ser reduzido a uma fonte de tensão equivalente em série com uma resistência equivalente.



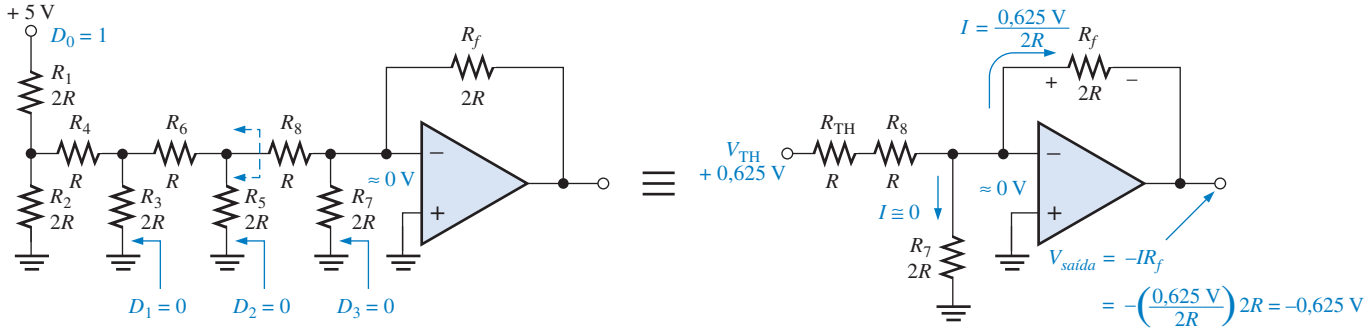
(a) Circuito equivalente para $D_3 = 1, D_2 = 0, D_1 = 0, D_0 = 0$



(b) Circuito equivalente para $D_3 = 0, D_2 = 1, D_1 = 0, D_0 = 0$



(c) Circuito equivalente para $D_3 = 0, D_2 = 0, D_1 = 1, D_0 = 0$



(d) Circuito equivalente para $D_3 = 0, D_2 = 0, D_1 = 0, D_0 = 1$

▲ **FIGURA 13–40**

Análise de um DAC com escada *R*/2*R*.

- *Precisão*. A precisão é deduzida da comparação da saída real com a saída esperada. Ela é expressa em porcentagem do fundo de escala, ou valor máximo da tensão de saída. Por exemplo, se um conversor tem uma tensão de fundo de escala de 10 V e a precisão é ±0,1%, então o erro máximo para qualquer tensão de saída é (10 V)(0,001) = 10 mV. Idealmente, a precisão deve ser não pior que ±1/2 do bit menos significativo. Para um conversor de 8 bits, o bit menos significativo é 0,39% do fundo de escala. A precisão deve ser aproximadamente ±0,2%,
- *Linearidade*. Um erro linear é um desvio a partir da reta ideal de saída de um DAC. Um caso especial é o erro de *offset*, o qual corresponde ao valor da tensão de saída quando todas as entradas são zero.
- *Monotonicidade*. Um DAC é **monotônico** se ele não apresenta nenhum degrau reverso quando recebe uma seqüência de bits de entrada em toda a faixa.
- *Tempo de ajuste*. O tempo de ajuste é normalmente definido como o tempo que DAC leva para se ajustar dentre de ±1/2 LSB do seu valor final quando ocorre uma variação no código de entrada.

EXEMPLO 13–4

Determine a resolução, expressa como uma porcentagem, dos seguintes conversores:

(a) um DAC de 8 bits **(b)** um DAC de 12 bits

Solução

(a) Para um conversor de 8 bits,

$$\frac{1}{2^8 - 1} \times 100 = \frac{1}{255} \times 100 = \mathbf{0{,}392\%}$$

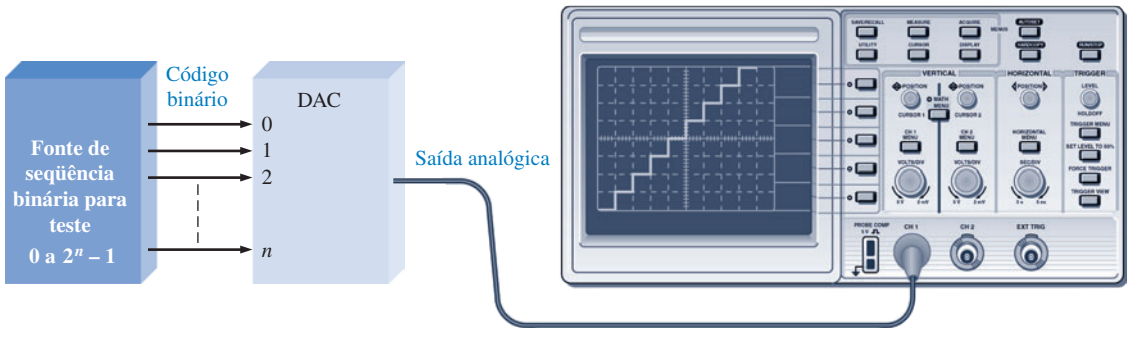
(b) Para um conversor de 12 bits,

$$\frac{1}{2^{12} - 1} \times 100 = \frac{1}{4095} \times 100 = \mathbf{0{,}0244\%}$$

Problema relacionado Calcule a resolução para um DAC de 16 bits.

Teste de Conversores Digital-analógico

O conceito do teste de um DAC é ilustrado na Figura 13–41. Nesse método básico, uma seqüência de códigos binários é aplicada às entradas, sendo a saída resultante observada. A seqüência de códigos binários se estende por toda a faixa de 0 a $2^n - 1$ em ordem ascendente, onde n é o número de bits.



▲ **FIGURA 13–41**

Configuração básica para teste de um DAC.

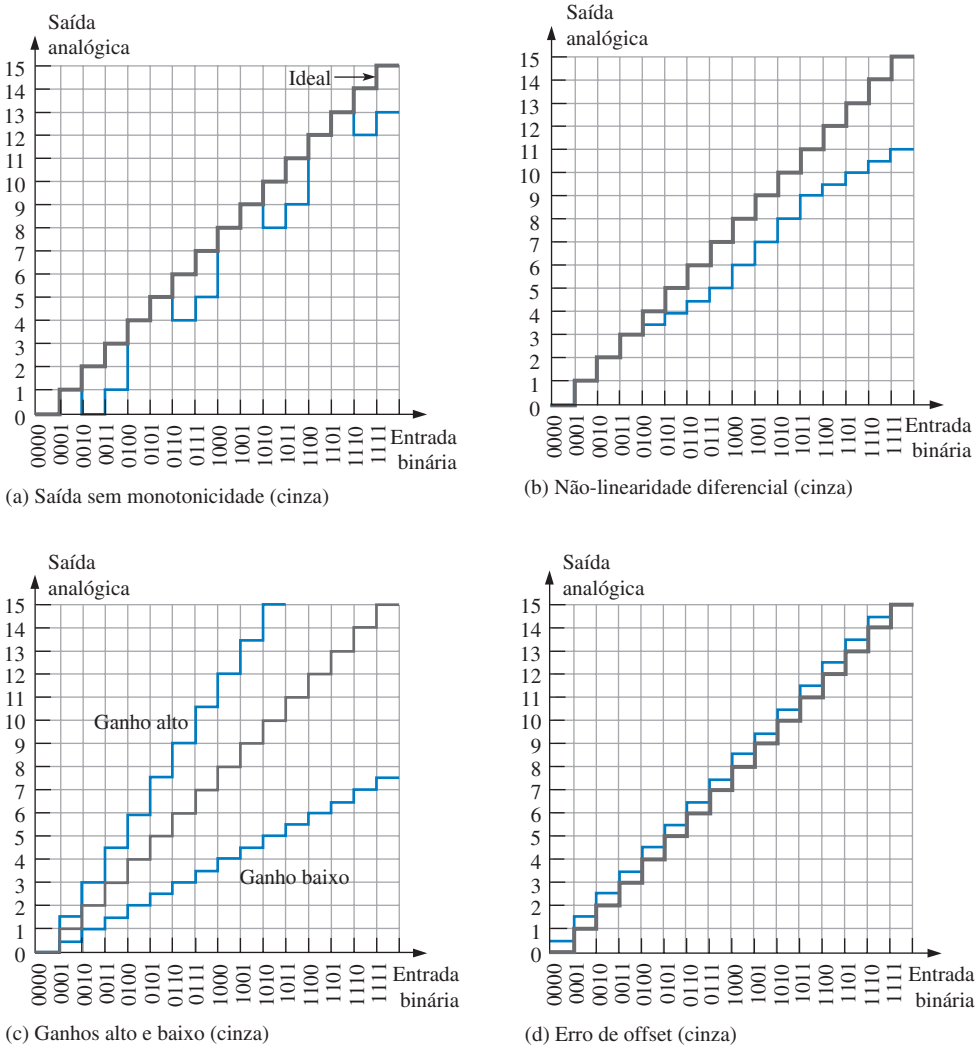
A saída ideal é uma linha reta, em degraus, conforme indicado. À medida que o número de bits no código binário aumenta, a resolução é melhorada. Ou seja, o número de degraus discretos aumenta e a saída se aproxima de uma rampa reta linear.

Erros de Conversão Digital-Analógico

Alguns erros de conversão digital-analógico a serem verificados são mostrados na Figura 13–42, os quais usam uma conversão de 4 bits para fins ilustrativos. Uma conversão de 4 bits produz quinze degraus discretos. Cada gráfico na figura inclui uma rampa, na forma de escada, ideal para fins de comparação com uma saída com erro.

Falta de Monotonicidade O degrau invertido na Figura 13–42(a) indica o desempenho de monotonicidade, o qual apresenta uma forma de não-linearidade. Nesse caso em particular, o erro ocorre porque o bit 2^1 no código binário é interpretado como uma constante 0. Ou seja, um curto-circuito está fazendo com que uma linha de entrada fique presa em nível BAIXO.

Não-Linearidade Diferencial A Figura 13–42(b) ilustra a não-linearidade diferencial na qual a amplitude de um degrau é menor do que deveria ser para certos códigos de entrada. Essa saída em particular pode ser provocada pelo peso insuficiente do bit 2^2, talvez por causa de um defeito no resistor de entrada. Poderíamos também estar observando degraus com amplitudes maiores que o normal caso um peso binário fosse maior do que deveria ser.



(a) Saída sem monotonicidade (cinza)

(b) Não-linearidade diferencial (cinza)

(c) Ganhos alto e baixo (cinza)

(d) Erro de offset (cinza)

FIGURA 13–42
Ilustrações de alguns erros na conversão digital-analógico.

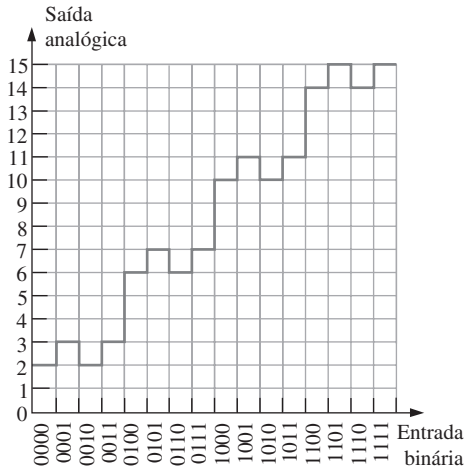
Ganho Baixo ou Alto Erros de saída provocados por um ganho baixo ou alto são ilustrados na Figura 13–42(c). No caso de ganho baixo, a amplitude de todos os degraus são menores que o ideal. No caso de ganho alto, a amplitude de todos os degraus são maiores que o ideal. Essa situação pode ser causada por um defeito no resistor de realimentação no circuito do amp-op.

Erro de Offset Um erro de *offset* é ilustrado na Figura 13–42(d). Observe que quando a entrada binária é 0000, a tensão de saída não é zero e que esse valor de *offset* é o mesmo para todos os degraus na conversão. Um defeito no amp-op pode ser o culpado dessa situação.

EXEMPLO 13–5

A saída do DAC vista na Figura 13–43 é observada quando uma seqüência binária de 4 bits direta é aplicada nas entradas. Identifique o tipo de erro e sugira uma abordagem para isolar o defeito.

▶ **FIGURA 13–43**



Solução O DAC nesse caso não é monotônico. Uma análise da saída revela que o dispositivo está convertendo a seguinte seqüência, em vez da seqüência binária real que é aplicada às entradas.

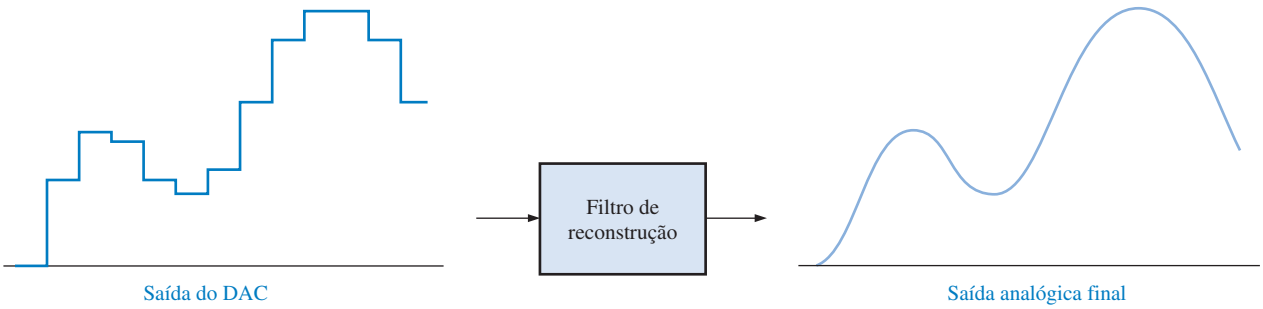
0010, 0011, 0010, 0011, 0110, 0111, 0110, 0111, 1010, 1011, 1010, 1011, 1110, 1111, 1110, 1111

Aparentemente, o bit 2^1 está preso no estado ALTO (1). Para encontrar o problema, primeiro monitore o pino de entrada do bit no dispositivo. Se ele muda de estado, o defeito é interno ao DAC e o CI deve ser substituído. Se o pino externo não muda de estado e está sempre em nível ALTO, verifique um curto-circuito externo para +V que pode ser causado por uma ponte de solda em algum lugar da placa de circuito.

Problema relacionado Determine a saída de um DAC quando uma seqüência binária direta de 4 bits é aplicada às entradas e o bit 2^0 está preso em nível ALTO.

Filtro de Reconstrução

A saída de um DAC é uma aproximação, na forma de uma "escada", do sinal analógico original após ser processado por um DSP. A finalidade do filtro de reconstrução passa-baixas (algumas vezes denominado de pós-filtro) é para "suavizar" a saída do DAC eliminando o conteúdo de alta freqüência que resulta das transições dos "degraus da escada", como aproximadamente ilustrado na Figura 13–44.



▲ **FIGURA 13–44**

O filtro de reconstrução "suaviza" a saída do DAC.

SEÇÃO 13–5 REVISÃO

1. Qual a desvantagem do DAC com entradas binárias ponderadas?
2. Qual a resolução de um DAC de 4 bits?
3. Como você detecta a falta de monotonicidade no comportamento de um DAC?
4. Qual o efeito que um ganho baixo tem na saída de um DAC?

RESUMO

- O processamento de sinais digitais é o processamento digital de sinais analógicos, geralmente em tempo real, com a finalidade de modificar ou melhorar o sinal de alguma forma.
- Em geral, o processamento de sinais digitais consiste de um filtro *anti-aliasing*, um circuito de amostragem e retenção, um conversor analógico-digital, um DSP (processador de sinais digitais) e um filtro de reconstrução.
- A amostragem converte um sinal analógico em uma série de impulsos, cada um representando a amplitude do sinal analógico num dado instante de tempo.
- O teorema da amostragem diz que a freqüência de amostragem tem que ser pelo menos duas vezes a maior freqüência amostrada (freqüência de Nyquist).
- A conversão analógico-digital transforma um sinal analógico em uma série de códigos digitais.
- Quatro tipos de conversores analógico-digital (DACs) são o flash (simultâneo), dupla rampa, aproximação sucessiva e sigma-delta.
- Um DSP é um microprocessador especializado otimizado em termos de velocidade para processar dados à medida que são gerados (tempo real).
- A maioria dos DSPs é baseada na arquitetura Harvard, na qual existem uma memória de dados e uma memória de programa.
- Uma operação *pipeline* consiste de estágios busca, decodificação e execução.
- A conversão digital-analógico transforma uma série de códigos digitais, que representam um sinal analógico, de volta para o formato analógico.
- Dois tipos de conversores digital-analógico (DACs) são o de entrada ponderada, no valor dos pesos binários, e o de escada *R*/2*R*.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Aliasing O efeito criado quando um sinal é amostrado com pelo menos duas vezes a freqüência do sinal. O *aliasing* cria freqüências indesejadas que interferem com a freqüência do sinal

Amostragem O processo de obtenção de um número suficiente de valores discretos em pontos da forma de onda que definem o formato da forma de onda.

Busca Um estágio da operação *pipeline* do DSP na qual as instruções são obtidas a partir da memória de programa.

Conversor analógico-digital (ADC) Um circuito usado para converter um sinal analógico para o formato digital.

Conversor digital-analógico (DAC) Um circuito usado para converter a representação digital de um sinal analógico de volta para o formato analógico.

Decodificação Um estágio da operação *pipeline* do DSP no qual as instruções são associadas a unidades funcionais e são decodificadas.

DSP Processador de sinais digitais; um tipo especial de microprocessador que processa dados em tempo real.

Execução Um estágio da operação *pipeline* do DSP na qual as instruções decodificadas são executadas.

Freqüência de Nyquist O maior sinal de freqüência que pode ser amostrado para uma freqüência de amostragem especificada; uma freqüência igual ou menor que metade da freqüência de amostragem.

MFLOPS Milhões de operações de ponto flutuante por segundo.

MIPS Milhões de instruções por segundo.

MMACS Milhões de multiplicações/acumulações por segundo.

Núcleo de DSP A unidade central de processamento de um DSP.

Pipeline Parte da arquitetura de um DSP a qual permite que múltiplas instruções sejam processadas simultaneamente.

Quantização O processo por meio do qual um código binário é associado a cada valor amostrado durante a conversão analógico digital.

AUTOTESTE

As respostas estão no final do capítulo.

1. Um ADC é um
(a) codificador de dados alfanumérico **(b)** conversor analógico-digital
(c) portador de dispositivo analógico **(d)** comparador analógico-digital

2. Um DAC é um
(a) computador digital-analógico **(b)** calculadora de análise digital
(c) conversor de acumulação de dados **(d)** conversor digital-analógico

3. Um sistema de processamento digital de sinais opera geralmente
(a) em tempo real **(b)** em tempo imaginário
(c) em tempo comprimido **(d)** no tempo do computador

4. A amostragem de um sinal analógico produz
(a) uma série de impulsos que são proporcionais à amplitude do sinal.
(b) uma série de impulsos que são proporcionais à freqüência do sinal.
(c) códigos digitais que representam a amplitude do sinal analógico.
(d) códigos digitais que representam o instante de cada amostra.

5. De acordo com o teorema da amostragem, a freqüência de amostragem deve ser
(a) menor que metade da maior freqüência do sinal.
(b) maior que duas vezes a maior freqüência do sinal.
(c) menor que metade da menor freqüência do sinal.
(d) maior que a menor freqüência do sinal.

6. Uma ação de retenção ocorre
(a) antes de cada amostra **(b)** durante cada amostra
(c) após a conversão analógico digital **(d)** imediatamente após uma amostra

7. O processo de quantização
(a) converte a saída do circuito de amostragem e retenção para código binário.
(b) converte um impulso da amostragem em um nível.
(c) converte uma seqüência de códigos binários num sinal analógico reconstruído.
(d) filtra (elimina) as freqüências indesejadas antes da amostragem.

8. Geralmente, um sinal analógico pode ser reconstruído com mais precisão tendo
(a) mais níveis de quantização.
(b) poucos níveis de quantização.
(c) uma maior freqüência de amostragem.
(d) uma menor freqüência de amostragem.
(e) as alternativas (a) e (c) estão corretas.

9. Um ADC flash usa
(a) contadores **(b)** amp-ops
(c) um integrador **(d)** flip-flops
(e) as alternativas (a) e (c) estão corretas.º

10. Um ADC de dupla rampa usa
(a) contadores **(b)** amp-ops
(c) um integrador **(d)** um diferenciador

11. A saída de um ADC sigma-delta é (são)
(a) códigos binários em paralelo **(b)** dados de múltiplos bits
(c) um dado de um único bit **(d)** uma tensão de diferença

12. O termo *arquitetura Harvard* significa.
(a) uma CPU e uma memória principal.
(b) uma CPU e duas memórias de dados.
(c) uma CPU, uma memória de programa e uma memória de dados.
(d) uma CPU e dois arquivos de registradores.

13. O número mínimo de registradores de propósito geral em um CI DSP da série TMS320C6000 é
(a) 32 **(b)** 64
(c) 16 **(d)** 8

14. As duas memórias internas em um CI da série TMS320C6000 têm cada uma a capacidade de
(a) 1 MB **(b)** 512 kB
(c) 64 kB **(d)** 32 kB

15. O número de instruções processadas simultaneamente na operação *pipeline* da série TMS320C6000 é
(a) oito **(b)** quatro
(c) dois **(d)** um

16. O estágio da operação *pipeline* na qual as instruções são recuperadas da memória é denominado de
(a) execução **(b)** acumulação
(c) decodificação **(d)** busca

17. Os resistores nas entradas de um DAC com ponderação binária
(a) determinam a amplitude do sinal analógico
(b) determinam os pesos das entradas digitais
(c) limitam o consumo de potência
(d) atenua o efeito de carga para a fonte

18. Em um DAC R/2R existem
(a) quatro valores de resistores
(b) um valor de resistor
(c) dois valores de resistores
(d) um número de valores de resistores igual ao número de entradas

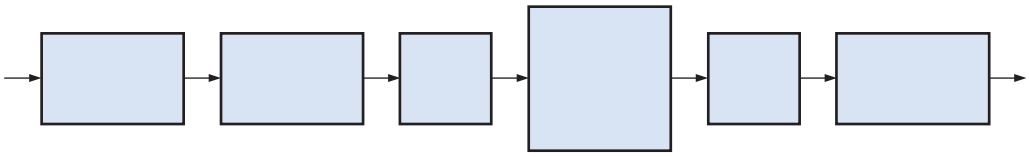
PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 13–1 Fundamentos de Processamento de Sinais Digitais

1. Explique a finalidade da conversão analógico-digital

2. Preencha com o nome funcional apropriado os blocos do diagrama de um sistema de processamento de sinais digitais mostrado na Figura 13–45.

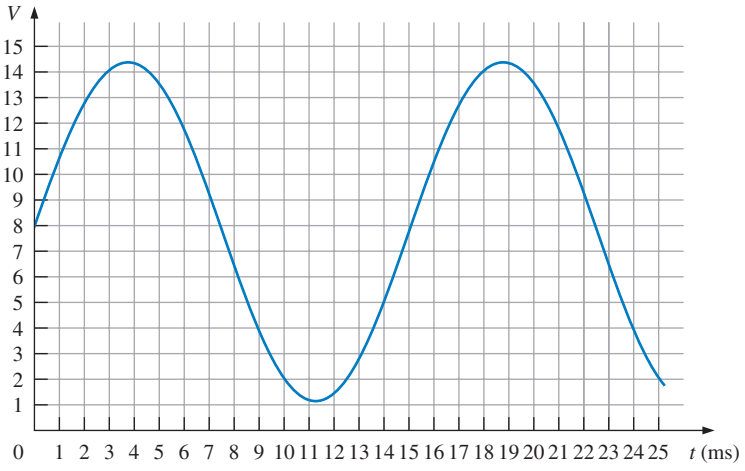


▲ **FIGURA 13–45**

3. Explique a finalidade da conversão digital-analógica

SEÇÃO 13–2 Conversão de Sinal Analógico para Digital

4. A forma de onda mostrada na Figura 13–46 é aplicada em um circuito de amostragem sendo amostrada a cada 3 ms. Mostre a forma de onda de saída do circuito de amostragem. Considere a uma correspondência de tensão individual entre a entrada e a saída.



▲ **FIGURA 13–46**

5. A saída do circuito de amostragem no Problema 4 é aplicada a um circuito de retenção. Mostre a forma de onda de saída desse circuito.

6. Se a saída do circuito no Problema 5 for quantizada usando dois bits, qual a seqüência resultante de códigos binários?

7. Repita o Problema 6 usando quantização de 4 bits.

8. **(a)** Faça a reconstrução do sinal analógico a partir da quantização de 2 bits do Problema 6.
(b) Faça a reconstrução do sinal analógico a partir da quantização de 4 bits do Problema 7.

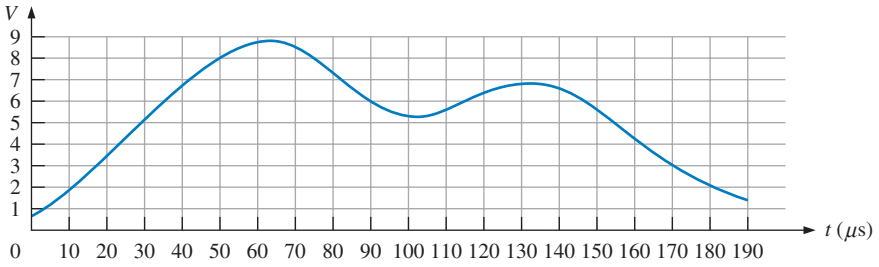
9. Faça o gráfico de uma função analógica representada pela seguinte seqüência de números binários:
1111, 1110, 1101, 1100, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1100, 1100, 1011, 1010, 1001

SEÇÃO 13–3 Métodos de Conversão Analógico-Digital

10. A tensão de entrada em um certo amplificador inversor implementado com amp-op é 10 mV e a saída é 2 V. Qual é o ganho de tensão de malha fechada?

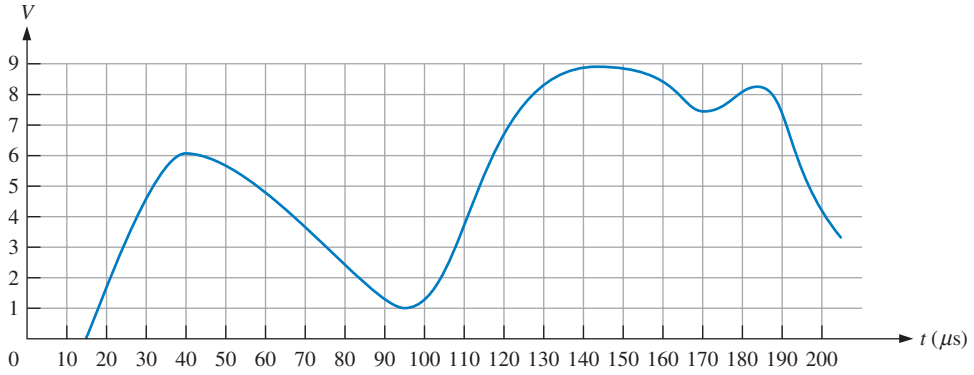
11. Para conseguir um ganho de tensão em malha fechada de 330 com um amplificador inversor, qual o valor do resistor de realimentação se $R_i = 1\ \text{k}\Omega$?

12. Determine o código de saída em binário de um ADC flash de 3 bits para o sinal analógico de entrada mostrado na Figura 13–47.



▶ **FIGURA 13–47**

13. Repita o Problema 12 para a forma de onda analógica mostrada na Figura 13–48.



▶ **FIGURA 13–48**

14. Para um certo ADC de aproximação sucessiva de 2 bits, a saída em escada máxima é +8 V. Se uma tensão constante de +6 V for aplicada na entrada analógica, determine a seqüência de estados binários para o SAR.

15. Repita o Problema 14 para um ADC de aproximação sucessiva de 4 bits.

16. Um ADC produz a seguinte seqüência de números em binário quando um sinal analógico é aplicado na sua entrada: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000.

(a) Faça a reconstrução da entrada digital.

(b) Se o ADC apresentar um defeito de forma que o código 0111 esteja ausente, como seria a saída reconstruída?

SEÇÃO 13–4 Processador de Sinais Digitais (DSP)

17. Um DSP TMS320C62xx tem 32 bits de instrução e opera com 2000 MIPS. Quantos bytes por segundo esse DSP processa?

18. Se a taxa de clock de um TMS320C64xx é 400 MHz, quantas instruções podem ser fornecidas às unidades funcionais da CPU em um segundo?

19. Quantas operações de ponto flutuante um DSP pode realizar em um segundo se ele é especificado para 1000 MFLOPS.

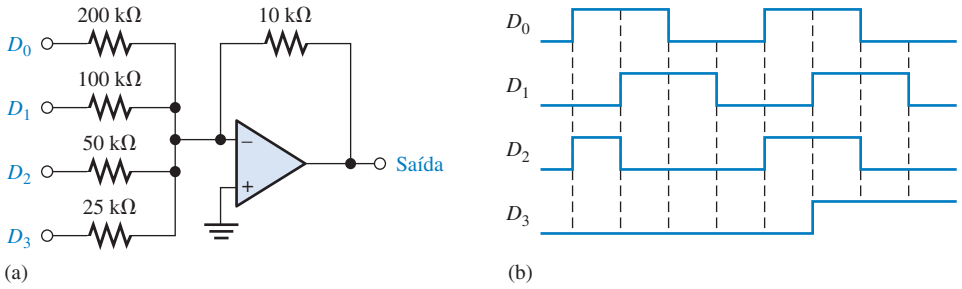
20. Liste e descreva as quatro fases da operação de busca em um DSP da série TMS320C6000.

21. Liste e descreva as duas fases da operação de decodificação em um DSP da série TMS320C6000.

SEÇÃO 13–5 Métodos de Conversão Digital-Analógico

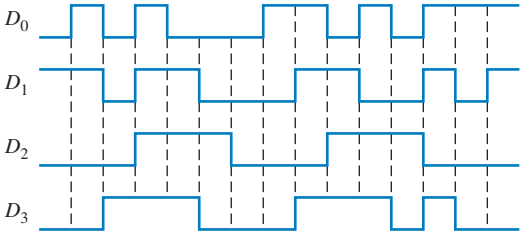
22. No DAC de 4 bits mostrado na Figura 13–36, o resistor de menor peso tem um valor de 10 kΩ. Quais devem ser os valores dos outros resistores de entrada?

23. Determine a saída do DAC mostrado na Figura 13–49(a) se a seqüência de números de 4 bits, vista na parte (b) da figura, for aplicada nas entradas. As entradas de dados têm um valor de 0 V para nível BAIXO e +5 V para nível ALTO.



▶ FIGURA 13–49

24. Repita o Problema 23 para a entrada dada na Figura 13–50.



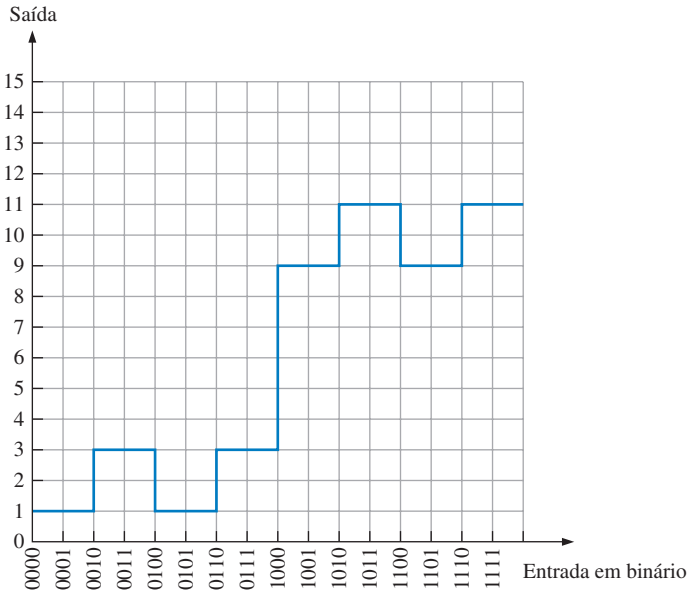
▶ FIGURA 13–50

25. Para cada um dos seguintes DACs determine a solução expressa como uma porcentagem.
(a) 3-bit **(b)** 10-bit **(c)** 18-bit

26. Desenvolva um circuito para gerar uma seqüência binária com 8 bits para a configuração de teste mostrada na Figura 13–41.

27. Um DAC de 4 bits apresenta um defeito de forma que o MSB está preso no estado 0. Desenhe a saída analógica quando uma seqüência binária direta for aplicada às entradas.

28. Uma seqüência binária direta é aplicada em um DAC de 4 bits, sendo observada a saída mostrada na Figura 13–51. Qual é o problema?



▶ FIGURA 13–51

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 13–1 Fundamentos de Processamento de Sinais Digitais

1. DSP significa processador de sinais digitais
2. ADC significa conversor analógico-digital
3. DAC significa conversor digital-analógico
4. O ADC transforma um sinal analógico para a forma binária codificada.
5. O DAC transforma um sinal codificado em binário para a forma analógica.

SEÇÃO 13–2 Conversão de Sinal Analógico para Digital

1. A amostragem é o processo de conversão de um sinal analógico em uma série de impulsos, cada um representando a amplitude do sinal analógico.
2. Um valor amostrado é retido para proporcionar um tempo para a conversão dele em um código binário.
3. A freqüência de amostragem mínima é 40 kHz.
4. Quantização é o processo de conversão de um nível amostrado em um código binário.
5. O número de bits determina a precisão da quantização.

SEÇÃO 13–3 Métodos de Conversão Analógico-Digital

1. O método simultâneo (flash) é mais rápido.
2. O método sigma-delta produz um fluxo de dados de um único bit.
3. Sim, a aproximação sucessiva tem um tempo de conversão fixo.
4. A ausência de um código, um código incorreto e um *offset* são tipos de erros na saída de um ADC.

SEÇÃO 13–4 Processador de Sinais Digitais (DSP)

1. A arquitetura Harvard significa que existe uma CPU e duas memórias, uma para dados e outra para programa.
2. O núcleo de um DSP é a CPU.
3. Os DSPs podem ser de ponto fixo ou ponto flutuante.
4. Os tipos de memória interna são as de dados e programa.
5. **(a)** MIPS – milhões de instruções por segundo
   (b) MFLOPS – milhões de operações de ponto flutuante por segundo
   (c) MMACS – milhões de multiplicações/acumulações por segundo
6. O *pipeline* provê o processamento de múltiplas instruções simultaneamente.
7. Os estágios da operação *pipeline* são busca, decodificação e execução.
8. Durante a busca, as instruções são recuperadas a partir da memória de programa.

SEÇÃO 13–5 Métodos de Conversão Digital-Analógico

1. Em um DAC ponderado, cada resistor tem um valor diferente.
2. $(1/(2^4 - 1))100\% = 6{,}67\%$
3. Um degrau invertido indica falta de monotonicidade em um DAC.
4. Com um ganho baixo, as amplitudes dos degraus em um DAC são menores que o ideal.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

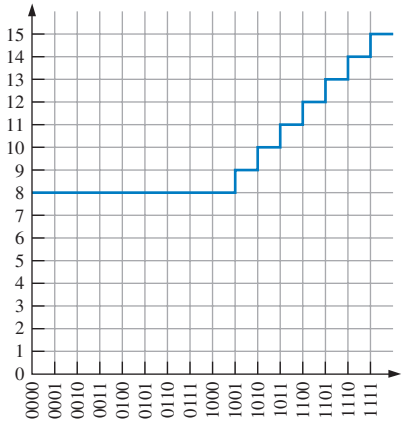
13–1. 100, 111, 100, 011, 110. Sim, uma informação é perdida.

13–2. Veja a Figura 13–52.

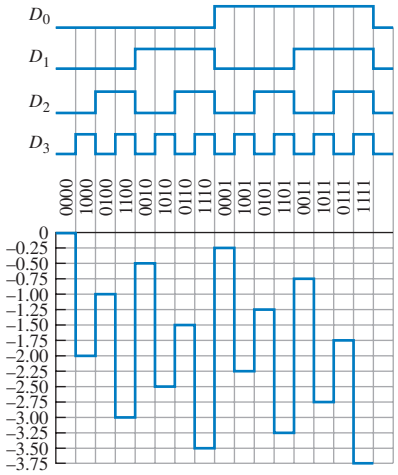
13–3. Veja a Figura 13–53.

13–4. $(1/(216 - 1))100\% = 0{,}00153\%$

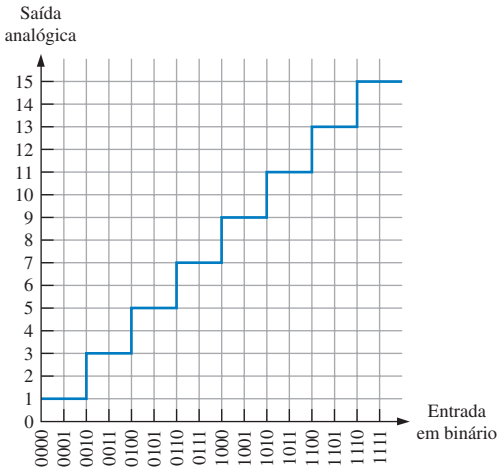
13–5. Veja a Figura 13–54.



▲ FIGURA 13–52



▲ FIGURA 13–53



▲ FIGURA 13–54

AUTOTESTE

1.	(b)	**2.**	(d)	**3.**	(a)	**4.**	(a)	**5.**	(b)	**6.**	(d)
7.	(a)	**8.**	(e)	**9.**	(b)	**10.**	(e)	**11.**	(c)	**12.**	(c)
13.	(a)	**14.**	(c)	**15.**	(a)	**16.**	(d)	**17.**	(b)	**18.**	(c)

Referências

Dahnoun, Naim. *Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform.* Reading, Mass.: Addison-Wesley Longman. 2000.

Hayes, Monson. *Schaum's Outline of Digital Signal Processing.* New York: McGraw-Hill. 1998.

Kuo, Sen, and Bob Lee. *Real-Time Digital Signal Processing: Implementations, Applications, and Experiments with the TMS320C55x.* New York: John Wiley & Sons. 2001.

Lyons, Richard. *Understanding Digital Signal Processing.* Reading, Mass.: Addison-Wesley Longman. 1996.

Marven, Craig, and Gillian Ewers. *A Simple Approach to Digital Signal Processing.* New York: John Wiley & Sons. 1996.

Oppenheim, Alan, and Ronald Schafer. *Digital Signal Processing.* Englewood Cliffs, N.J.: PrenticeHall. 1974.

Orfanidis, Sophocles. *Introduction to Signal Processing.* Upper Saddle River, N.J.: PrenticeHall. 1996.

Proakis, John, and Dimitris Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications,* 3d ed. Upper Saddle River, N.J.: Prentice-Hall. 1996.

Steiglitz, Ken. *Digital Signal Processing Primer: With Applications to Digital Audio and Computer* Music. Reading, Mass.: Addison-Wesley Longman. 1996.

Williams, Douglas, and Vijay Madisetti. *Digital Signal Processing Handbook.* Boca Raton, Fl.: CRC Press. 1997.

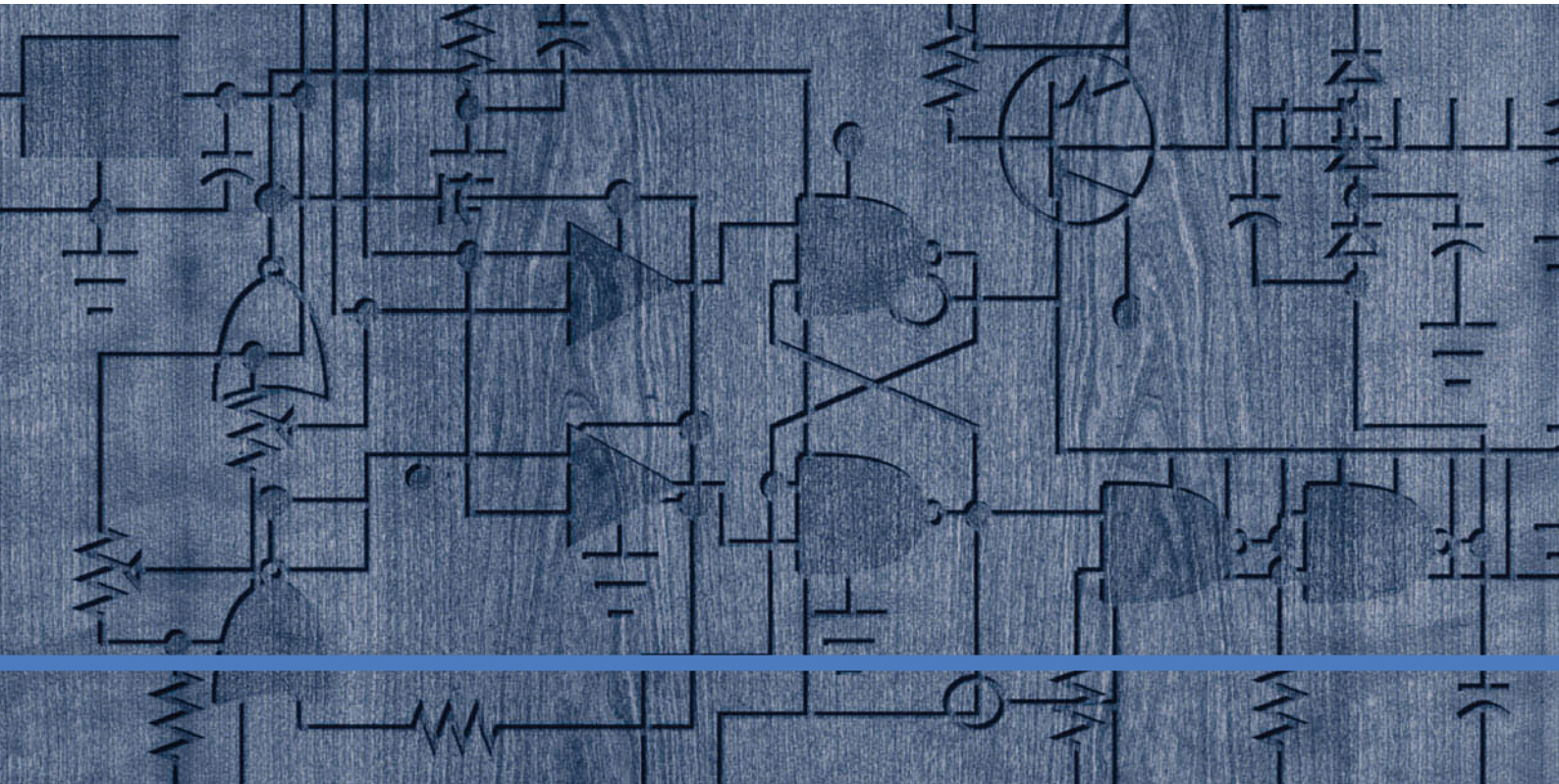
14

TECNOLOGIAS DE CIRCUITOS INTEGRADOS

TÓPICOS DO CAPÍTULO

OBJETIVOS DO CAPÍTULO

- Determinar a margem de ruído de um dispositivo a partir dos parâmetros fornecidos pelas folhas de dados
- Calcular a dissipação de potência de um dispositivo
- Explicar como o tempo de atraso de propagação afeta a freqüência de operação ou a velocidade de um circuito
- Interpretar o produto velocidade-potência como uma medida de desempenho
- Usar folhas de dados para obter informação a respeito de um determinado dispositivo
- Explicar o que significa o fan-out de uma porta



- Descrever como funciona as portas lógicas básicas TTL e CMOS em nível de componente
- Reconhecer a diferença entre saídas TTL, totem-pole e coletor aberto e entender as limitações no uso de cada uma delas
- Conectar circuitos na configuração de lógica AND com fios (*wired-AND*)
- Descrever a operação de circuitos tristate
- Fazer a terminação correta nas entradas não usadas das portas lógicas
- Comparar o desempenho das famílias TTL e CMOS
- Manusear dispositivos CMOS sem riscos de danificá-los devido à eletricidade estática
- Enunciar as vantagens da família ECL
- Descrever os circuitos PMOS e NMOS
- Descrever uma célula E^2CMOS

TERMOS IMPORTANTES

- TTL
- CMOS
- Imunidade a ruído
- Margem de ruído
- Dissipação de potência
- Tempo de atraso de propagação
- Fan-out
- Fornecimento de corrente
- Absorção de corrente
- Carga unitária
- Resistor de pull-up
- Tristate
- Totem-pole
- Coletor aberto
- ECL
- E^2CMOS

INTRODUÇÃO

Esse capítulo foi idealizado para ser estudado em qualquer momento, podendo ser abordado totalmente, em parte ou até mesmo suprimido, dependendo dos objetivos do curso. A Seção 3–8 deve ser abordada antes do estudo desse capítulo.

No Capítulo 3 (Seção 3–8), aprendemos sobre portas lógicas básicas em circuito integrado. Esse capítulo fornece uma introdução à tecnologia de circuito usada para implementar as portas lógicas estudadas, bem como outros tipos de dispositivos na forma de CI.

As duas principais tecnologia de CIs, CMOS e TTL, são abordadas sendo seus parâmetros de operação definidos. Além disso, são comparadas as características operacionais de diversas famílias dentro dessas tecnologias de circuito. Outras tecnologias de circuito também são apresentadas. É importante ter em mente que uma tecnologia de circuito em particular usada para implementar uma porta lógica não tem efeito na operação lógica da porta. Em termos da operação da tabela-verdade, um certo tipo de porta que é implementada com CMOS tem o mesmo comportamento lógico quando implementado com TTL. A única diferença nas portas é as características elétricas tais como a dissipação de potência, velocidade de comutação e imunidade a ruído.

WWW. ACESSE O SITE

Recursos que o ajudarão no estudo deste capítulo estão disponíveis em

http://www.prenhall.com/floyd

14-1 CARACTERÍSTICAS E PARÂMETROS OPERACIONAIS BÁSICOS

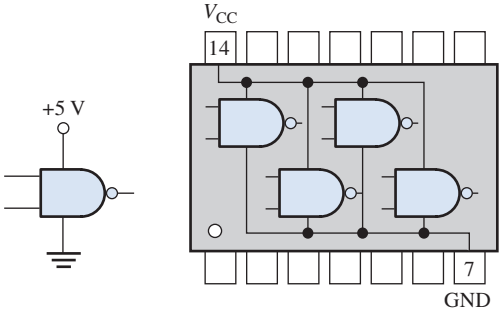
Quando trabalhamos com CIs digitais, devemos nos familiarizar não somente com as suas operações lógicas, mas também com propriedades operacionais tais como níveis de tensão, imunidade a ruído, dissipação de potência, fan-out e tempo de atraso de propagação. Nesta seção, discutiremos os aspectos práticos dessas propriedades.

Ao final do estudo desta seção você deverá ser capaz de:

■ Determinar as conexões de +V e GND ■ Descrever os níveis lógicos para CMOS e TTL ■ Discutir a imunidade a ruído ■ Determinar a dissipação de potência de um circuito lógico ■ Definir o tempo de atraso de propagação de uma porta lógica ■ Discutir o produto velocidade-potência e explicar o seu significado ■ Discutir a carga e o fan-out de TTL e CMOS

Tensão de Alimentação CC

O valor nominal da tensão de alimentação cc para dispositivos **TTL** (lógica transistor-transistor) é +5 V. TTL também é designado por T^2L. Os dispositivos **CMOS** (semicondutor de óxido metálico complementar) são comercializados com diferentes categorias de tensão de alimentação: +5 V, +3,3 V, 2,5 V e 1,2 V. Embora omitido do diagrama por questão de simplificação, a tensão de alimentação +V é conectada no pino V_{CC} do CI e o terra é conectado no pino GND. A tensão de alimentação +V e o GND são distribuídos internamente para todos os elementos dentro do encapsulamento, como ilustra a Figura 14–1 para um encapsulamento de 14 pinos.



(a) Uma única porta (b) Encapsulamento DIP

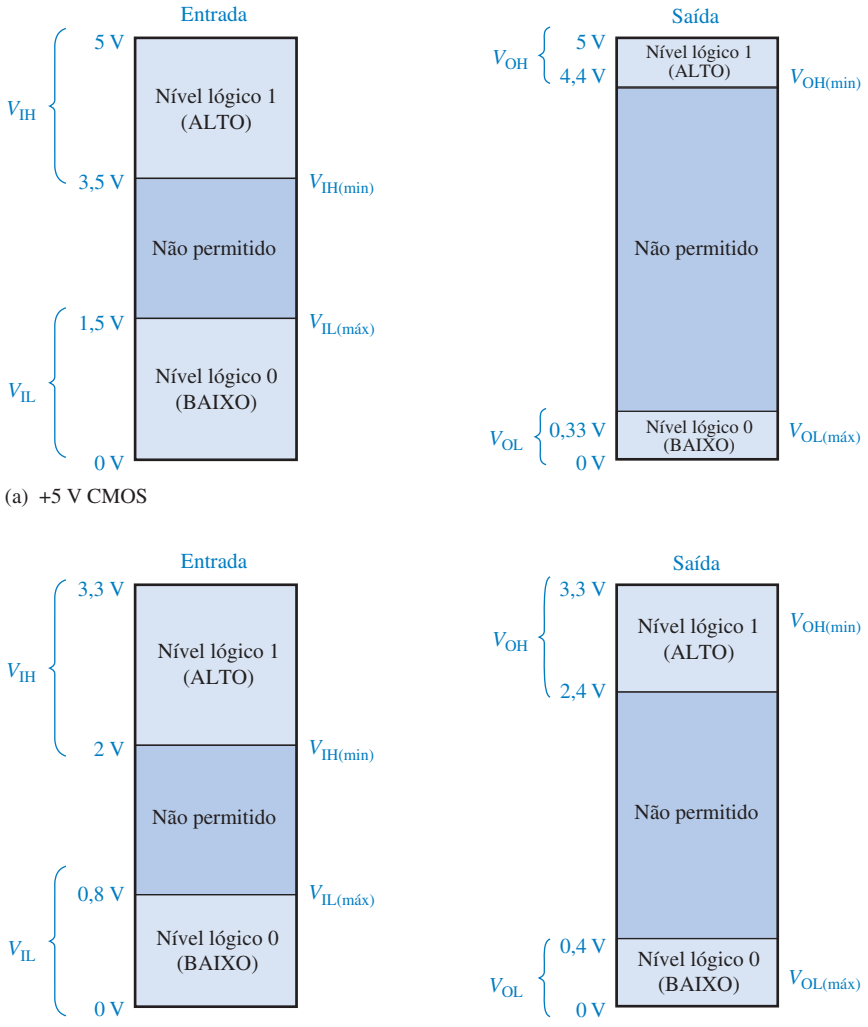
► **FIGURA 14–1**

Exemplo de conexão de V_{CC} e terra e a distribuição destes no encapsulamento do CI. Os outros pinos de conexão foram suprimidos por questão de simplificação.

Níveis Lógicos CMOS

Os níveis lógicos foram discutidos brevemente no Capítulo 1. Existem quatro especificações diferentes de níveis lógicos: V_{IL}, V_{IH}, V_{OL} e V_{OH}. Para circuitos CMOS, as variações das tensões de entrada (V_{IL}) que podem representar um nível BAIXO válido (nível lógico 0) são de 0 V a 1,5 V para uma alimentação de +5 V e de 0 V a 0,8 V para uma alimentação de +3,3 V. A faixa de tensão de entrada (V_{IH}) que pode representar um nível ALTO válido (nível lógico 1) é de 3,5 V a 5 V para uma alimentação de +5 V e de 2 V a 3,3 V para uma alimentação de +3,3 V, conforme indicado na Figura 14–2. A faixa de valores de 1,5 V a 3,5 V para uma alimentação de +5 V e de 0,8 V a 2 V para uma alimentação de +3,3 V são regiões de resposta imprevisível, sendo os valores dessas faixas não permitidos. Quando uma tensão de entrada está em uma dessas faixas, ela pode ser interpretada como nível BAIXO ou ALTO pelo circuito. Portanto, as portas CMOS não podem operar confiavelmente quando as tensões de entrada estão nessas faixas não permitidas.

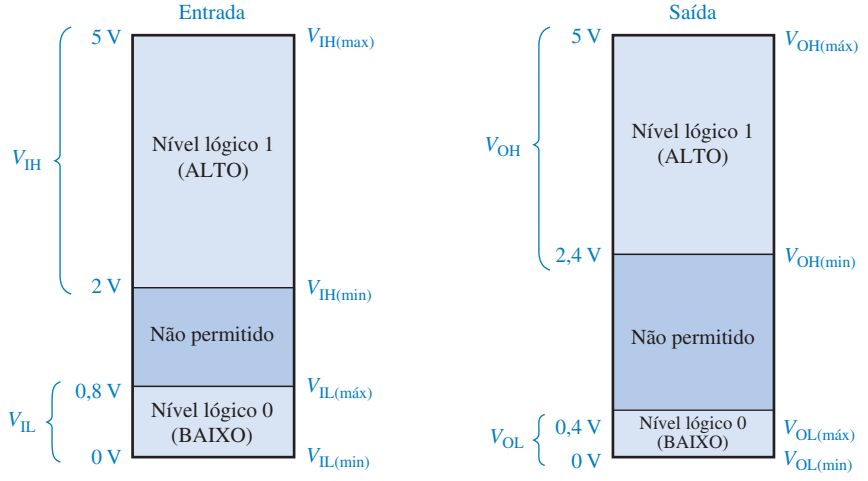
As faixas de tensões de saída de CMOS (V_{OL} e V_{OH}) tanto para uma alimentação de 5 V quanto de 3,3 V são mostradas na Figura 14–2. Observe que a tensão de saída de nível ALTO mínima, $V_{OH(mín)}$, é maior que a tensão de entrada de nível ALTO mínima, $V_{IH(mín)}$. Além disso, observe que a tensão de saída de nível BAIXO máxima, $V_{OL(máx)}$, é menor que a tensão de entrada de nível BAIXO máxima, $V_{IL(máx)}$.



◀ **FIGURA 14–2**
Níveis lógicos de entrada e saída para CMOS.

Níveis Lógicos TTL

Os níveis lógicos de entrada e saída para TTL são dados na Figura 14–3. Assim como para CMOS, existem quatro especificações diferentes de níveis lógicos: V_{IL}, V_{IH}, V_{OL} e V_{OH}.

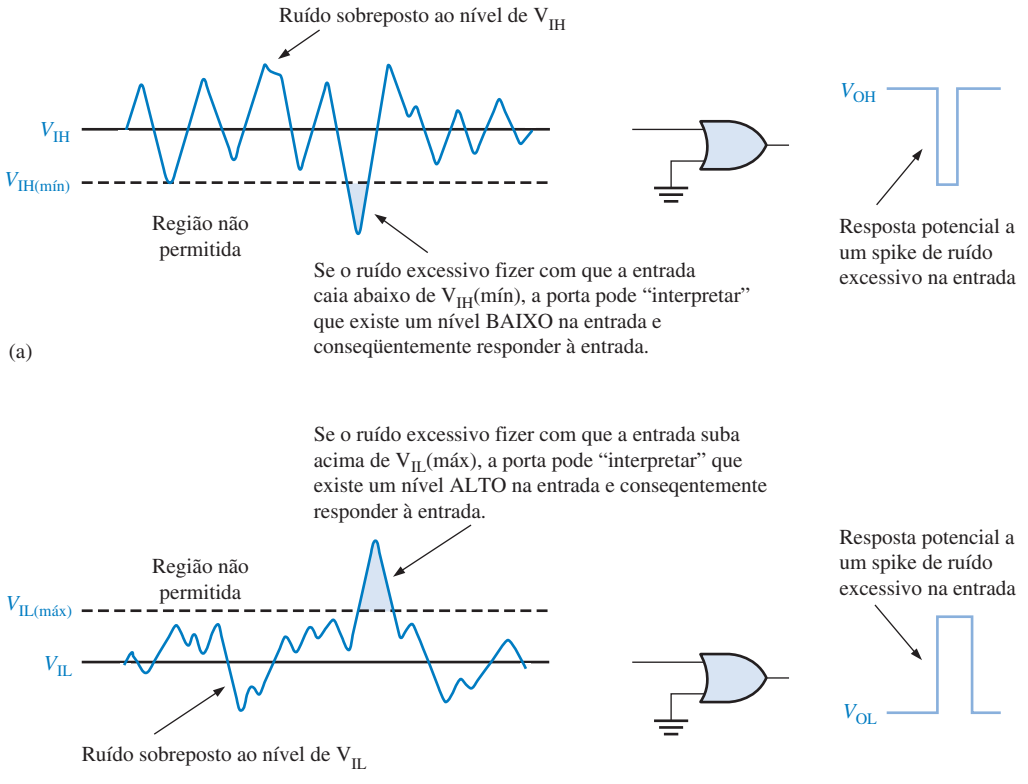


◀ **FIGURA 14–3**
Níveis lógicos de entrada e saída para TTL.

Imunidade a Ruído

Ruído é uma tensão indesejada que é induzida em circuitos elétricos e pode apresentar uma ameaça ao funcionamento correto de um circuito. Os fios e outros condutores em um sistema podem captar radiação eletromagnética de alta freqüência de condutores adjacentes nos quais as correntes variam rapidamente ou a partir de muitas outras fontes externas ao sistema. Além disso, as flutuações nas linhas de alimentação são uma forma de ruído de baixa freqüência.

Para não ser adversamente afetado pelo ruído, um circuito lógico tem que ter uma certa **imunidade ao ruído**. Essa característica representa a capacidade do circuito de tolerar uma certa quantidade de flutuações indesejadas nas tensões de entrada sem alterar o seu estado de saída. Por exemplo, se uma tensão de ruído fizer com que uma entrada de 5 V de uma porta CMOS caia abaixo de 3,5 V no estado ALTO, a entrada entra na região não permitida, sendo a resposta imprevisível (veja a Figura 14–2). Portanto, a porta pode interpretar a flutuação abaixo de 3,5 V como um nível BAIXO, conforme ilustrado na Figura 14–4(a). De forma similar, se o ruído fizer com que a tensão de entrada de uma porta suba para um valor acima de 1,5 V no estado BAIXO, uma condição incerta é gerada, como ilustra a parte (b) da figura.



▲ **FIGURA 14–4**

Ilustração dos efeitos do ruído de entrada na operação de uma porta.

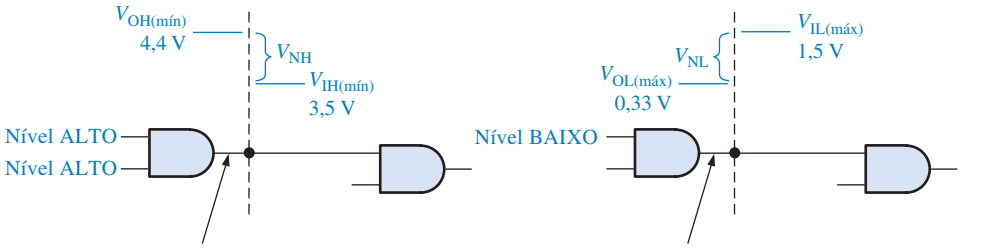
Margem de Ruído

Uma medida da imunidade ao ruído de um circuito é denominada de **margem de ruído**, a qual é expressa em volts. Existem dois valores de margem de ruído especificados para um determinado circuito lógico: a margem de ruído (*noise*) de nível ALTO (V_{NH}) e a margem de ruído de nível BAIXO (V_{NL}). Esses parâmetros são definidos pelas seguintes equações:

Equação 14–1
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)}$$

Equação 14–2
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)}$$

Algumas vezes podemos encontrar a margem de ruído expressa como uma porcentagem de V_{CC}. A partir de equações, V_{NH} é a diferença entre a menor tensão de saída em nível ALTO possível de uma porta acionadora ($V_{OH(mín)}$) e a menor tensão de entrada em nível ALTO de uma porta acionada ($V_{IH(mín)}$). A margem de ruído, V_{NL}, é a diferença entre a máxima tensão de entrada que uma porta acionada pode tolerar em nível BAIXO ($V_{IL(máx)}$) e a máxima tensão possível de saída em nível BAIXO de uma porta acionadora ($V_{OL(máx)}$). As margens de ruído estão ilustradas na Figura 14–5.



A tensão na linha nunca será menor que 4,4 V a menos que um ruído ou uma operação incorreta aconteça.

(a) Margem de ruído de nível ALTO

A tensão na linha nunca excederá a 0,33 V a menos que um ruído ou uma operação incorreta aconteça.

(b) Margem de ruído de nível BAIXO

▲ **FIGURA 14–5**

Ilustração das margens de ruído. Os valores são para CMOS operando com 5 V, mas o princípio se aplica a qualquer família lógica.

EXEMPLO 14–1

Determine as margens de ruído dos níveis ALTO e BAIXO para CMOS e TTL usando as informações dadas nas Figuras 14–2 e 14–3.

Solução Para CMOS operando com 5 V,

$$V_{IH(mín)} = 3{,}5\text{ V}$$
$$V_{IL(máx)} = 1{,}5\text{ V}$$
$$V_{OH(mín)} = 4{,}4\text{ V}$$
$$V_{OL(máx)} = 0{,}33\text{ V}$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 4{,}4\text{ V} - 3{,}5\text{ V} = \mathbf{0{,}9\text{ V}}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 1{,}5\text{ V} - 0{,}33\text{ V} = \mathbf{1{,}17\text{ V}}$$

Para TTL,

$$V_{IH(mín)} = 2\text{ V}$$
$$V_{IL(máx)} = 0{,}8\text{ V}$$
$$V_{OH(mín)} = 2{,}4\text{ V}$$
$$V_{OL(máx)} = 0{,}4\text{ V}$$
$$V_{NH} = V_{OH(mín)} - V_{IH(mín)} = 2{,}4\text{ V} - 2\text{ V} = \mathbf{0{,}4\text{ V}}$$
$$V_{NL} = V_{IL(máx)} - V_{OL(máx)} = 0{,}8\text{ V} - 0{,}4\text{ V} = \mathbf{0{,}4\text{ V}}$$

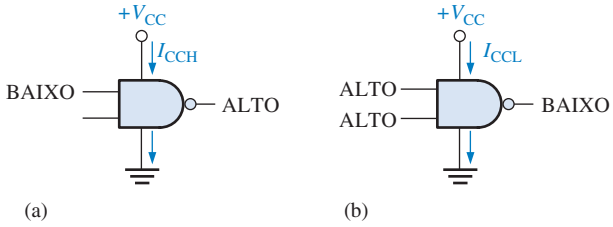
Uma porta TTL é imune a um ruído de até 0,4 V para os estados de entrada ALTO ou BAIXO.

Problema relacionado* Baseado nos cálculos de margem de ruído, qual família de dispositivos, CMOS operando em 5 V ou TTL, deve ser usada um ambientes com alto ruído?

* As respostas estão no final do capítulo.

Dissipação de Potência

Uma porta lógica drena corrente da fonte de alimentação cc, conforme indicado na Figura 14–6. Quando a porta está com a saída no estado ALTO, uma certa quantidade de corrente denominada de I_{CCH} é drenada; e quando a saída está no estado BAIXO, uma quantidade diferente de corrente, I_{CCL}, é drenada.



▶ FIGURA 14–6

Correntes provenientes da fonte de alimentação. É mostrado a direção convencional da corrente. A notação de fluxo de elétrons é oposta.

Como um exemplo, se I_{CCH} for especificado com 1,5 mA quando V_{CC} for 5 V e se a porta estiver estática (sem comutar) no estado de saída ALTO, a **dissipação de potência** (P_D) da porta é

$$P_D = V_{CC}I_{CCH} = (5\text{ V})(1{,}5\text{ mA}) = 7{,}5\text{ mW}$$

Quando uma porta é pulsada, sua saída comuta entre os níveis ALTO e BAIXO, sendo que a quantidade de corrente de alimentação varia entre I_{CCH} e I_{CCL}. A dissipação de potência média depende do ciclo de trabalho e é geralmente especificada para um ciclo de trabalho de 50%. Quando o ciclo de trabalho é 50%, a saída está metade do tempo em nível ALTO e metade do tempo em nível BAIXO. A corrente de alimentação média é portanto

Equação 14–3

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2}$$

A dissipação média de potência é

Equação 14–4

$$P_D = V_{CC}I_{CC}$$

EXEMPLO 14–2

Uma certa porta drena 2μA quando sua saída é nível ALTO e 3,6μA quando sua saída é nível BAIXO. Qual a dissipação média de potência se V_{CC} é 5 V e a porta está operando com um ciclo de trabalho de 50%?

Solução O I_{CC} médio é

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{2{,}0\ \mu\text{A} + 3{,}6\ \mu\text{A}}{2} = 2{,}8\ \mu\text{A}$$

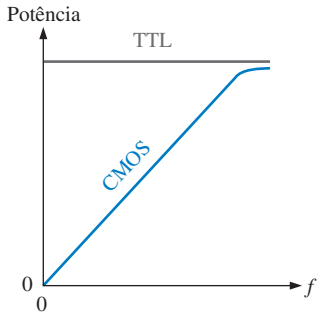
A dissipação média de potência é

$$P_D = V_{CC}I_{CC} = (5\text{ V})(2{,}8\ \mu\text{A}) = \mathbf{14\ \mu W}$$

Problema relacionado Um certo CI de portas lógicas tem um $I_{CCH} = 1{,}5\mu$A e $I_{CCL} = 2{,}8\mu$A. Determine a dissipação média de potência para um ciclo de trabalho de 50% sendo $V_{CC} = 5$ V.

A dissipação de potência em um circuito TTL é essencialmente constante ao longo de sua faixa de freqüência de operação. Entretanto, a dissipação de potência em CMOS é dependente da freqüência. Essa dissipação é extremamente baixa sob condições estáticas (cc) e aumenta conforme

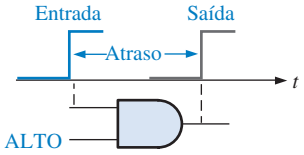
a freqüência aumenta. Essas características são mostradas nas curvas gerais da Figura 14–7. Por exemplo, a dissipação de potência de uma porta TTL Schottky de baixa potência (LS) é 2,2 mW constante. A dissipação de potência de uma porta HCMOS é 2,75μW sob condições estáticas e 170μW em 100 kHz.



◀ **FIGURA 14–7**
Curvas de potência *versus* freqüência para TTL e CMOS.

Tempo de Atraso de Propagação

Quando um sinal passa (se propaga) através de um circuito lógico, ele sempre sofre um atraso de tempo, como ilustra a Figura 14–8. Uma mudança no nível lógico de saída sempre ocorre com um curto intervalo de tempo, denominado de **tempo de atraso de propagação**, após a alteração do nível lógico na entrada que o provocou.

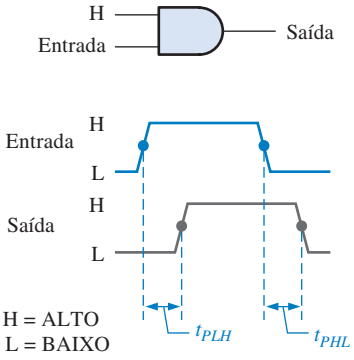


◀ **FIGURA 14–8**
Uma ilustração básica do tempo de atraso de propagação.

Conforme mencionado no Capítulo 3, existem dois tempos de atraso de propagação especificados para as portas lógicas:

- t_{PHL}: O tempo entre um ponto designado no pulso de entrada e o correspondente ponto no pulso de saída quando a saída comuta de nível ALTO para BAIXO.
- t_{PLH}: O tempo entre um ponto designado no pulso de entrada e o correspondente ponto no pulso de saída quando a saída comuta de nível BAIXO para ALTO.

Esses tempos de atraso de propagação são ilustrados na Figura 14–9, tendo como referência um ponto de 50% da borda de transição.



◀ **FIGURA 14–9**
Tempos de atraso de propagação.

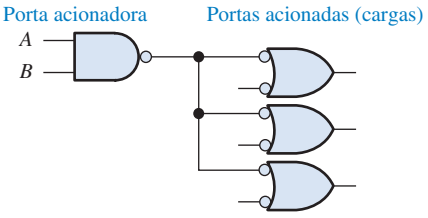
O tempo de atraso de propagação de uma porta limita a freqüência na qual ela pode operar. Quanto maior o tempo de atraso de propagação, menor a freqüência máxima. Portanto, um circuito com uma velocidade maior é aquele que tem um menor tempo de atraso de propagação. Por exemplo, uma porta com um atraso de 3 ns é mais rápida que uma com um atraso de 10 ns.

Produto Velocidade-Potência

O produto velocidade-potência fornece uma base de comparação quando *tanto* o tempo de atraso de propagação *quanto* à dissipação de potência são considerações importantes na seleção de um tipo de circuito lógico a ser usado numa certa aplicação. Quanto menor o produto velocidade-potência, melhor. A unidade do produto velocidade-potência é o picojoule (pJ). Por exemplo, a família HCMOS tem um produto velocidade-potência de 1,2 pJ a 100 kHz enquanto que a família TTL LS tem um valor de 22 pJ.

Carga e Fan-out

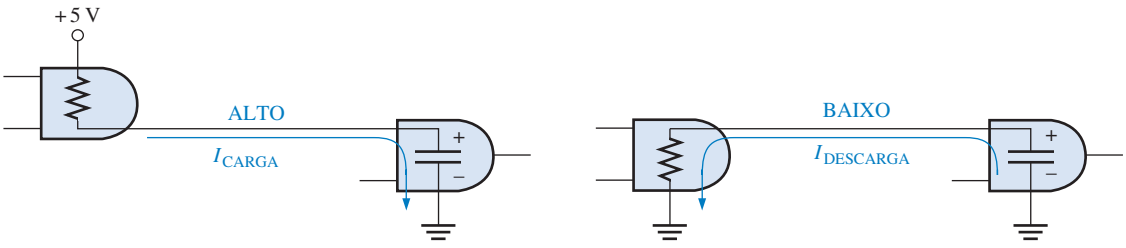
Quando a saída de uma porta lógica está conectada a uma ou mais entradas de outras portas, surge uma carga na porta acionadora, como mostra a Figura 14–10. Existe um limite no número de cargas (portas acionadas) que uma porta pode acionar. Esse limite é denominado de **fan-out** da porta.



▶ **FIGURA 14–10**

Conexão de cargas (entradas de portas) na saída de uma porta.

Carga CMOS As cargas (entradas de portas) CMOS diferem das TTL por causa do tipo de transistor usado em lógica CMOS apresentar predominantemente uma carga capacitiva para a porta acionadora, como ilustra a Figura 14–11. Nesse caso, as limitações são os tempos de carga e descarga associados com a resistência de saída da porta acionadora e a capacitância de entrada das portas que funcionam como carga. Quando a saída da porta acionadora for nível ALTO, a capacitância de entrada da carga (porta) é carregada através da resistência de saída da porta acionadora. Quando a saída da porta acionadora for nível BAIXO, a capacitância é descarregada, conforme indicado na Figura 14–11.



(a) Carga (b) Descarga

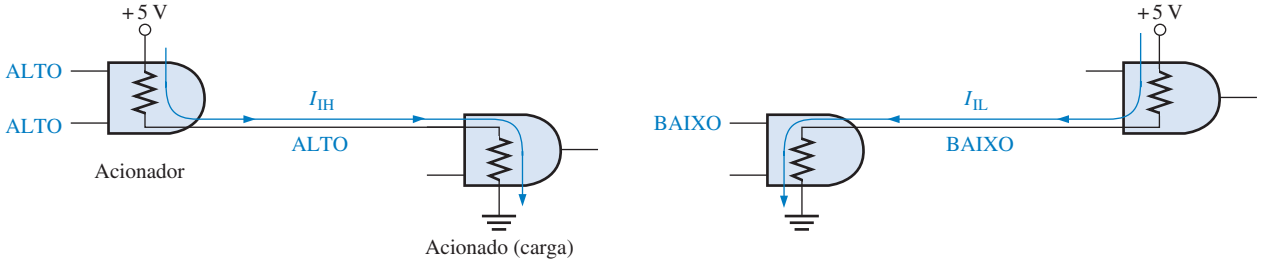
▲ **FIGURA 14–11**

Carga/descarga do capacitor de entrada de uma porta CMOS.

Quando mais entradas de portas (cargas) são acrescentadas na saída da porta acionadora, a capacitância total aumenta porque a capacitância de entrada aparece efetivamente em paralelo. Esse aumento na capacitância faz aumentar os tempos de carga e descarga, reduzindo assim a máxi-

ma freqüência na qual a porta pode operar. Portanto, o fan-out de uma porta CMOS depende da freqüência de operação. Quanto menos cargas (entradas), maior a freqüência máxima.

Carga TTL Uma porta acionadora TTL fornece corrente para a entrada de uma porta (carga) no estado ALTO (I_{IH}) e absorve corrente da entrada de uma porta (carga) no estado BAIXO (I_{IL}). O **fornecimento de corrente e a absorção de corrente** estão ilustrados de uma forma simples na Figura 14–12, onde os resistores presentes são resistências de entrada e saída internas à porta para as duas condições.

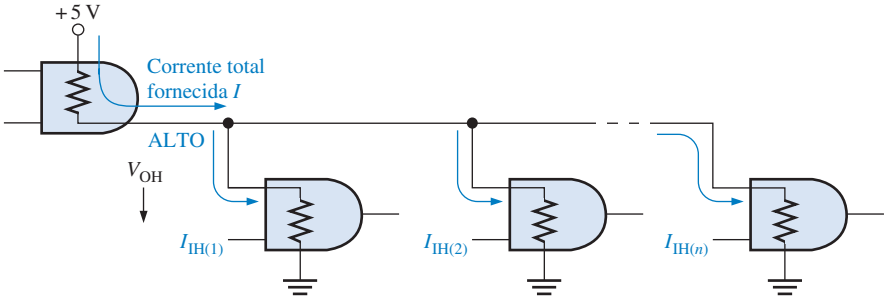


(a) Fornecimento de corrente (b) Absorção de corrente

▲ FIGURA 14–12

Ilustração básica do fornecimento e absorção de corrente em portas lógicas.

Quanto mais portas (cargas) são conectadas na porta acionadora, a carga para essa porta aumenta. A corrente fornecida total aumenta a cada entrada de porta acrescentada, conforme ilustra a Figura 14–13. à medida que essa corrente aumenta, a queda de tensão interna na porta acionadora aumenta, fazendo com que a tensão de saída, V_{OH}, diminua. Se um número excessivo de portas (cargas) forem conectadas, V_{OH} cai abaixo de $V_{OH(mín)}$, sendo a margem de ruído para o nível ALTO reduzida, comprometendo assim a operação do circuito. Além disso, conforme o fornecimento de corrente total aumenta, a dissipação de potência da porta acionadora aumenta.



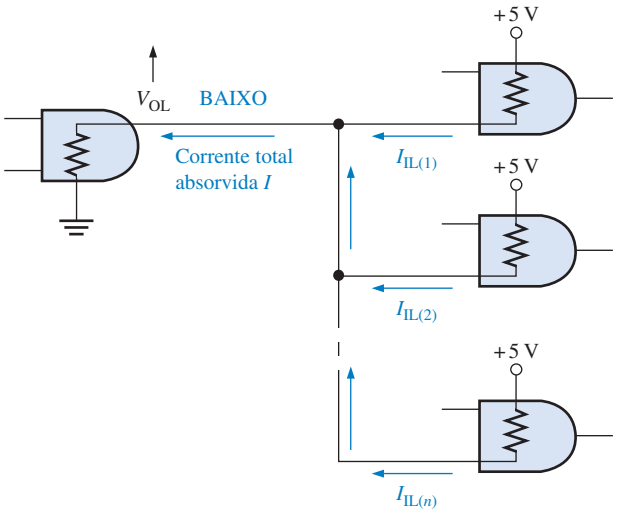
◀ FIGURA 14–13

Carga TTL no estado ALTO.

O fan-out é o número máximo de entradas de porta (cargas) que podem ser conectadas sem afetar contrariamente as características operacionais especificadas para a porta. Por exemplo, a família TTL de baixa potência Schottky (LS) tem um fan-out de 20 unidades de carga. Uma entrada de uma mesma família lógica da porta acionadora é denominada e **unidade de carga**.

A corrente absorvida total também aumenta a cada entrada de porta (carga) acrescentada, como mostra a Figura 14–14. Conforme essa corrente aumenta, a queda de tensão interna na porta acionadora também aumenta, fazendo com que V_{OL} aumente. Se um número excessivo de cargas for acrescentado, V_{OL} excede a $V_{OL(máx)}$, sendo reduzida a margem de ruído de nível BAIXO.

Em TTL, a capacidade de absorção de corrente (estado de saída BAIXO) é o fator de limitação na determinação do fan-out.



▶ **FIGURA 14–14**
Carga TTL no estado BAIXO.

SEÇÃO 14–1 REVISÃO
As respostas estão no final do capítulo.

1. Defina V_{IH}, V_{IL}, V_{OH} e V_{OL}.
2. O melhor é ter um baixo valor de margem de ruído ou um alto valor?
3. A porta *A* tem um tempo de atraso de propagação maior que a porta *B*. Qual dessas portas pode operar numa freqüência maior?
4. Como uma carga em excesso afeta a margem de ruído de uma porta?

14-2 CIRCUITOS CMOS

Esta seção discute basicamente o circuito interno de um dispositivo CMOS. A abreviação CMOS vem de *complementary metal-oxide semiconductor* (semicondutor de óxido metálico complementar). O termo complementar se refere ao uso de dois tipos de transistores no circuito de saída. Um MOSFET canal *n* (transistor de efeito de campo MOS) e um MOSFET canal *p* são usados.

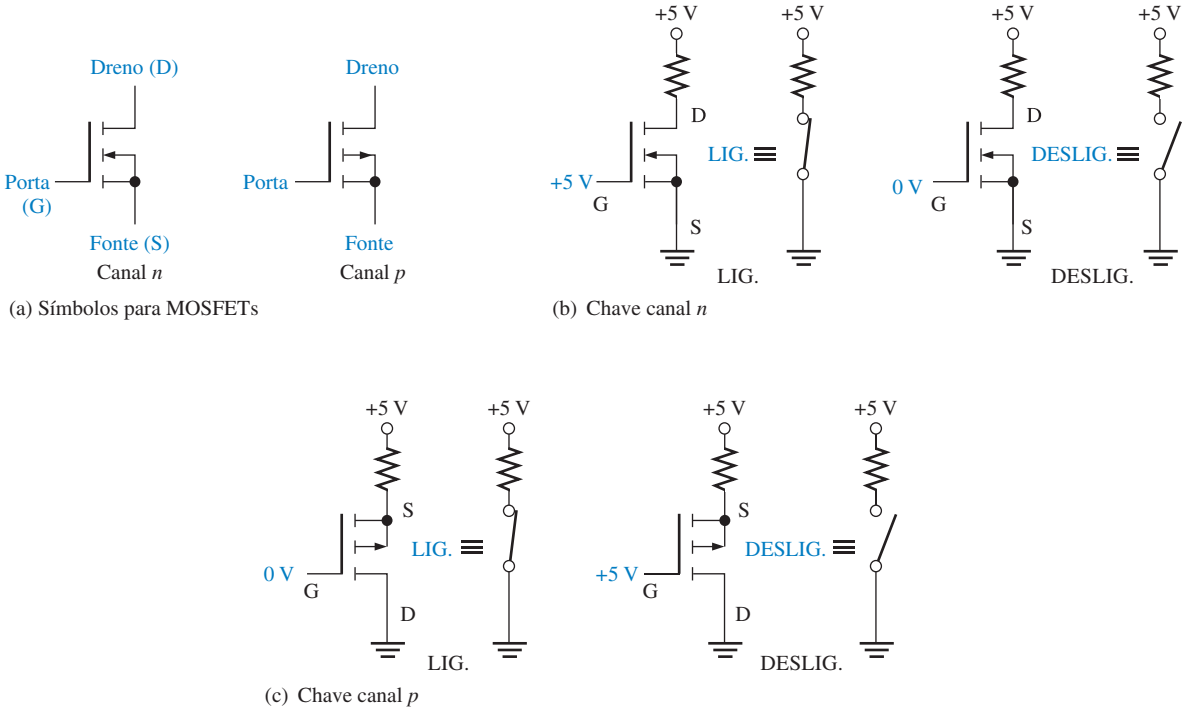
Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar um MOSFET pelo símbolo ■ Discutir a ação de comutação de um MOSFET ■ Descrever a operação básica de um circuito inversor CMOS ■ Descrever a operação básica das porta NAND e NOR CMOS ■ Explicar a operação de uma porta CMOS com saída em dreno aberto ■ Discutir a operação de portas CMOS tristate ■ Fazer uma lista das precauções necessárias quando se manuseia dispositivos CMOS

MOSFET

Os transistores de efeito de campo de semicondutor de óxido metálico (**MOSFETs**) são os elementos ativos de comutação em circuitos CMOS. Esses dispositivos são bastante diferentes na construção e operação interna do transistor bipolar de junção usado em circuitos TTL, porém a ação de comutação é basicamente a mesma: eles funcionam idealmente como chaves abertas ou fechadas, dependendo do sinal de entrada.

A Figura 14–15(a) mostra os símbolos para os MOSFETs canal *n* e canal *p*. Conforme indicado, os três terminais de um MOSFET são **porta**, **fonte** e **dreno**. Quando a tensão na porta de um MOSFET canal *n* for mais positiva que a da fonte, o MOSFET está ligado (*saturação*), existindo, idealmente, uma chave fechada entre dreno e fonte. Quando a tensão porta-fonte é zero, o MOSFET está desligado (*corte*), existindo, idealmente, uma chave aberta entre dreno e fonte. Essa ope-

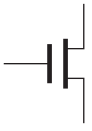


▲ **FIGURA 14–15**

Símbolos básicos e ações de comutação de MOSFETs.

ração é ilustrada na Figura 14–15(b). O MOSFET canal *p* opera com polaridades de tensões opostas, como mostra a parte (c) da figura.

Algumas vezes é usado um símbolo simplificado para o MOSFET, como mostra a Figura 14–16.

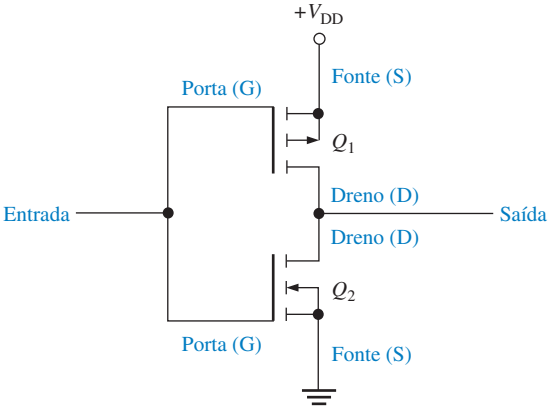


◄ **FIGURA 14–16**

Símbolo simplificado para o MOSFET.

Inversor CMOS

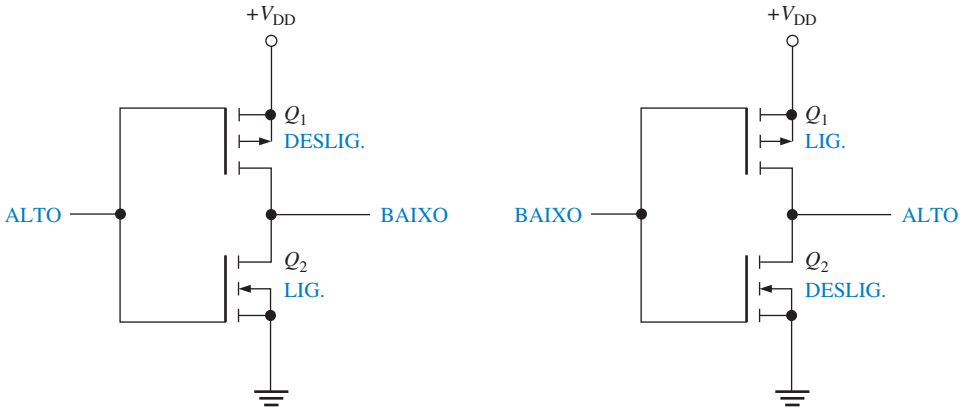
O circuito MOS complementar (CMOS) usa o MOSFET em pares complementares como elemento básico. Um par complementar usa MOSFETs canal *n* e canal *p* tipo enriquecimento, como mostra a Figura 14–17.



◄ **FIGURA 14–17**

Um circuito inversor CMOS.

Quando um nível ALTO é aplicado na entrada, como mostra a Figura 14–18(a), o MOSFET Q_1 (canal *p*) está desligado (*off*) e o MOSFET Q_2 (canal *n*) está ligado (*on*). Essa condição conecta a saída em GND através da resistência de Q_2 ligado, resultando em uma saída de nível BAIXO. Quando um nível BAIXO é aplicado na entrada, como mostra a Figura 14–18(b), Q_1 está ligado e Q_2 está desligado. Essa condição conecta a saída em +VDD (tensão de alimentação cc) através da resistência de Q_1 ligado, resultando em uma saída de nível ALTO.

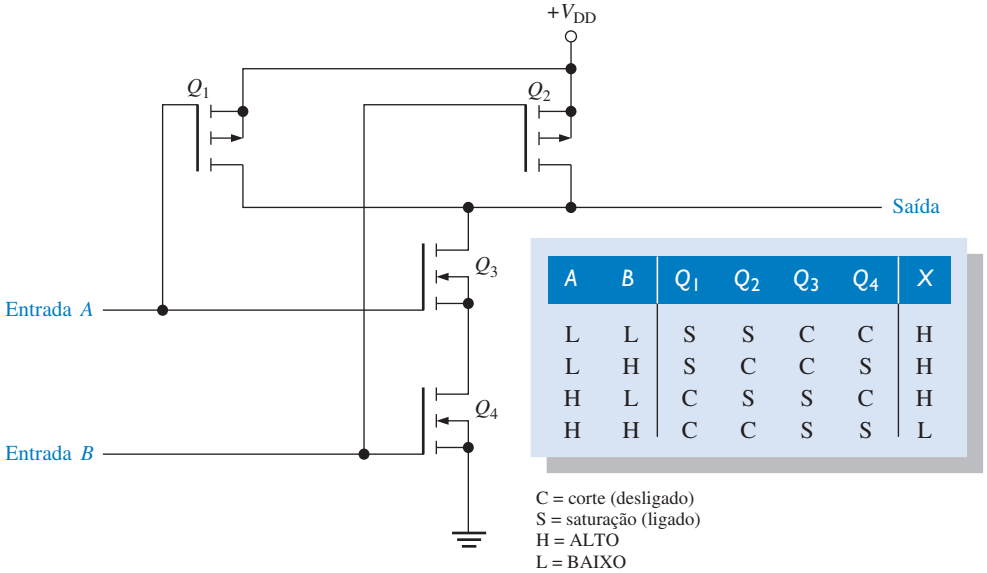


(a) Entrada em nível ALTO, saída em nível BAIXO (b) Entrada em nível BAIXO, saída em nível ALTO

▶ **FIGURA 14–18**
Operação de um inversor CMOS.

Porta NAND CMOS

A Figura 14–19 mostra uma porta NAND CMOS com duas entradas. Observe a configuração dos pares complementares (MOSFETs canal *n* e canal *p*).



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	H
H	L	C	S	S	C	H
H	H	C	C	S	S	L

C = corte (desligado)
S = saturação (ligado)
H = ALTO
L = BAIXO

▶ **FIGURA 14–19**
Circuito de uma porta NAND CMOS.

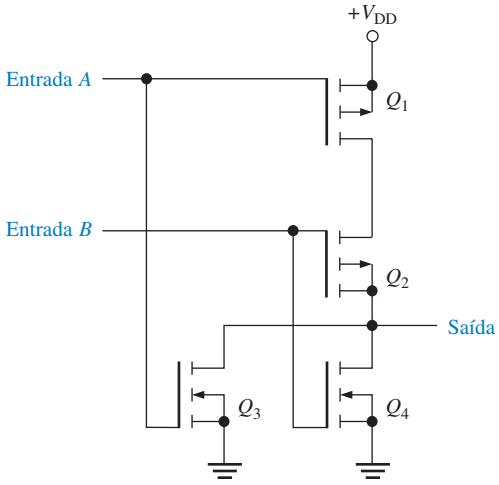
A operação de uma porta NAND CMOS é a seguinte:

- Quando as duas entradas estão em nível BAIXO, Q_1 e Q_2 estão ligados, e Q_3 e Q_4 desligados. A saída é colocada em nível ALTO através da resistência em paralelo de Q_1 e Q_2 ligados.

- Quando a entrada *A* está em nível BAIXO e a entrada *B* está em nível ALTO, Q_1 e Q_4 estão ligados, e Q_2 e Q_3 desligados. A saída é colocada em nível ALTO através da baixa resistência de Q_1 ligado.
- Quando a entrada *A* está em nível ALTO e a entrada *B* em nível BAIXO, Q_1 e Q_4 estão desligados, e Q_2 e Q_3 ligados. A saída é colocada em nível ALTO através da baixa resistência de Q_2 ligado.
- Finalmente, quando as duas entradas estão em nível ALTO, Q_1 e Q_2 estão desligados, e Q_3 e Q_4 ligados. Nesse caso, a saída é colocada em nível BAIXO através da resistência em série de Q_3 e Q_4 ligados para GND.

Porta NOR CMOS

A Figura 14–20 mostra uma porta NOR CMOS com duas entradas. Observe a configuração dos pares complementares.



A	B	Q_1	Q_2	Q_3	Q_4	X
L	L	S	S	C	C	H
L	H	S	C	C	S	L
H	L	C	S	S	C	L
H	H	C	C	S	S	L

C = corte (desligado)
S = saturação (ligado)
H = ALTO
L = BAIXO

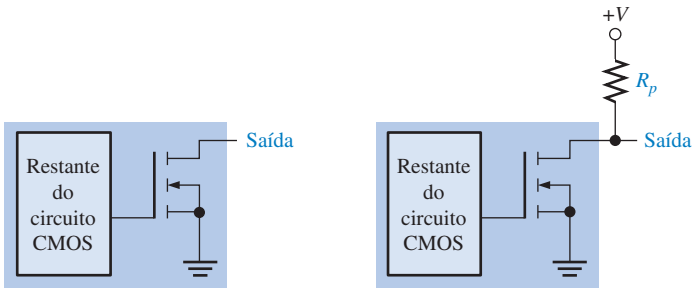
◀ FIGURA 14–20
Circuito de uma porta NOR CMOS.

A operação de uma porta NOR CMOS é a seguinte:

- Quando as duas entradas estão em nível BAIXO, Q_1 e Q_2 estão ligados, e Q_3 e Q_4 desligados. Como resultado, a saída é colocada em nível ALTO através das resistências em série de Q_1 e Q_2 em série.
- Quando a entrada A está em nível BAIXO e a entrada *B* em nível ALTO, Q_1 e Q_4 estão ligados, e Q_2 e Q_3 desligados. A saída é colocada em nível BAIXO através da baixa resistência, para GND, de Q_4 ligado.
- Quando a entrada A está em nível ALTO e a entrada *B* em nível BAIXO, Q_1 e Q_4 estão desligados, e Q_2 e Q_3 estão ligados. A saída é colocada em nível BAIXO através da resistência, para GND, de Q_3 ligado.
- Quando as duas entradas estão em nível ALTO, Q_1 e Q_2 estão desligados, e Q_3 e Q_4 ligados. A saída é colocada em nível BAIXO através das resistências em paralelo de Q_3 e Q_4 ligados para GND.

Portas de Dreno Aberto

Dreno aberto significa que o terminal do dreno do transistor de saída não tem conexão, devendo ser conectado externamente a V_{DD} através de uma carga. Uma porta CMOS de dreno aberto equivale a uma porta TTL de coletor aberto (discutida na Seção 14–3). Um circuito com saída de dreno aberto apresenta um único MOSFET de canal *n* como vemos na Figura 14–21(a). Um **resistor de pull-up** (elevador) externo tem que ser usado, com mostra a parte (b) da figura, para produzir um estado de saída ALTO. Alem disso, as saídas de dreno aberto podem ser conectadas em configuração de uma lógica AND com fios (wired-AND), um conceito discutido na próxima seção em relação a TTL.

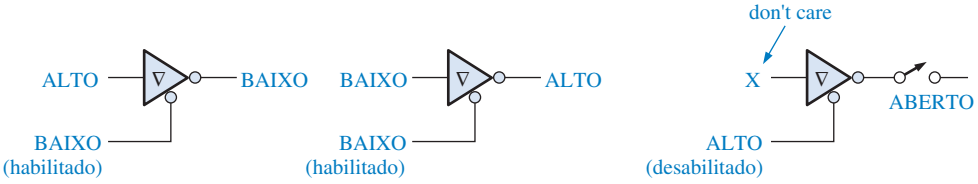


(a) Saída desconectada (b) Com resistor de pull-up

▶ **FIGURA 14–21**
Portas CMOS de dreno aberto.

Portas CMOS Tristate

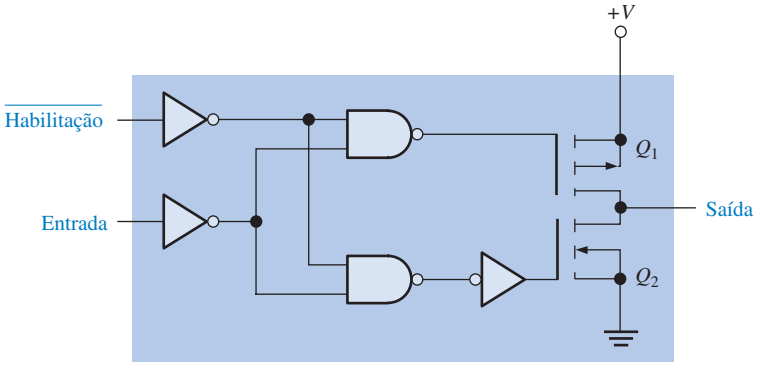
As saídas tristate são encontradas nos circuitos TTL e CMOS. A saída **tristate** combina as vantagens dos circuitos *totem-pole* e coletor aberto. Como você se lembra, os três estados de saída são ALTO, BAIXO e alta impedância (**Hi-Z**). Quando selecionado para operação normal com níveis lógicos, conforme determinado pelo estado da entrada de habilitação, um circuito tristate opera da mesma forma que uma porta comum. Quando um circuito tristate é selecionado para operar com a saída em alta impedância, esta é efetivamente desconectada do restante do circuito através de polarização do circuito interno. A Figura 14–22 ilustra a operação de um circuito tristate. O triângulo invertido (∇) indica uma saída tristate.



(a) Habilitado para operação lógica normal (b) Alta impedância

▶ **FIGURA 14–22**
Os três estados do circuito tristate.

O circuito em uma porta CMOS tristate, como mostra a Figura 14–23, permite que cada um dos transistores de saída Q_1 e Q_2 sejam desligados ao mesmo tempo, desconectando assim a saída do restante do circuito.



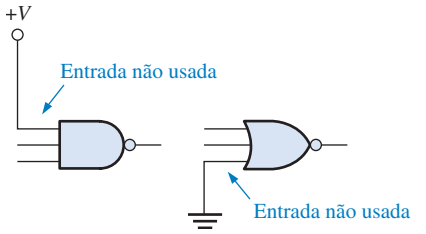
▶ **FIGURA 14–23**
Um inversor CMOS tristate.

Quando a entrada de habilitação for nível BAIXO, o dispositivo é habilitado para operação lógica normal. Quando a entrada de habilitação for nível ALTO, os transistores Q_1 e Q_2 são desligados e o circuito entra no estado de alta impedância.

Precauções no Manuseio de Dispositivos CMOS

Conforme estudamos, todos os dispositivos CMOS estão sujeitos a danos em função de descarga eletrostática (ESD). Portanto, eles têm que ser manuseados com um cuidado especial. Como revisão, veja a seguir algumas precauções:

1. Todos os dispositivos CMOS devem ser transportados e armazenados em espuma condutiva para evitar a formação de carga eletrostática. Quando removidos da espuma, os pinos não devem ser tocados.
2. Os dispositivos devem ser colocados com os pinos voltados para baixo sobre uma superfície aterrada, tal como uma placa metálica, quando removidos do material de proteção. Não coloque os dispositivos CMOS em espumas de poliestireno ou bandejas plásticas.
3. Todas as ferramentas, equipamentos de teste e bancadas metálicas devem ser aterrados. A pessoa que trabalha com dispositivos CMOS deve, em certos ambientes, ter o pulso aterrado com uma pulseira anti-estática (esta possui um resistor em série de alto valor). O resistor evita que a pessoa tome um choque mais intenso caso entre em contato com uma fonte de tensão.
4. Não coloque dispositivos CMOS (ou quaisquer outros CIs) nos soquetes ou placas de circuito energizadas.
5. Todas as entradas não usadas devem ser conectadas à linha de alimentação, ou GND, conforme indica a Figura 14–24. Caso alguma entrada seja deixada aberta, ela pode adquirir carga eletrostática atingindo níveis não permitidos provocando estados imprevisíveis na saída.



◀ **FIGURA 14–24**
Conexão das entradas CMOS não usadas.

6. Após a montagem da placa de circuito impresso, uma proteção deve ser colocada para armazenar ou transportar as placas com seus conectores em espuma condutiva. Os pinos de entrada e saída CMOS também podem ser protegidos com resistores de alto valor conectados em GND.

SEÇÃO 14–2 REVISÃO

1. Que tipo de transistor é usado em circuitos CMOS?
2. Qual é o significado do termo *MOS complementar*?
3. Porque os dispositivos CMOS têm que ser manuseados com cuidado?

14-3 CIRCUITOS TTL

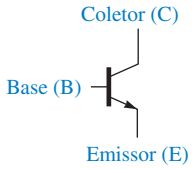
Essa seção aborda a operação do circuito interno de portas lógicas TTL com saídas totem-pole. Além disso, a operação de portas TTL com saídas de coletor aberto e a operação de portas tristate são abordadas.

Ao final do estudo desta seção você deverá ser capaz de:

■ Identificar um transistor de junção bipolar (BJT) pelo seu símbolo ■ Descrever a ação de chaveamento de um BJT ■ Descrever a operação básica de um circuito inversor TTL ■ Descrever a operação básica dos circuitos TTL das portas AND, NAND, OR e NOR ■ Explicar o que é uma saída totem-pole ■ Explicar a operação e o uso de uma porta TTL com saída de coletor aberto ■ Explicar a operação de uma porta com saída tristate

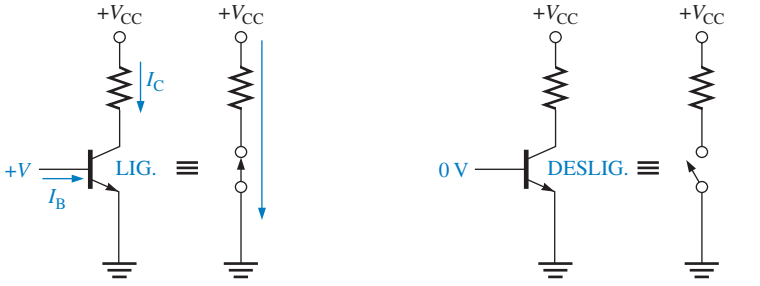
Transistor de Junção Bipolar

O transistor de junção **bipolar** (BJT) é o elemento de chaveamento usado em todos os circuitos TTL. A Figura 14–25 mostra o símbolo para um BJT *npn* com seus três terminais; **base**, **emissor** e **coletor**. Um BJT tem duas junções, a **junção** base-emissor e a junção base-coletor.



▶ **FIGURA 14–25**
O símbolo para um BJT.

A operação básica de comutação pode ser descrita da seguinte forma: Quando a base é aproximadamente 0,7 V mais positiva que o emissor e quando uma corrente suficiente é fornecida à base, o **transistor** liga e entra em saturação. Na saturação, o transistor se comporta idealmente como uma chave fechada entre coletor e emissor, como ilustra a Figura 14–26(a). Quando a base é menos que 0,7 V mais positiva que o emissor, o transistor desliga, se tornando uma chave aberta entre coletor e emissor, como mostra a parte (b) da figura. Sintetizando em termos gerais, um nível ALTO na base liga o transistor tornando-o como uma chave fechada. Um nível BAIXO na base desliga o transistor tornando-o como uma chave aberta. Em TTL, alguns BJTs têm múltiplos emissores.



(a) Transistor saturado (LIG.) e o circuito equivalente ideal com uma chave

(b) Transistor DESLIG. e o circuito equivalente ideal com uma chave

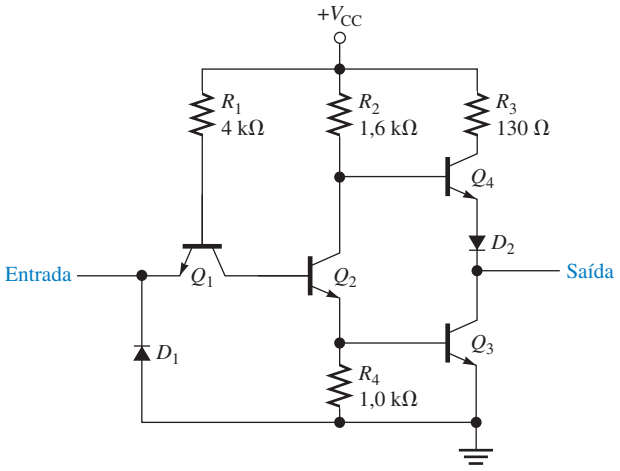
▶ **FIGURA 14–26**
Comportamento do BJT como chave ideal. A direção da corrente mostrada é a convencional. A notação do fluxo de elétrons é na direção oposta.

Inversor TTL

A função lógica de um inversor ou qualquer tipo de porta é sempre a mesma, independente do tipo de tecnologia de circuito usada. A Figura 14–27 mostra um circuito TTL padrão para um inversor. Nessa figura Q_1 é o transistor de acoplamento de entrada e D_1 é o **diodo** de grampeamento de entrada. O transistor Q_2 é denominado de *divisor de fase*, e a combinação de Q_3 e Q_4 forma o circuito de saída freqüentemente denominado de arranjo **totem-pole**.

Quando a entrada for nível ALTO, a junção base-emissor de Q_1 é **polarizada reversamente** e a junção base-coletor é **polarizada diretamente**. Essa condição permite uma corrente através de R_1 e da junção base-coletor de Q_1 para a base de Q_2 colocando-o na saturação. Como resultado, Q_3 é ligado por Q_2 e sua tensão de coletor, que é a saída, está próxima do potencial de GND. Portanto, temos uma saída em nível BAIXO em resposta a uma entrada em nível ALTO. No mesmo instante, o coletor de Q_2 está com uma tensão suficientemente baixa para manter Q_4 em corte.

Quando a entrada é nível BAIXO, a junção base-emissor de Q_1 é polarizada diretamente e a junção base-coletor é polarizada reversamente. Existe uma corrente através de R_1 e da junção base-emissor de Q_1 para a entrada em nível BAIXO. O nível BAIXO na entrada cria uma percurso até GND para a corrente. Não existe corrente na base de Q_2, assim ele fica desligado. O coletor

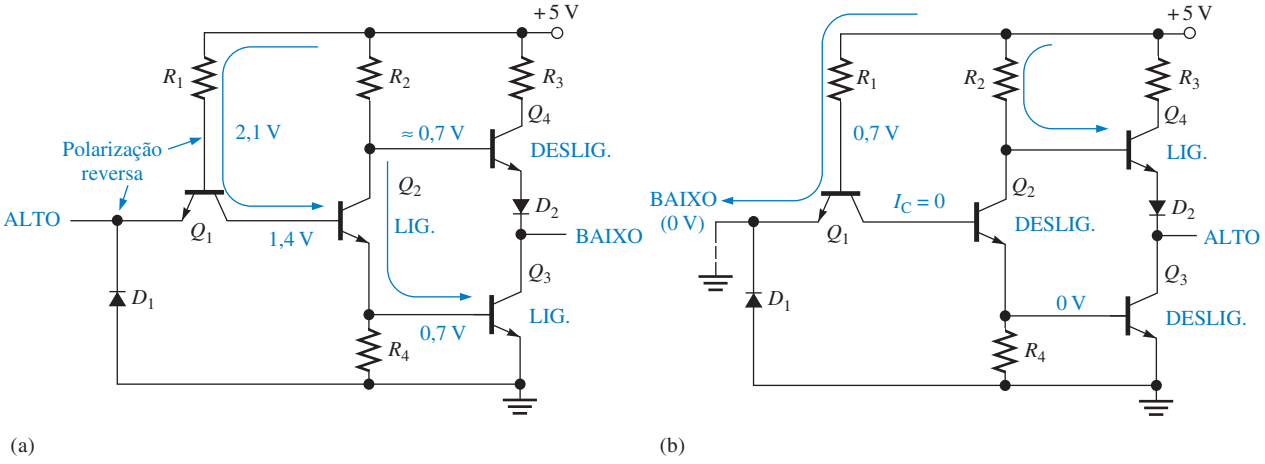


◀ **FIGURA 14–27**
Circuito de um inversor TTL padrão.

de Q_2 está em nível ALTO, ligando assim Q_4. Este transistor saturado fornece um percurso de baixa resistência de V_{CC} para a saída; temos assim um nível ALTO na saída em resposta a um nível BAIXO na entrada. Ao mesmo tempo, o emissor de Q_2 está no potencial GND, o que mantém Q_3 em corte.

O diodo D_1 no circuito TTL evita *spikes* negativos de tensão na entrada que poderiam danificar Q_1. O diodo D_2 garante que Q_4 estará desligado quando Q_2 estiver ligado (entrada em nível ALTO). Nessa condição, a tensão de coletor de Q_2 é igual à tensão base-emissor, V_{BE}, de Q_3 mais a tensão coletor-emissor, V_{CE}, de Q_2. O diodo D_2 fornece adicionalmente uma queda de tensão equivalente a V_{BE} em série com a junção base emissor de Q_4 para garantir o corte deste quando Q_2 estiver ligado.

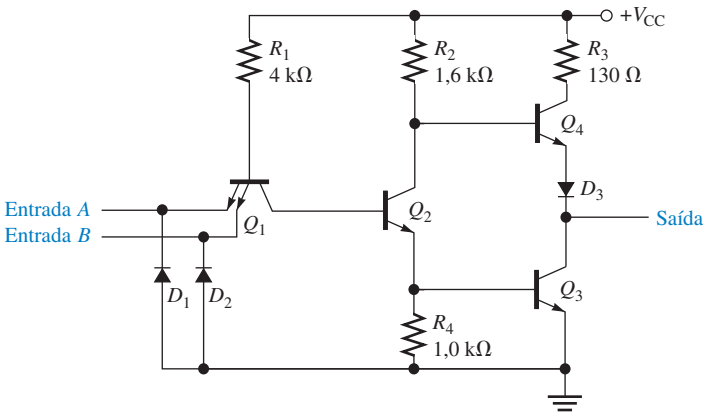
A operação do inversor TTL para os dois estados de entrada é ilustrado na Figura 14–28. No circuito mostrado na parte (a) da figura, a base de Q_1 está 2,1 V acima de GND, assim Q_2 e Q_3 estão ligados. No circuito mostrado na parte (b) da figura, a base de Q_1 está cerca de 0,7 V acima de GND — que não é suficiente para ligar Q_2 e Q_3.



▲ **FIGURA 14–28**
Operação de um inversor TTL.

Porta NAND TTL

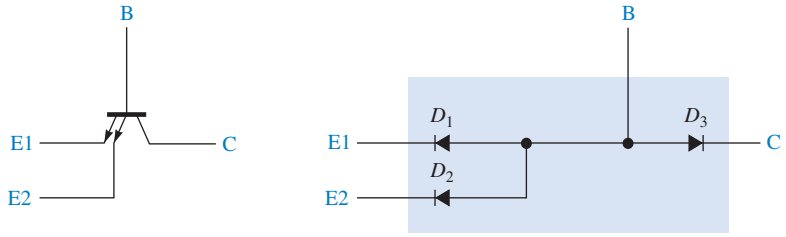
Uma porta NAND TTL de 2 entradas é mostrada na Figura 14–29. Basicamente, é o mesmo que o circuito de um inversor exceto por um emissor de entrada adicional de Q_1. Os transistores de



▶ **FIGURA 14–29**

Circuito de uma porta NAND TTL.

múltiplos emissores na tecnologia TTL são usados para dispositivos de entrada. Esses transistores podem ser comparados a um arranjo de diodos, como mostra a Figura 14–30.



▲ **FIGURA 14–30**

Circuito equivalente com diodos de um transistor de múltiplos emissores TTL.

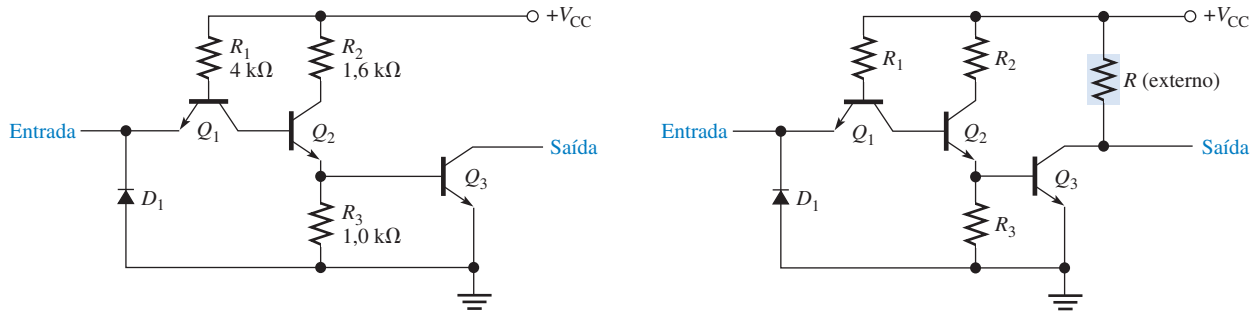
Talvez você entenda melhor a operação desse circuito visualizando Q_1 na Figura 14–29 substituindo pelo arranjo com diodos mostrado na Figura 14–30. Um nível BAIXO na entrada *A* ou *B* polariza diretamente o diodo correspondente e polariza reversamente D_3 (junção base-coletor de Q_1). Essa ação mantém Q_2 desligado, resultando em um nível ALTO na saída da mesma forma como descrito para o inversor TTL. É claro que, um nível BAIXO nas duas entradas terá o mesmo resultado.

Um nível ALTO nas duas entradas polariza reversamente os dois diodos de entrada e polariza diretamente D_3 (junção base-coletor de Q_1). Essa ação liga Q_2, resultando em um nível BAIXO na saída da mesma forma como descrito para o inversor TTL. O leitor deve ter reconhecido essa operação como a da função AND: A saída é nível BAIXO apenas se todas as entradas forem nível ALTO.

Portas de Coletor Aberto

Todas as portas TTL descritas nas seções anteriores têm um circuito de saída totem-pole. Um outro tipo de saída disponível em CIs TTL é a saída de **coletor aberto**. Essa saída é comparável à saída de dreno aberto de um CMOS. Um inversor TTL padrão com coletor aberto é mostrado na Figura 14–31(a). Os outros tipos de portas também são encontradas com saídas de coletor aberto.

Observe que a saída é o coletor do transistor Q_3 sem nenhuma conexão, por isso o nome de *coletor aberto*. Para obter os níveis lógicos ALTO e BAIXO corretamente na saída do circuito, tem que ser conectado um resistor de pull-up externo do coletor de Q_3 para V_{CC}, como mostra a Figura 14–31(b). Quando Q_3 estiver desligado, a saída é elevada para V_{CC} através do resistor externo. Quando Q_3 estiver ligado, a saída é conectada a GND através do transistor saturado.

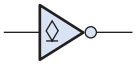


(a) Circuito de um inversor com coletor aberto

(b) Com um resistor de pull-up externo

▲ **FIGURA 14–31**

Inversor TTL com saída de coletor aberto.



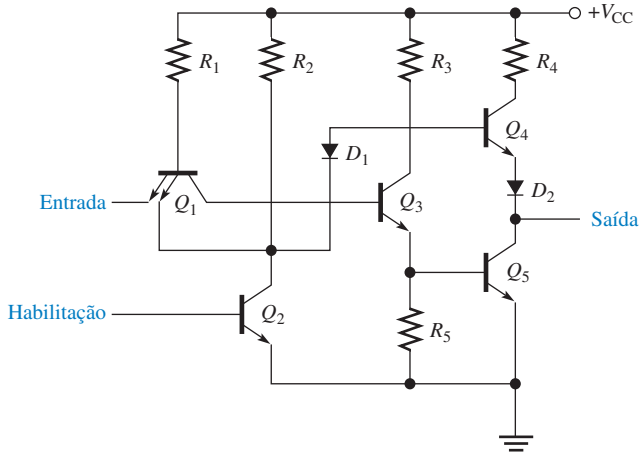
◀ **FIGURA 14–32**

Símbolo de um inversor com coletor aberto.

O símbolo padrão ANSI/IEEE que indica uma saída de coletor aberto é mostrado na Figura 14–32 para um inversor. Este símbolo é o mesmo usado no caso de uma saída de dreno aberto.

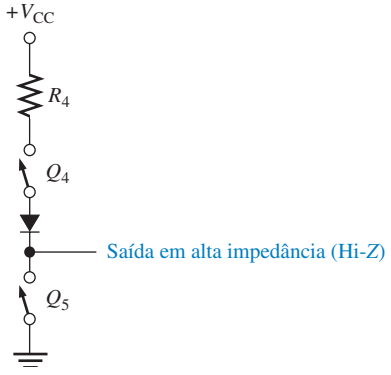
Portas TTL Tristate

A Figura 14–33 mostra o circuito básico para um inversor tristate TTL. Quando a entrada de habilitação estiver em nível BAIXO, Q_2 estará desligado e o circuito de saída opera como uma configuração totem-pole normal, na qual o estado da saída depende do estado da entrada. Quando a entrada de habilitação for nível ALTO, Q_2 liga. Assim haverá um nível BAIXO no segundo emissor de Q_1, fazendo desligar Q_3 e Q_5 e o diodo D_1 é polarizado diretamente, fazendo com que Q_4 também desligue. Quando os dois transistores totem-pole estão desligados, eles estão efetivamente abertos e a saída está completamente desconectada do circuito interno, como ilustra a Figura 14–34.



▲ **FIGURA 14–33**

Circuito inversor tristate básico.

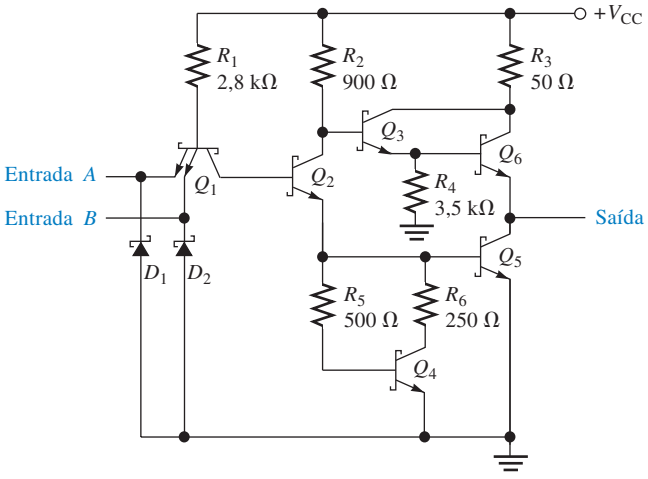


▲ **FIGURA 14–34**

Circuito equivalente para a saída tristate no estado de alta impedância.

TTL Schottky

O circuito da porta NAND TTL padrão ou básica foi discutido anteriormente. Ele é um tipo de circuito que drena corrente através da carga quando a saída está em nível BAIXO e fornece uma corrente desprezível através da carga quando a saída está em nível ALTO. A maioria dos circuitos TTL usados atualmente é de alguma forma da família TTL *Schottky*, a qual provê uma comutação mais rápida através da incorporação de diodos *Schottky* para evitar que os transistores entrem na saturação, diminuindo assim o tempo para que o transistor ligue ou desligue. A Figura 14–35 mostra um circuito de uma porta Schottky. Observe os símbolos para o transistor e o diodo Schottky. Os dispositivos Schottky são indicados por um S na especificação, tal como 74S00. Outros tipos de TTL Schottky são o Schottky de baixa potência indicado por LS, Schottky avançado indicado por AS, Schottky de baixa potência avançado indicado por ALS e o fast indicado por F.



▲ **FIGURA 14–35**

Porta NAND TTL Schottky.

SEÇÃO 14–3 REVISÃO

1. Um BJT *npn* está ligado quando a base é mais negativa que o emissor. (V ou F)
2. Em termos de ação de comutação, o que representam os estados ligado e desligado de um BJT?
3. Quais os dois principais tipos de circuitos de saída TTL?
4. Explique em que um circuito tristate difere de um normal (que tem dois estados lógicos).

14-4 CONSIDERAÇÕES PRÁTICAS NO USO DE TTL

Embora CMOS seja a tecnologia de CI predominante em aplicações comerciais e industriais, TTL ainda é usado. Em aplicações educacionais, TTL é geralmente preferido porque ele não tem as restrições de manuseio que CMOS tem devido à ESD. Por causa disso, considerações práticas importantes no uso e aplicação de circuitos TTL serão abordadas usando a família TTL padrão para ilustração.

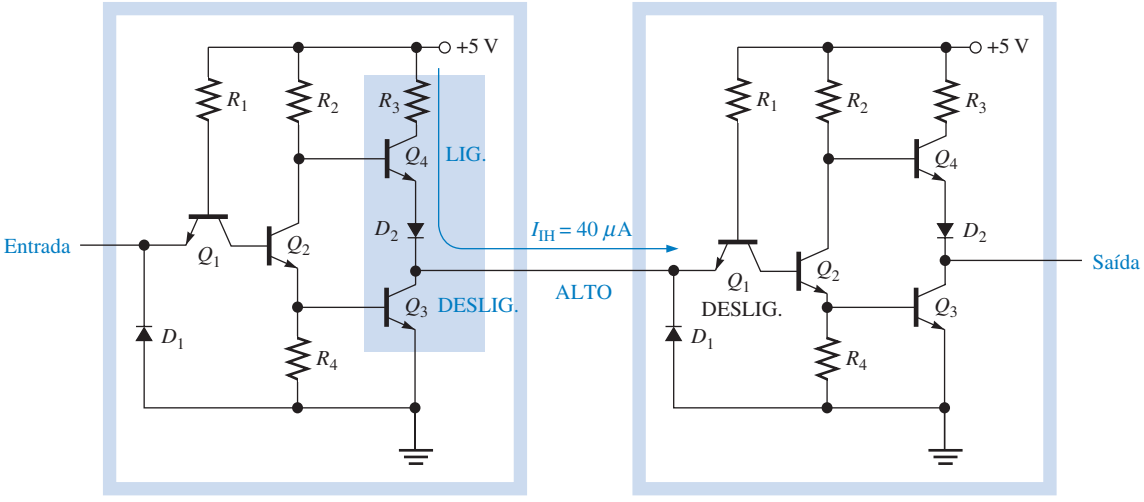
Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever absorção e fornecimento de corrente ■ Usar um circuito de coletor aberto para implementar uma lógica AND com fios ■ Descrever os efeitos da interconexão de duas ou mais saídas totem-pole ■ Usar portas de coletor aberto para acionar LEDs e lâmpadas ■ Explicar o que fazer com as entradas não usadas em TTL

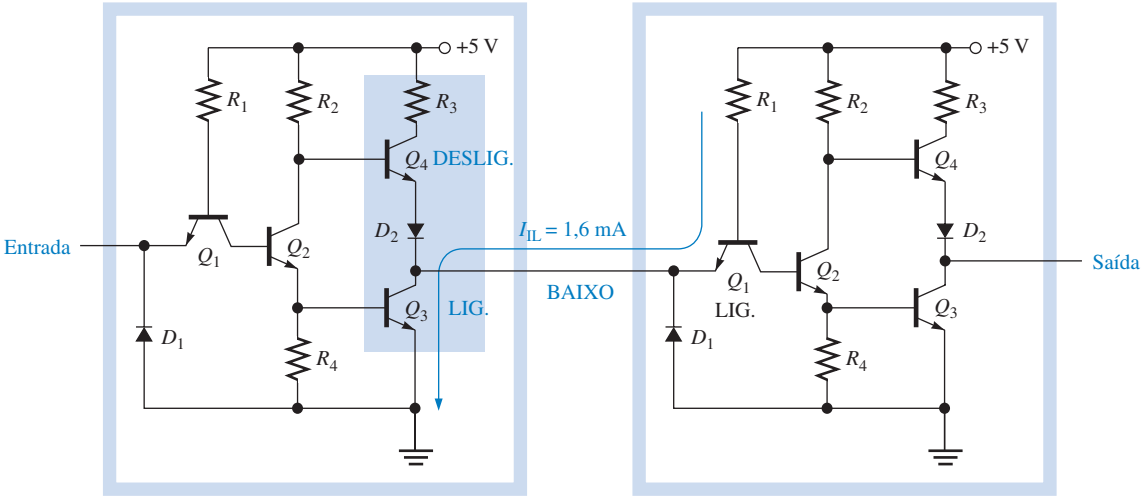
Absorção e Fornecimento de Corrente

Os conceitos de absorção e fornecimento de corrente foram introduzidos na Seção 14–1. Agora que o leitor está mais familiarizado com a configuração do circuito de saída totem-pole usado em TTL, vamos focar na ação de absorção e fornecimento de corrente.

A Figura 14–36 mostra um inversor TTL padrão com uma saída totem-pole conectada à entrada de um outro inversor TTL. Quando a porta acionadora está no estado de saída ALTO, esta



(a) Fornecimento de corrente (valor máximo de I_{IH})



(b) Absorção de corrente (valor máximo de I_{IL})

▲ **FIGURA 14–36**

Ação de absorção e fornecimento de corrente em TTL.

fornece corrente para a carga, como mostra a Figura 14–36(a). A entrada da porta acionada (carga) é semelhante a um diodo polarizado reversamente, assim não existe corrente, em termos práticos, solicitada pela carga. Na realidade, como a entrada não é ideal, existe um máximo de 40μA da saída totem-pole da porta acionadora para a entrada da porta acionada.

Quando a porta acionadora está com a saída no estado BAIXO, esta absorve corrente da carga, como mostra a Figura 14–36(b). Essa corrente tem um valor máximo de 1,6 mA para TTL padrão e é indicada nas **folhas de dados** com um valor negativo porque ela *sai* pela entrada.

EXEMPLO 14–3

Quando uma porta NAND TTL aciona cinco portas TTL, qual o valor da corrente fornecida pela saída e qual o valor da corrente absorvida? (Consulte a Figura 14–36)

Solução

Corrente total fornecida (saída no estado ALTO):

$$I_{IH(máx)} = 40\ \mu A \quad \text{pela entrada}$$

$$I_{T(fonte)} = (5\ \text{entradas})(40\ \mu A/\text{entrada}) = 5(40\ \mu A) = \mathbf{200\ \mu A}$$

Corrente absorvida total (saída no estado BAIXO):

$$I_{IL(máx)} = -1{,}6\ \text{mA por entrada}$$

$$I_{T(absorção)} = (5\ \text{entradas})(-1{,}6\ \text{mA entrada}) = 5(-1{,}6\ \text{mA}) = \mathbf{-8{,}0\ mA}$$

Problema relacionado

Repita o cálculo para uma porta NAND TTL LS. Consulte a folha de dados da Texas Instruments no CD-ROM que acompanha o livro.

EXEMPLO 14–4

Consulte a folha e dados da Texas Instruments no CD-ROM que acompanha o livro e determine o fan-out da porta NAND 7400.

Solução

De acordo com a folha de dados, os parâmetros de corrente são os seguintes:

$$I_{IH(máx)} = 40\ \mu A \qquad I_{OH(máx)} = -400\ \mu A$$

$$I_{IL(máx)} = -1{,}6\ \text{mA} \qquad I_{OL(máx)} = 16\ \text{mA}$$

O fan-out para a saída no estado ALTO é calculado como segue: A corrente $I_{OH(máx)}$ é a corrente máxima que a porta pode fornecer para uma carga. Cada entrada de porta (carga) requer uma corrente $I_{IH(máx)}$ de 40μA. O fan-out para o estado ALTO é

$$\left|\frac{I_{OH(máx)}}{I_{IH(máx)}}\right| = \frac{400\ \mu A}{40\ \mu A} = \mathbf{10}$$

Para a saída no estado BAIXO, o fan-out é calculado como segue: $I_{OL(máx)}$ é a corrente máxima que a porta pode absorver. Cada entrada de porta (carga) produz uma corrente $I_{IL(máx)}$ de –1,6 mA. O fan-out no estado BAIXO é

$$\left|\frac{I_{OL(máx)}}{I_{IL(máx)}}\right| = \frac{16\ \text{mA}}{1{,}6\ \text{mA}} = \mathbf{10}$$

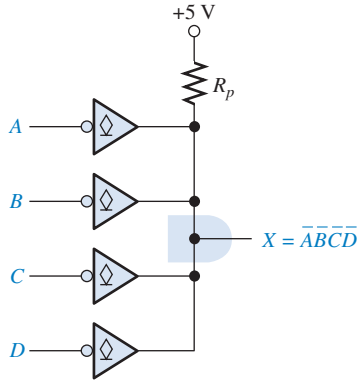
Nesse caso o fan-out nos estados ALTO e BAIXO são iguais.

Problema relacionado

Determine o fan-out para uma porta NAND 74LS00.

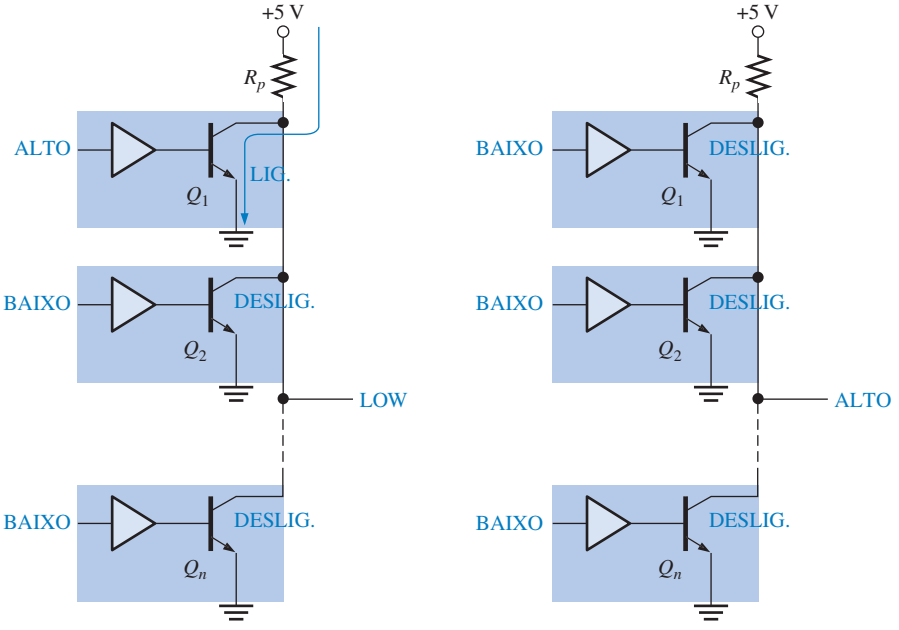
Usando Portas de Coletor Aberto para Implementar Lógica AND com Fios

As saídas das portas de coletor aberto podem ser interconectadas para formar o que denominamos de lógica AND com fios (*wired-AND*). A Figura 14–37 ilustra como quatro inversores são conectados para produzir uma porta AND negativa de 4 entradas. Um único resistor de pull-up externo (R_p) é necessário em todos os circuitos de lógica AND com fios.



◀ **FIGURA 14–37**
Lógica AND com fios usando quatro inversores.

Quando uma (ou mais) entradas estiverem em nível ALTO, a saída *X* é levada para o nível BAIXO porque um transistor de saída estará ligado se comportando como uma chave fechada para GND, conforme ilustra a Figura 14–38(a). Nesse caso apenas um inversor tem uma entrada em nível ALTO, mas isso é suficiente para colocar a saída em nível BAIXO através da saturação do transistor Q_1 conforme indicado.



(a) Quando um ou mais transistores de saída estiverem ligados, a saída será nível BAIXO.

(b) Quando todos os transistores de saída estiverem desligados, a saída será nível ALTO.

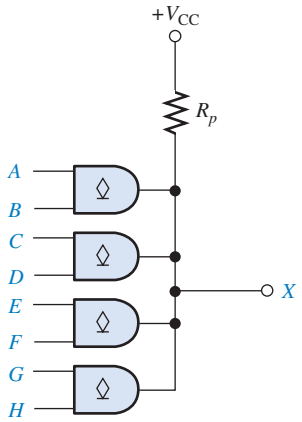
◀ **FIGURA 14–38**
Operação da lógica AND com fios com inversores de coletor aberto.

Para a saída *X* ser nível ALTO, *todas* as entradas dos inversores têm que ser nível baixo de forma que todos os transistores de saída (coletor aberto) estarão desligados, conforme indica a Figura 14–38(b). Quando essa condição existe, a saída *X* é levada para o nível ALTO através do resistor de pull-up. Portanto, a saída *X* é nível ALTO apenas quando *todas* as entradas estiverem em nível BAIXO. Assim, temos uma função AND negativa, conforme expresso pela seguinte equação:

$$X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$$

EXEMPLO 14–5

Escreva a expressão de saída para a configuração da lógica AND com fios com as porta AND de coletor aberto mostradas na Figura 14–39.



▶ FIGURA 14–39

Solução A expressão de saída é

$$X = ABCDEFGH$$

A conexão da lógica AND com fios de quatro portas AND de 2 entradas produz uma porta AND de 8 entradas.

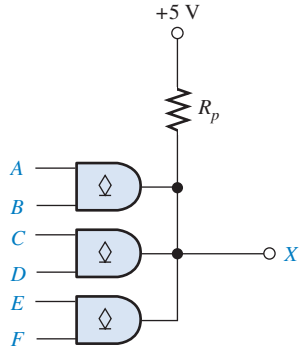
Problema relacionado Determine a expressão de saída se usarmos portas NAND na Figura 14–39.

EXEMPLO 14–6

Três portas AND de coletor aberto são interconectadas em lógica AND com fios conforme mostra a Figura 14–40. Considere que o circuito de lógica AND com fios está acionando quatro entradas TTL padrão (–1,6 mA cada uma).

(a) Escreva a expressão lógica para X.

(b) Determine o valor máximo de R_p se $I_{OL(máx)}$ for de 30 mA para cada porta e $V_{OL(máx)}$ for 0,4 V.



▶ FIGURA 14–40

Solução **(a)** $X = ABCDEF$

(b) 4(1,6 mA) = 6,4 mA

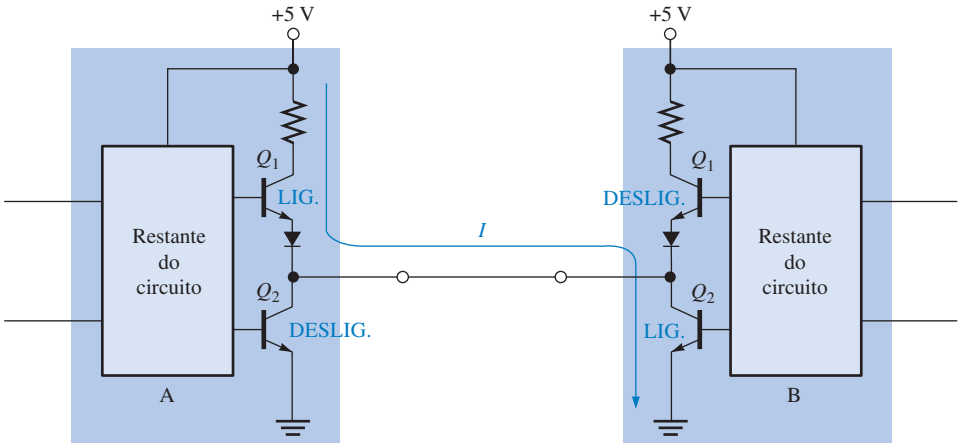
$$I_{R_P} = I_{OL(max)} - 6{,}4\text{ mA} = 30\text{ mA} - 6{,}4\text{ mA} = 23{,}6\text{ mA}$$

$$R_P = \frac{V_{CC} - V_{OL(max)}}{I_{R_P}} = \frac{5\text{ V} - 0{,}4\text{ V}}{23{,}6\text{ mA}} = \mathbf{195\ \Omega}$$

Problema relacionado Mostre o circuito de lógica AND com fios para obter uma função AND de 10 entradas usando um 74LS09 (quatro portas AND de 2 entradas).

Interconexão de Saídas Totem-Pole

As saídas totem-pole não podem ser interconectadas porque uma conexão desse tipo poderia produzir uma corrente excessiva resultando em danos aos dispositivos. Por exemplo, na Figura 14–41, quando Q_1 do dispositivo A e Q_2 do dispositivo B estiverem ligados, a saída do dispositivo A é colocada efetivamente em curto-circuito com GND através de Q_2 no dispositivo B.



◀ FIGURA 14–41

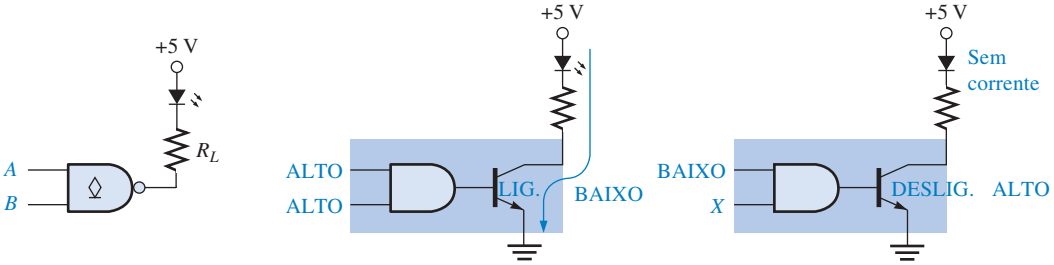
Saídas totem-pole interconectadas. Tal conexão pode provocar uma corrente excessiva através de Q_1 no dispositivo A e Q_2 no dispositivo B, não devendo ser usada nunca.

Buffers/Drivers de Coletor Aberto

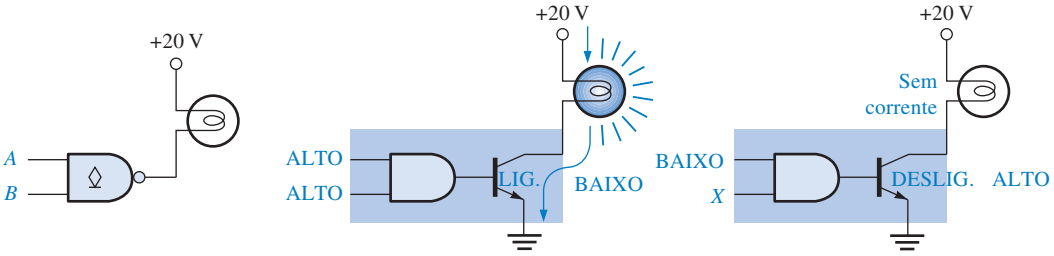
Um circuito TTL com uma saída totem-pole está limitado quanto à intensidade de corrente que pode absorver no estado BAIXO ($I_{OL(máx)}$), que é 16 mA para TTL padrão e 8 mA para TTL LS. Em muitas aplicações especiais, uma porta tem que acionar dispositivos externos, tais como LEDs, lâmpadas, ou relés que necessitam de uma corrente maior.

Devido à capacidade de operarem com tensões e correntes maiores, os circuitos com saídas de coletor aberto são geralmente usados para acionar (driver) LEDs, lâmpadas ou relés. Entretanto, as saídas totem-pole podem ser usadas, enquanto a corrente de saída necessitada pelo dispositivo externo não exceder ao valor que o driver TTL pode absorver.

Com uma porta TTL de coletor aberto, o coletor do transistor de saída é conectado a um **LED** ou lâmpada incandescente, conforme ilustra a Figura 14–42. Na parte (a) da figura o resistor de limitação (R_L) é usado para manter a corrente abaixo da corrente máxima do LED. Quando a saída da porta for nível BAIXO, o transistor de saída absorve a corrente ligando o LED. O LED é desligado quando o transistor de saída é desligado, sendo que a saída vai para nível ALTO. Um buffer de coletor aberto típico pode absorver (drenar) até 40 mA. Na parte (b) da figura, a lâmpada não necessita de um resistor de limitação por causa da resistência do filamento. Normalmente, até 30 V pode ser usado no coletor aberto, dependendo da família lógica em particular.



(a) Acionamento de um LED



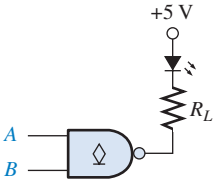
(b) Acionamento de uma lâmpada de baixa corrente

▲ **FIGURA 14–42**

Algumas aplicações de drivers de coletor aberto.

EXEMPLO 14–7

Determine o valor do resistor de limitação (R_L) no circuito de coletor aberto visto na Figura 14–43 se a corrente do LED é 20 mA. Considere uma queda de tensão de 1,5 V no LED quando polarizado diretamente e uma tensão de saída no estado BAIXO de 0,1 V na saída da porta.



▶ **FIGURA 14–43**

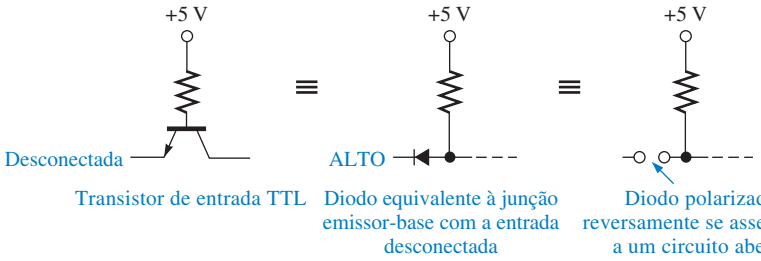
Solução $V_{R_L} = 5\text{ V} - 1{,}5\text{ V} - 0{,}1\text{ V} = 3{,}4\text{ V}$

$$R_L = \frac{V_{R_L}}{I} = \frac{3{,}4\text{ V}}{20\text{ mA}} = \mathbf{170\ \Omega}$$

Problema relacionado Determine o valor do resistor de limitação (R_L) se o LED requer 35 mA.

Entradas TTL não Usadas

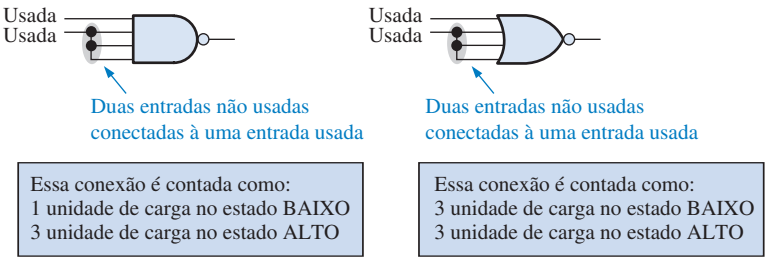
Uma entrada desconectada em uma porta TTL se comporta com se houvesse um nível ALTO porque uma entrada aberta resulta na polarização reversa da junção emissor do transistor de entrada, semelhante à situação de nível ALTO. Esse efeito é ilustrado na Figura 14–44. Entretanto, devido à sensibilidade ao ruído, é melhor não deixar entradas não usadas desconectadas (abertas). Existem algumas alternativas para conectar as entradas não usadas.



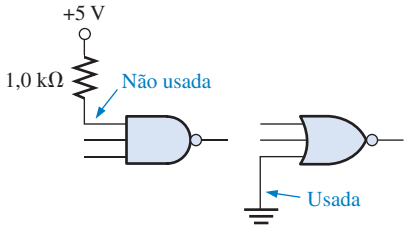
◀ **FIGURA 14–44**

Comparação de uma entrada TTL aberta e uma entrada em nível ALTO.

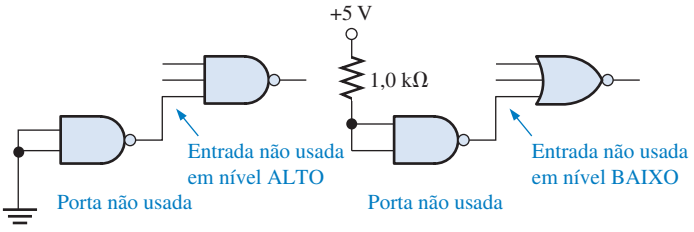
Entradas Interconectadas O método mais comum de destinação às entradas não usadas em portas é conectá-las às entradas usadas da mesma porta. Para portas AND e NAND, todas as entradas interconectadas correspondem a uma unidade de carga no estado BAIXO; porém para as porta OR e NOR, cada entrada conectada a uma outra entrada conta como uma unidade de carga em separado no estado BAIXO. No estado ALTO, cada entrada conta como uma carga em separado para todos os tipos de portas TTL. Na Figura 14–45(a) são dados dois exemplos da conexão de duas entradas não usadas à uma entrada usada.



(a) Entradas interconectadas



(b) Entradas conectadas em V_{CC} ou GND



(c) Entradas conectadas a saídas não usadas

▲ **FIGURA 14–45**

Métodos de utilização de entradas não usadas.

As portas AND e NAND apresentam apenas uma única unidade de carga não importando quantas entradas são interconectadas, ao passo que as portas OR e NOR apresentam uma unidade de carga para cada entrada interconectada. Isso ocorre devido às portas NAND usarem na entrada um transistor de múltiplos emissores; assim não importa quantas entradas estão em nível BAIXO, a corrente total para o estado BAIXO é limitada em um valor fixo. As portas NOR usam um transistor separado para cada entrada; portanto, a corrente no estado BAIXO é a soma das correntes a partir de todas as entradas interconectadas.

Entradas conectadas em V_{CC} ou G_{ND} As entradas não usadas de portas AND e NAND podem ser conectadas a V_{CC} através de um resistor de 1 kΩ. Essa conexão eleva a entrada não usada ao nível ALTO. As entradas não usadas de portas OR e NOR podem ser conectadas a GND. Esses métodos são ilustrados na Figura 14–45(b).

Entradas Conectadas à Saídas não Usadas Um terceiro método para estabelecer uma terminação às entradas não usadas pode ser apropriado em alguns casos quando uma porta não usada ou um inversor estiver disponível. A saída da porta não usada tem que estar constantemente em nível ALTO para entradas AND e NAND não usadas e constantemente em nível BAIXO para entradas não usadas de portas OR e NOR, conforme ilustra a Figura 14–45(c).

SEÇÃO 14–4 REVISÃO

1. Em qual estado de saída um circuito TTL drena corrente da carga?
2. Por que um circuito TTL fornece menos corrente para uma carga TTL do que drena dela?
3. Por que circuitos TTL com saídas totem-pole não podem ser interconectados?
4. Que tipo de circuito TTL deve ser usado para implementar a lógica AND com fios?
5. Qual tipo de circuito TTL deve ser usado para acionar uma lâmpada?
6. Uma entrada TTL desconectada se comporta como um nível BAIXO. (V ou F)

14-5 COMPARAÇÃO DE DESEMPENHO ENTRE CMOS E TTL

Nesta seção, comparamos as características operacionais principais e o desempenho de séries CMOS selecionadas com as principais séries TTL e com BiCMOS.

Ao final do estudo desta seção você deverá ser capaz de:

■ Comparar os dispositivos TTL (bipolar), BiCMOS e CMOS em termos de atraso de propagação, freqüência de clock máxima, dissipação de potência e capacidade de acionamento

Há algum tempo, as características superiores da tecnologia TTL (bipolar) comparadas com a CMOS eram a velocidade relativamente alta e a capacidade de corrente de saída. Atualmente, essas vantagens da TTL têm diminuído ao ponto da tecnologia CMOS ser freqüentemente igual ou superior em muitas áreas e ter se tornado a tecnologia de CIs dominante, embora TTL ainda esteja disponível e em uso, como sabemos. Uma família de CIs lógicos, a BiCMOS, combina a lógica CMOS com o circuito de saída TTL num esforço de combinar as vantagens de ambas.

A Tabela 14–1 fornece uma comparação do desempenho de alguns CIs de famílias lógicas.

▼ TABELA 14–1

Comparação de determinados parâmetros de desempenho de algumas famílias lógicas de CIs.

	BIPOLAR (TTL)			BiCMOS	CMOS					
					5 V			3,3 V		
	F	LS	ALS	ABT	HC	AC	AHC	LV	LVC	ALVC
Velocidade										
Atraso de propagação da porta, t_p (ns)	3,3	10	7	3,2	7	5	3,7	9	4,3	3
Freq. máxima de um FF (MHz)	145	33	45	150	50	160	170	90	100	150
Dissipação de potência por porta										
Bipolar: 50% cc (mW)	6	2,2	1,4							
CMOS: quiescente (μW)				17	2,75	0,55	2,75	1,6	0,8	0,8
Acionamento da saída										
I_{OL} (mA)	20	8	8	64	4	24	8	12	24	24

SEÇÃO 14–45 REVISÃO

1. O que é um circuito BiCMOS?
2. Em geral, qual a principal vantagem do CMOS sobre o bipolar (TTL)?

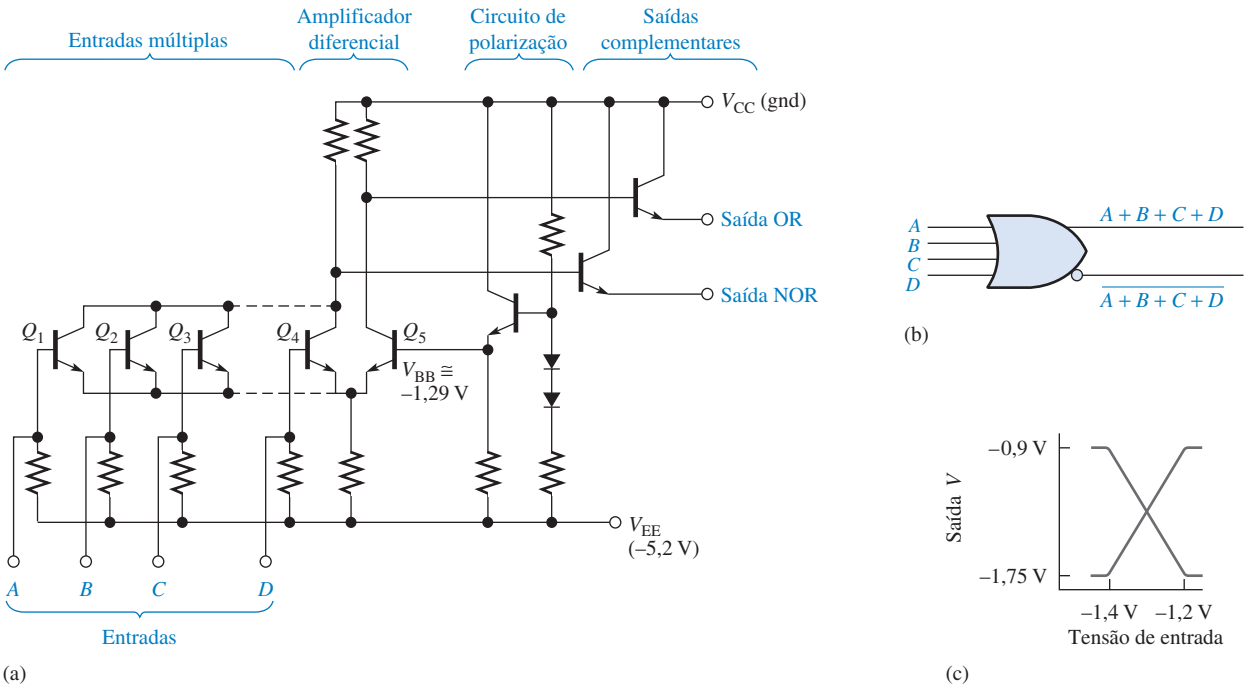
14-6 CIRCUITOS DE LÓGICA ACOPLADA PELO EMISSOR (ECL)

A lógica acoplada pelo emissor, assim como a TTL, é uma tecnologia bipolar. O circuito ECL típico consiste de um circuito de entrada com amplificador diferencial, um circuito de polarização e uma saída seguidor-de-emissor. ECL é muito mais rápida que TTL porque os transistores não operam na saturação e ela é usada na maioria das aplicações especializadas de alta velocidade.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever em que a ECL difere da TTL e da CMOS ■ Explicar as vantagens e desvantagens da ECL

Uma porta **ECL** OR/NOR é mostrada na Figura 14–46(a). As saídas seguidor-de-emissor provêem a função lógica OR e o seu complemento (NOR), conforme indicado pela Figura 14–46(b).



▲ **FIGURA 14–46**

Circuito de uma porta OR/NOR ECL.

Devido à baixa impedância de saída do seguidor-de-emissor e a alta impedância de entrada do amplificador diferencial, é possível uma operação com um fan-out alto. Nesse tipo de circuito a saturação não ocorre. A falta de saturação resulta num maior consumo de potência e uma variação de tensão limitada (menor que 1 V), mas ela permite uma maior freqüência de comutação.

O pino V_{CC} normalmente é conectado em GND e o pino V_{EE} é conectado em –5,2 V da fonte de alimentação para uma melhor operação. Observe que na Figura 14–46(c) a saída varia de um nível BAIXO de –1,75 V a um nível ALTO de –0,9 V em relação à GND. Em lógica positiva um nível 1 é o nível ALTO (menos negativo) e o nível 0 é o nível BAIXO (mais negativo).

Margem de Ruído

Conforme estudamos, a margem de ruído de uma porta é a medida de sua imunidade a flutuações de tensão indesejadas (ruído). Normalmente os circuitos ECL têm margens de ruído de cerca de 0,2 V a 0,25 V. Esses valores são menores que para TTL e tornam a ECL menos apropriada para ambientes de alto ruído.

Comparação de ECL com TTL e CMOS

A Tabela 14–2 mostra uma comparação dos principais parâmetros de desempenho para as tecnologias F, AHC e ECL.

▶ **TABELA 14–2**

Comparação de parâmetros de desempenho de series ECL com F e AHC.

	BIPOLAR (TTL) F	CMOS AHC	BIPOLAR (ECL)
Velocidade			
Atraso de propagação da porta, t_p (ns)	3,3	3,7	0,22–1
FF maximum	145	170	330–2800
Dissipação de potência por porta			
Bipolar: 50% cc	8,9 mW		25 mW–73 mW
CMOS: quiescente		2,5 µW	

SEÇÃO 14–6 REVISÃO

1. Qual é a principal vantagem da ECL sobre a TTL?
2. Cite duas desvantagens da ECL em comparação com TTL.

14-7 PMOS, NMOS E E²CMOS

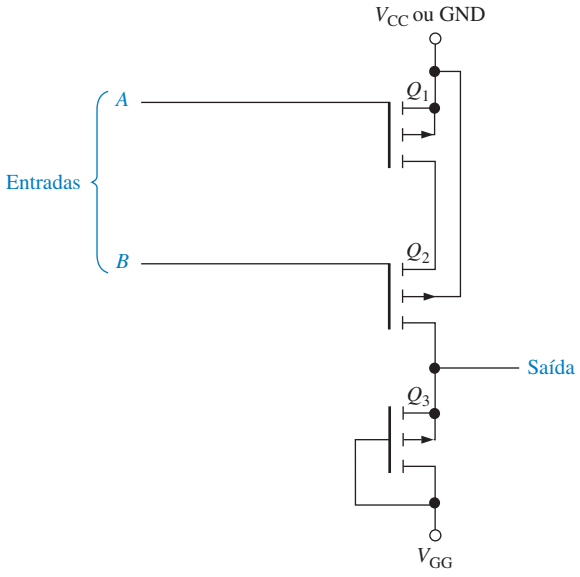
Os circuito PMOS e NMOS são usados amplamente em funções LSI, tais como grandes registradores de deslocamento, grandes memórias e microprocessadores. Tal uso é o resultado de um baixo consumo de potência e a área muito pequena requerida por um transistor MOS. A E²CMOS é usada em PLDs reprogramáveis.

Ao final do estudo desta seção você deverá ser capaz de:

■ Descrever uma porta PMOS básica ■ Descrever uma porta NMOS básica ■ Descrever uma célula E²CMOS básica

PMOS

Uma das primeiras tecnologias de circuitos **MOS** de alta densidade a ser produzida foi a **PMOS**. Ela utiliza transistores MOS canal *p* de modo enriquecimento para formar os blocos construtivos das portas básicas. A Figura 14–47 mostra uma porta PMOS básica que produz a função NOR em lógica positiva.

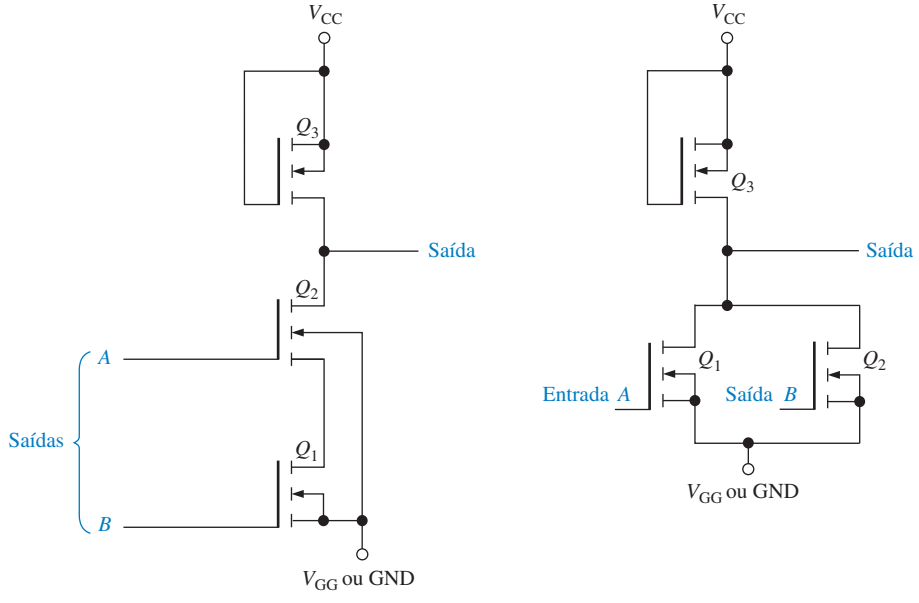


◀ **FIGURA 14–47**
Porta PMOS básica.

A operação da porta PMOS é a seguinte: A tensão de alimentação V_{GG} é uma tensão negativa e V_{CC} é uma tensão positiva ou GND (0 V). O transistor Q_3 é polarizado permanentemente para criar uma resistência dreno-fonte constante. Sua finalidade exclusiva é funcionar como um resistor de limitação de corrente. Se um nível ALTO (V_{CC}) for aplicado na entrada A ou B, então Q_1 ou Q_2 estará desligado e a saída é colocada numa tensão próxima de V_{GG}, que representa um nível BAIXO. Quando uma tensão de nível BAIXO (V_{GG}) é aplicada nas entradas A e B, Q_1 e Q_2 são ligados. Isso faz com que a saída seja colocada em nível ALTO (próximo a V_{CC}). Como uma saída em nível BAIXO ocorre quando as duas entradas estão em nível ALTO e uma saída em nível ALTO ocorre apenas quando todas as entradas estão em nível BAIXO, temos uma porta NOR.

NMOS

Os dispositivos NMOS foram desenvolvidos como uma melhoria na tecnologia de processamento. O transistor MOS canal *n* é usado em circuitos NMOS, como mostra a Figura 14–48 para as portas NAND e NOR.



(a) NAND (b) NOR

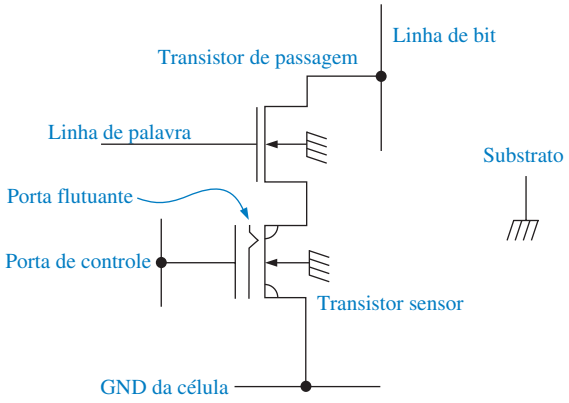
◀ **FIGURA 14–48**
Duas portas NMOS.

Na Figura 14–48(a), Q_3 faz o papel de um resistor para limitar a corrente. Quando um nível BAIXO (V_{GG} ou GND) é aplicado em uma das duas entradas, então pelo menos um dos transistores (Q_1 ou Q_2) será desligado e a saída será colocada em nível ALTO num valor próximo de V_{CC}. Quando nas entradas *A* e *B* for aplicado nível ALTO (V_{CC}), Q_1 e Q_2 entram em condução, fazendo com que a saída seja nível BAIXO. Essa operação, é claro, identifica esse circuito como uma porta NAND,

Na Figura 14–48(b), Q_3 continua fazendo o papel de um resistor. Um nível ALTO em qualquer uma das entradas liga Q_1 ou Q_2, fazendo com que a saída seja nível BAIXO. Quando as duas entradas estiverem em nível BAIXO, os dos transistores estarão desligados, fazendo com que a saída seja nível ALTO.

E²CMOS

A tecnologia **E²CMOS** (CMOS apagável eletricamente) é baseada em uma combinação das tecnologias CMOS e NMOS sendo usada em dispositivos programáveis tais como PROMs e CPLDs. Uma célula E²CMOS é construída em torno de um transistor MOS com uma porta flutuante que é carregada ou descarregada externamente por uma pequena corrente de programação. A Figura 14–49 mostra um diagrama desse tipo de célula.



▶ **FIGURA 14–49**
Célula E²CMOS.

Quando a porta flutuante é carregada pela remoção de cargas passando a ter um potencial positivo, o transistor sensor é ligado armazenando um zero binário. Quando a porta flutuante é carregada com uma carga negativa por meio da transferência de elétrons para ela, o transistor sensor é desligado armazenando um 1 binário. A porta de controle controla o potencial da porta flutuante. O transistor de passagem isola o transistor sensor do arranjo durante as operações de leitura e escrita que usam as linhas de palavra e de bit.

A célula é programada aplicando um pulso de programação na porta de controle ou na linha de bit de uma célula que foi selecionada por uma tensão na linha de palavra. Durante o ciclo de programação a célula é apagada primeiro com a aplicação de uma tensão na porta de controle para tornar a porta flutuante negativa. Isso leva o transistor sensor para o estado *desligado* (armazena nível 1). Um pulso de escrita é aplicado na linha de bit de uma célula na qual se deseja armazenar um 0. O bit armazenado na célula é lido pela detecção da presença ou ausência de um pequena corrente na célula na linha de bit. Quando um nível 1 é armazenado, não existe corrente na célula porque o transistor sensor está desligado. Quando um nível 0 é armazenado, existe uma pequena corrente na célula porque o transistor sensor está ligado. Uma vez armazenado um bit na célula, ele permanece indefinidamente a menos que a célula seja apagada ou um novo bit seja escrito na célula.

SEÇÃO 14–7 REVISÃO

1. Qual é a principal característica das tecnologias NMOS e PMOS em CIs?
2. Qual é o mecanismo do armazenamento de carga numa célula E²CMOS?

RESUMO

- *Fórmulas*:

14–1	$V_{NH} = V_{OH(mín)} - V_{IH(mín)}$	Margem de ruído em nível ALTO
14–2	$V_{NL} = V_{IL(máx)} - V_{OL(máx)}$	Margem de ruído em nível BAIXO
14–3	$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2}$	Corrente de alimentação cc média
14–4	$P_D = V_{CC}I_{CC}$	Dissipação de potência

- As saídas totem-pole TTL não podem ser interconectadas.
- As saída de coletor aberto e dreno aberto podem ser interconectadas para se obter a lógica AND com fios.
- Os dispositivos CMOS oferecem menor dissipação de potência que qualquer série TTL.
- Um dispositivo TTL não é vulnerável à descarga eletrostática (ESD) como um dispositivo CMOS.
- Devido à ESD os dispositivos têm que ser manuseados com maior cuidado.
- ECL é o tipo de circuito lógico mais rápido.
- A tecnologia E^2CMOS é usada em PROMs e outras PLDs.

TERMOS IMPORTANTES

Os termos importantes e outros termos em negrito destacados no capítulo são definidos no glossário que se encontra no final do livro.

Absorção de corrente A ação de um circuito lógico na qual ele recebe corrente na saída a partir da carga.

CMOS Semicondutor de óxido metálico complementar; um tipo de circuito lógico que usa MOSFETs (transistores de efeito de campo de semicondutor de óxido metálico) canal *n* e canal *p*.

Coletor aberto Um tipo de saída para um circuito TTL na qual o coletor do transistor de saída é deixado desconectado internamente sendo possibilitado a conexão externa de uma carga que necessita de uma tensão ou corrente relativamente alta.

Dissipação de potência O produto da tensão de alimentação cc pela corrente de alimentação cc em um circuito eletrônico.

ECL Lógica por acoplamento de emissor; uma classe de circuitos lógicos que são implementados com transistores bipolares de junção não saturados.

E^2CMOS CMOS apagável eletricamente; tecnologia de CI usada em dispositivos lógicos programáveis (PLDs).

Fan-out O número de entradas de portas equivalentes da mesma série da família que uma porta lógica pode acionar.

Fornecimento de corrente A ação de um circuito lógico na qual ele envia corrente de sua saída para a carga.

Imunidade a ruído A capacidade de um circuito lógico de rejeitar sinais indesejados (ruído).

Margem de ruído A diferença entre a saída máxima em nível BAIXO de uma porta e a entrada máxima em nível BAIXO aceitável de uma porta equivalente; é também a diferença entre a saída mínima em nível ALTO de uma porta e a entrada mínima em nível ALTO de uma porta equivalente. A margem de ruído é expressa algumas vezes com uma porcentagem da tensão de alimentação cc.

Resistor de pull-up Um resistor com uma das extremidades conectada à fonte de alimentação cc usado para manter em nível ALTO um determinado ponto de um circuito lógico quando no estado inativo.

Tempo de atraso de propagação O intervalo de tempo entre a ocorrência de uma transição na entrada e a ocorrência da correspondente transição na saída em um circuito lógico.

Totem-pole Um tipo de saída em circuitos TTL.

Tristate Um tipo de saída em circuitos lógicos que apresenta três estados: ALTO, BAIXO e alta impedância.

TTL Lógica transistor-transistor; um tipo de circuito integrado que usa transistores bipolares de junção.

Unidade de carga Uma medida de fan-out. Uma entrada de porta representa uma unidade de carga para uma porta acionadora.

AUTOTESTE

As respostas estão no final do capítulo.

1. Quando a freqüência do sinal de entrada em uma porta CMOS é aumentada, a dissipação de potência média
   (a) diminui **(b)** aumenta
   (c) não se altera **(d)** diminui exponencialmente
2. CMOS opera de forma mais confiável que TTL em um ambiente de ALTO ruído por causa de sua
   (a) margem de ruído menor. **(b)** capacitância de entrada.
   (c) margem de ruído maior. **(d)** dissipação de potência menor.
3. O manuseio correto de um dispositivo CMOS é necessário por causa de sua
   (a) construção frágil. **(b)** alta imunidade a ruído.
   (c) susceptibilidade à descarga eletrostática. **(d)** baixa dissipação de potência.
4. Qual das seguintes alternativas não é um circuito TTL?
   (a) 74F00 **(b)** 74AS00 **(c)** 74HC00 **(d)** 74ALS00
5. Uma entrada de uma porta NOR TTL aberta
   (a) se comporta como nível BAIXO. **(b)** se comporta como nível ALTO.
   (c) deve ser aterrada. **(d)** deve ser conectada a V_{CC} através de um resistor.
   (e) as alternativas (b) e (c) estão corretas. **(f)** as alternativas (a) e (c) estão corretas.
6. Uma porta TTL LS pode acionar um máximo de
   (a) 20 unidades de carga. **(b)** 10 unidades de carga.
   (c) 40 unidades de carga. **(d)** uma quantidade ilimitada de unidades de carga.
7. Se duas entradas não usadas de uma porta TTL LS são conectadas à uma entrada que aciona outra porta TTL LS, o número total de unidades de carga restantes que podem ser acionadas por essa porta é
   (a) sete **(b)** oito **(c)** dezessete **(d)** ilimitado
8. A principal vantagem da ECL sobre TTL e CMOS é
   (a) ECL é mais barata.
   (b) ECL consome menos potência.
   (c) ECL disponibiliza uma maior variedade de tipos de circuitos.
   (d) ECL é mais rápida.
9. ECL não pode ser usada em
   (a) ambientes de alto ruído.
   (b) ambientes úmidos.
   (c) aplicações de alta freqüência.
10. O mecanismo básico para armazenamento de um bit de dado em uma célula E^2CMOS é
    (a) porta de controle **(b)** dreno flutuante
    (c) porta flutuante **(d)** corrente de célula

PROBLEMAS

As respostas para os problemas de número ímpar estão no final do livro.

SEÇÃO 14–1 Caracteríticas e Parâmetros Operacionais Básicos

1. Uma certa porta lógica tem um $V_{OH(mín)} = 2{,}2$ V e aciona uma porta com um $V_{IH(mín)} = 2{,}5$ V. Essas portas são compatíveis para operarem no estado ALTO? Por quê?
2. Uma certa porta lógica tem um $V_{OL(máx)} = 0{,}45$ V e aciona uma porta com um $V_{IL(máx)} = 0{,}75$ V. Essas portas são compatíveis para operarem no estado BAIXO? Por quê?
3. Uma porta TTL tem os seguintes níveis de tensão: $V_{IH(mín)} = 2{,}25$ V, $V_{IL(máx)} = 0{,}65$ V. Considerando que ela é acionada por uma porta com $V_{OH(mín)} = 2{,}4$ V e $V_{OL(máx)} = 0{,}4$ V, quais as margens de ruído para os níveis ALTO e BAIXO.
4. Qual a máxima amplitude dos *spikes* de ruído que podem ser tolerados nas entradas da porta do Problema 3, tanto em nível ALTO quanto em nível BAIXO?

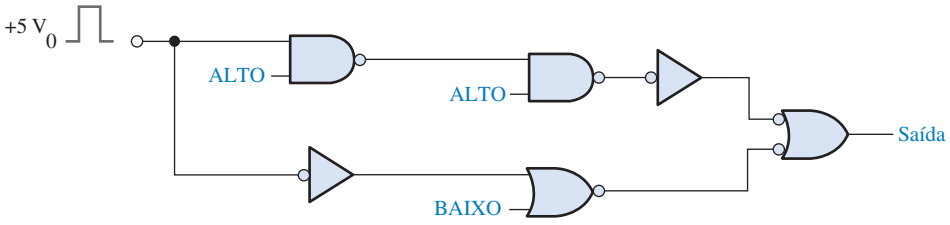
5. As especificações de tensão para três tipos de portas lógicas são dadas na Tabela 14–3. Selecione a porta que você usaria em um ambiente industrial de alto ruído.

▶ **TABELA 14–3**

	$V_{OH(MÍN)}$	$V_{OL(MÁX)}$	$V_{IH(MÍN)}$	$V_{IL(MÁX)}$
Porta *A*	2,4 V	0,4 V	2 V	0,8 V
Porta *B*	3,5 V	0,2 V	2,5 V	0,6 V
Porta *C*	4,2 V	0,2 V	3,2 V	0,8 V

6. Uma certa porta drena 2 mA de uma fonte de alimentação de +5 V no estado BAIXO e 3,5 mA no estado ALTO. Qual é a dissipação de potência no estado BAIXO? Qual é a dissipação de potência no estado ALTO? Considerando um ciclo de trabalho de 50%, qual é a dissipação de potência média?

7. Cada porta no circuito mostrado na Figura 14–50 tem um t_{PLH} e um t_{PHL} de 4 ns. Se um pulso positivo for aplicado na entrada conforme indicado, quanto tempo leva para o pulso aparecer na saída?



▶ **FIGURA 14–50**

8. Para uma certa porta, t_{PLH} = 3 ns e t_{PHL} = 2ns. Qual é o tempo de atraso de propagação médio?

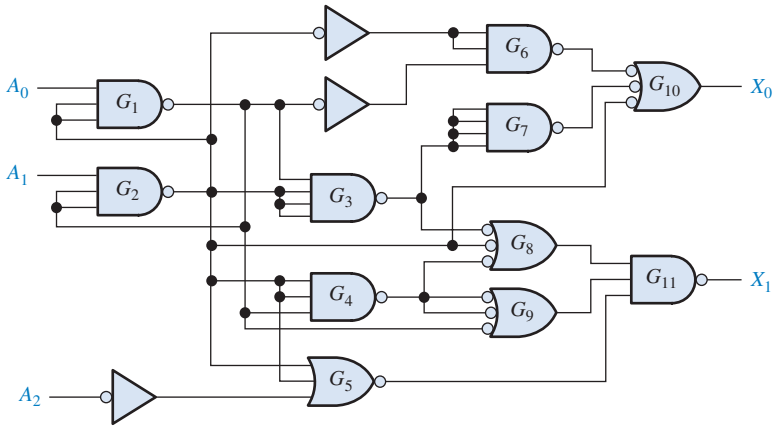
9. A Tabela 14–4 apresenta uma lista de parâmetros para três tipos de portas. Baseado em suas decisões em função do produto velocidade-potência, qual delas você selecionaria como sendo a de melhor desempenho?

▶ **TABELA 14–4**

	t_{PLH}	t_{PHL}	P_D
Porta *A*	1 ns	1,2 ns	15 mW
Porta *B*	5 ns	4 ns	8 mW
Porta *C*	10 ns	10 ns	0,5 mW

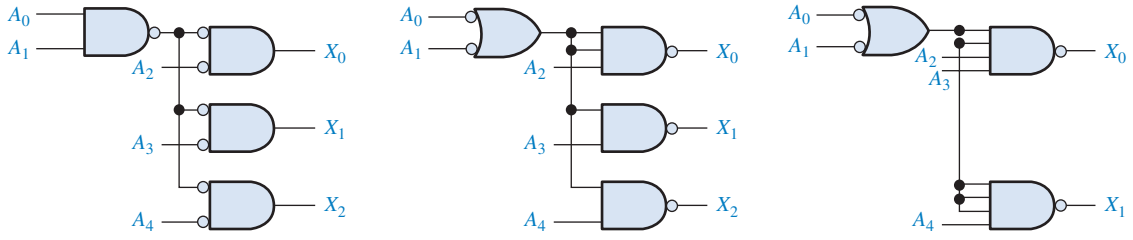
10. Se você precisasse de uma porta que operasse na maior freqüência possível, qual porta da Tabela 14-4 você selecionaria?

11. Uma porta TTL padrão tem um fan-out de 10. Alguma das portas no circuito da Figura 14–51 está sobrecarregada? Em caso afirmativo, quais?



▶ **FIGURA 14–51**

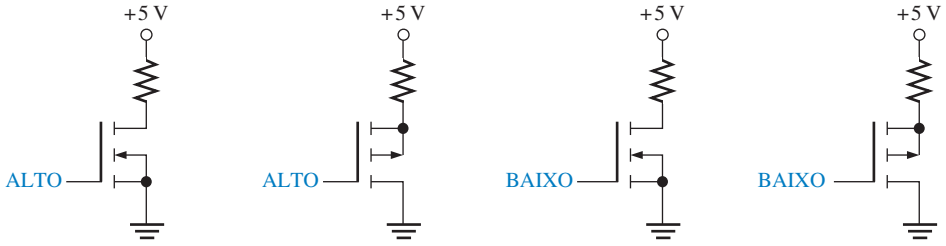
12. Qual dos circuitos com portas CMOS mostrados na Figura 14–52 pode operar com maior freqüência?



▲ FIGURA 14–52

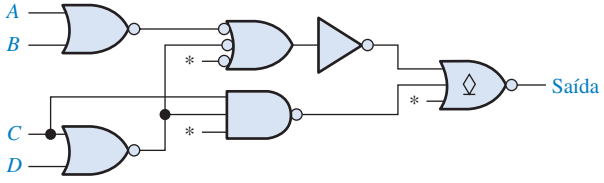
SEÇÃO 14–2 Circuitos CMOS

13. Determine o estado (ligado ou desligado) de cada MOSFET na Figura 14–53.



▶ FIGURA 14–53

14. O circuito com portas CMOS mostrado na Figura 14–54 está incompleto. Indique as mudanças que devem ser feitas.



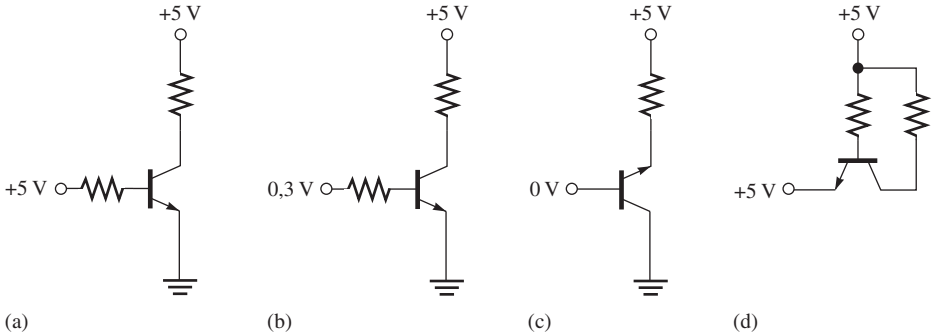
▶ FIGURA 14–54

*Entradas não usadas

15. Projete um circuito, usando portas lógicas CMOS e/ou inversores apropriados, com os quais os sinais de quatro fontes diferentes podem ser conectados para uma linha comum em diferentes instantes sem um interferir com o outro.

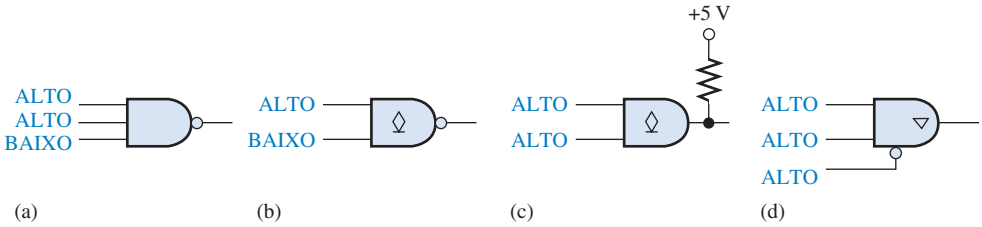
SEÇÃO 14–3 Circuitos TTL

16. Determine quais dos BJTs na Figura 14–55 estão ligados e quais estão desligados.



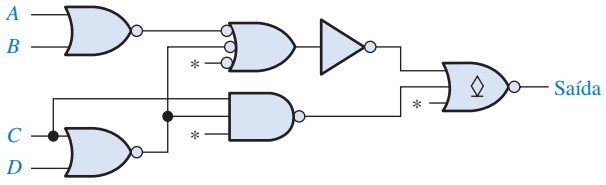
▶ FIGURA 14–55

17. Determine o estado de saída de cada porta TTL na Figura 14–56.



▶ FIGURA 14–56

18. O circuito com portas TTL mostrado na Figura 14–57 está incompleto. Indique a mudança que deve ser feita.

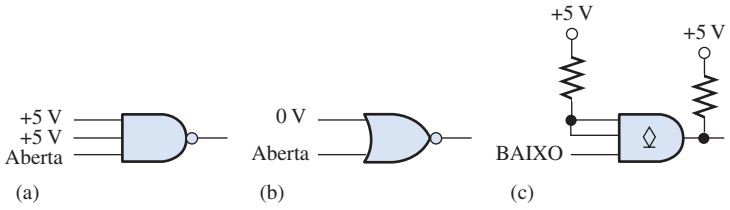


▶ FIGURA 14–57

* Entradas não usadas

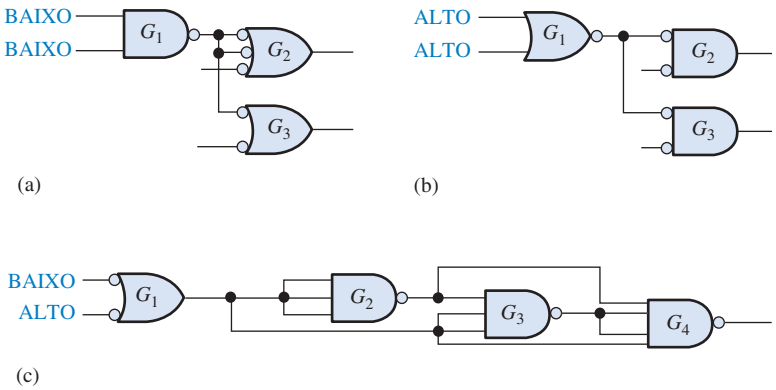
SEÇÃO 14–4 Considerações Práticas no Uso de TTL

19. Determine o nível de saída de cada porta TTL na Figura 14–58.



▶ FIGURA 14–58

20. Para cada parte da Figura 14–59, identifique se cada porta acionadora está fornecendo ou drenando corrente. Especifique a corrente máxima para fora ou para dentro da saída da porta(s) acionadora(s) em cada caso.

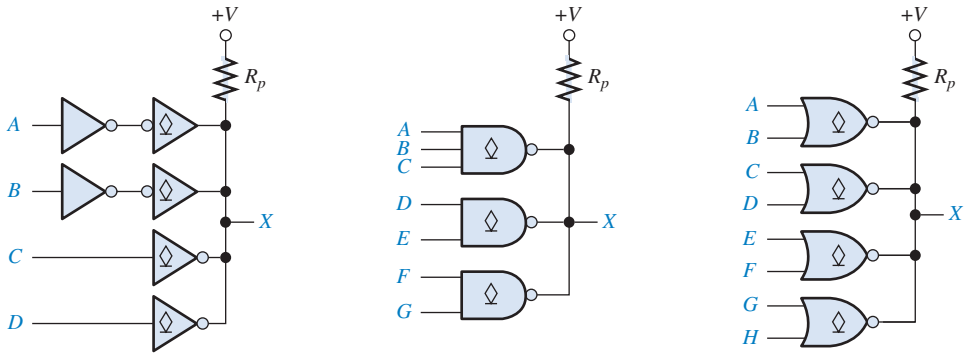


▶ FIGURA 14–59

21. Use inversores de coletor aberto para implementar as seguintes expressões lógicas:

(a) $X = \overline{A}\,\overline{B}\,\overline{C}$ **(b)** $X = A\overline{B}C\overline{D}$ **(c)** $X = ABC\overline{D}\,\overline{E}\,\overline{F}$

22. Escreva a expressão lógica para cada um dos circuitos mostrados na Figura 14–60.



▶ **FIGURA 14–60**

23. Determine o valor mínimo para o resistor de pull-up em cada circuito mostrado na Figura 14–60 se $I_{OL(máx)} = 40$ mA e $V_{OL(máx)} = 0{,}25$ V para cada porta. Considere que 10 unidades de carga TTL padrão são acionadas a partir da saída X e que a fonte de alimentação é 5 V.

24. Um certo relé necessita de 60 mA. Faça um projeto que use portas NAND de coletor aberto com $I_{OL(máx)} = 40$ mA para acionar o relé.

SEÇÃO 14–5 Comparação de Desempenho entre CMOS e TTL

25. Selecione na Tabela 14–1 a família de CI com o melhor produto velocidade-potência.

26. Determine a partir da Tabela 14–1 a família lógica que é a mais apropriada para cada um dos seguintes requisitos:

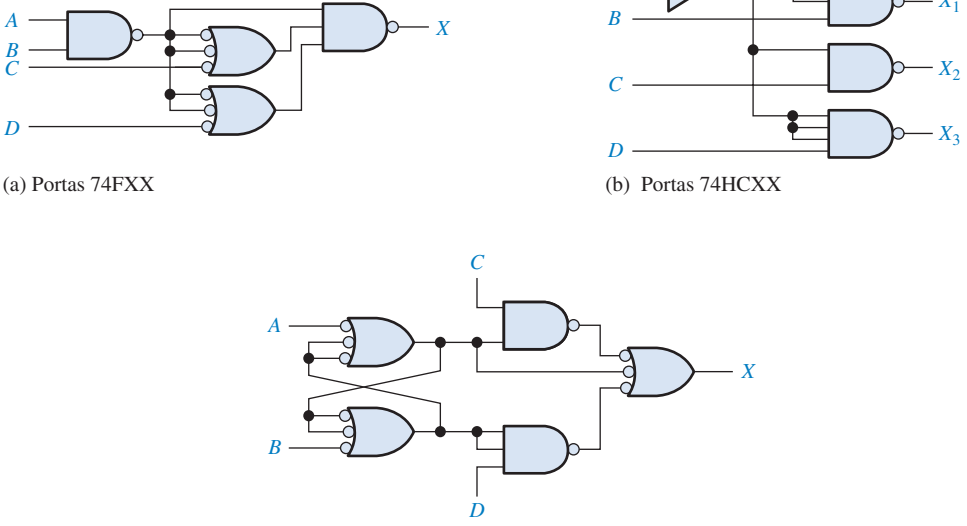
(a) menor tempo de atraso de propagação.

(b) o flip-flop mais rápido.

(c) dissipação de potência mais baixa.

(d) melhor compromisso entre velocidade e potência para uma porta lógica.

27. Determine o atraso de propagação total a partir de cada entrada para cada saída para cada um dos circuitos mostrados na Figura 14–61.



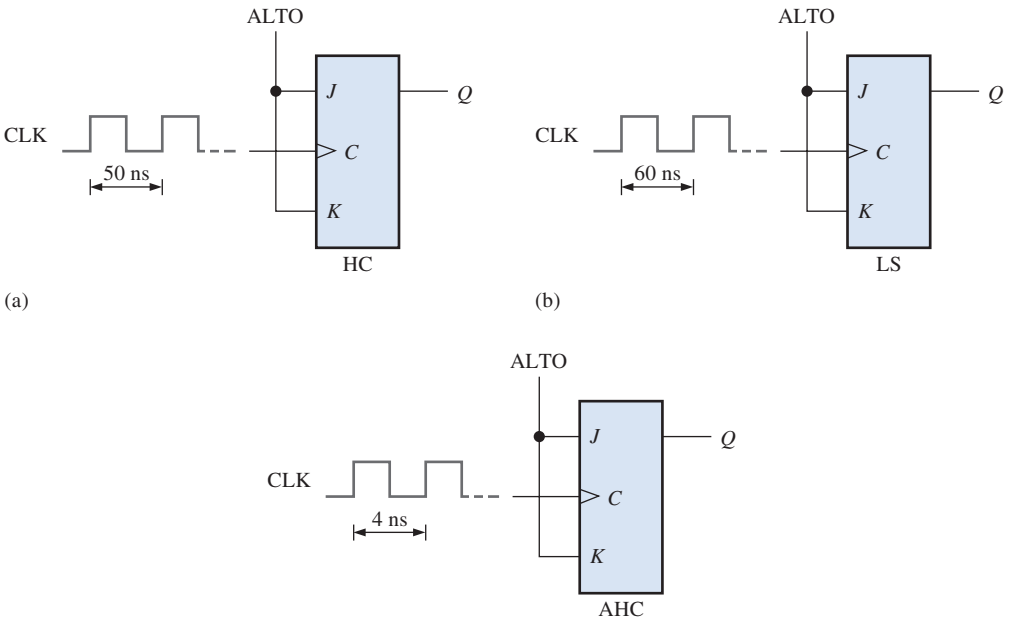
(a) Portas 74FXX

(b) Portas 74HCXX

(c) Portas 74AHCXX

▶ **FIGURA 14–61**

28. Um dos flip-flops na Figura 14–62 pode ter uma saída errada. Qual deles apresenta erro, caso tenha algum, e por quê?



▶ **FIGURA 14–62**

SEÇÃO 14–6 Circuitos de Lógica Acoplada pelo Emissor (ECL)

29. Qual a diferença básica entre os circuitos ECL e TTL?

30. Selecione dentre ECL, HCMOS ou a série TTL apropriada para cada um dos requisitos a seguir:
(a) maior velocidade.
(b) menor potência.
(c) melhor compromisso entre alta velocidade e baixa potência (produto velocidade-potência).

RESPOSTAS

SEÇÕES DE REVISÃO

SEÇÃO 14–1 Características e Parâmetros Operacionais Básicos

1. V_{IH}: Tensão de entrada em nível ALTO; V_{IL}: Tensão de entrada em nível BAIXO; V_{OH}: Tensão de saída em nível ALTO; VOL: Tensão de saída em nível BAIXO
2. Quanto maior o valor da margem de ruído melhor.
3. A porta *B* pode operar com uma freqüência maior.
4. A carga excessiva reduz a margem de ruído de uma porta.

SEÇÃO 14–2 Circuitos CMOS

1. Os MOSFETs são usados na lógica CMOS.
2. Um circuito de saída complementar consiste de dois MOSFETs, um canal *n* e outro canal *p*.
3. Devido a descarga eletrostática poder danificar os dispositivos CMOS.

SEÇÃO 14–3 Circuitos TTL

1. Falso, o BJT *npn* está desligado.
2. O estado ligado de um BJT é uma chave fechada; o estado desligado é uma chave aberta.
3. Os tipos de saídas TTL são totem-pole e coletor aberto.
4. A lógica tristate provê um estado de alta impedância, no qual a saída é desconectada do restante do circuito.

SEÇÃO 14–4 Considerações Práticas no Uso de TTL

1. A corrente drenada ocorre no estado BAIXO da saída.
2. A corrente fornecida é menor que a drenada porque uma carga TTL se assemelha a um diodo polarizado reversamente no estado ALTO.
3. Os transistores totem-pole não podem operar com a corrente que existe quando uma saída tenta ir para o nível ALTO e a outra tenta ir para o nível BAIXO.
4. A lógica AND com fios tem que usar saídas de coletor aberto.
5. Para acionar uma lâmpada a saída tem que ser de coletor aberto.
6. Falso, uma entrada TTL desconectada geralmente se comporta como nível ALTO.

SEÇÃO 14–5 Comparação de Desempenho Entre CMOS e TTL

1. BiCMOS usa transistores bipolares nos circuitos de entrada e saída e CMOS no circuito intermediário.
2. CMOS tem uma dissipação de potência menor que TTL.

SEÇÃO 14–6 Circuitos de Lógica Acoplada pelo Emissor (ECL)

1. ECL é mais rápida que TTL.
2. ECL tem mais consumo e menos margem de ruído que TTL.

SEÇÃO 14–7 PMOS, NMOS e E^{2}CMOS

1. NMOS e PMOS são de alta densidade.
2. A porta flutuante é o mecanismo de armazenamento de carga em uma célula E^{2}CMOS.

PROBLEMAS RELACIONADOS APRESENTADOS NOS EXEMPLOS

14–1. CMOS **14–2.** 10,75 μW

14–3. $I_{T(fonte)} = 5(20\ \mu A) = 100\ \mu A$

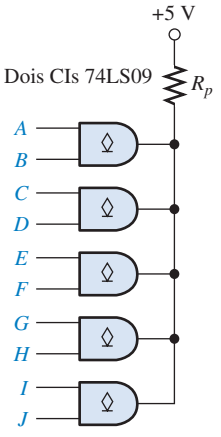
$I_{T(drenagem)} = 5(-0{,}4\ mA) = -2{,}0\ mA$

14–4. Fan-out = 20

14–5. $X = (\overline{AB})(\overline{CD})(\overline{EF})(\overline{GH}) = (\overline{A} + \overline{B})(\overline{C} + \overline{D})(\overline{E} + \overline{F})(\overline{G} + \overline{H})$

14–6. Veja a figura 14–63. **14–7.** $R_L = 97\ \Omega$

▶ FIGURA 14–63



AUTOTESTE

1. (b) **2.** (c) **3.** (c) **4.** (c) **5.** (e)

6. (a) **7.** (c) **8.** (d) **9.** (a) **10.** (c)

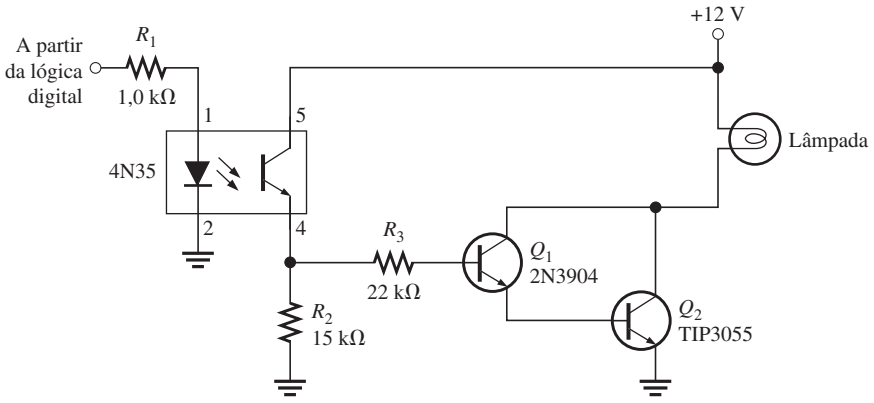
A: Conversões

DECIMAL	BCD(8421)	OCTAL	BINÁRIO	DECIMAL	BCD(8421)	OCTAL	BINÁRIO	DECIMAL	BCD(8421)	OCTAL	BINÁRIO
0	0000	0	0	34	00110100	42	100010	68	01101000	104	1000100
1	0001	1	1	35	00110101	43	100011	69	01101001	105	1000101
2	0010	2	10	36	00110110	44	100100	70	01110000	106	1000110
3	0011	3	11	37	00110111	45	100101	71	01110001	107	1000111
4	0100	4	100	38	00111000	46	100110	72	01110010	110	1001000
5	0101	5	101	39	00111001	47	100111	73	01110011	111	1001001
6	0110	6	110	40	01000000	50	101000	74	01110100	112	1001010
7	0111	7	111	41	01000001	51	101001	75	01110101	113	1001011
8	1000	10	1000	42	01000010	52	101010	76	01110110	114	1001100
9	1001	11	1001	43	01000011	53	101011	77	01110111	115	1001101
10	00010000	12	1010	44	01000100	54	101100	78	01111000	116	1001110
11	00010001	13	1011	45	01000101	55	101101	79	01111001	117	1001111
12	00010010	14	1100	46	01000110	56	101110	80	10000000	120	1010000
13	00010011	15	1101	47	01000111	57	101111	81	10000001	121	1010001
14	00010100	16	1110	48	01001000	60	110000	82	10000010	122	1010010
15	00010101	17	1111	49	01001001	61	110001	83	10000011	123	1010011
16	00010110	20	10000	50	01010000	62	110010	84	10000100	124	1010100
17	00010111	21	10001	51	01010001	63	110011	85	10000101	125	1010101
18	00011000	22	10010	52	01010010	64	110100	86	10000110	126	1010110
19	00011001	23	10011	53	01010011	65	110101	87	10000111	127	1010111
20	00100000	24	10100	54	01010100	66	110110	88	10001000	130	1011000
21	00100001	25	10101	55	01010101	67	110111	89	10001001	131	1011001
22	00100010	26	10110	56	01010110	70	111000	90	10010000	132	1011010
23	00100011	27	10111	57	01010111	71	111001	91	10010001	133	1011011
24	00100100	30	11000	58	01011000	72	111010	92	10010010	134	1011100
25	00100101	31	11001	59	01011001	73	111011	93	10010011	135	1011101
26	00100110	32	11010	60	01100000	74	111100	94	10010100	136	1011110
27	00100111	33	11011	61	01100001	75	111101	95	10010101	137	1011111
28	00101000	34	11100	62	01100010	76	111110	96	10010110	140	1100000
29	00101001	35	11101	63	01100011	77	111111	97	10010111	141	1100001
30	00110000	36	11110	64	01100100	100	1000000	98	10011000	142	1100010
31	00110001	37	11111	65	01100101	101	1000001	99	10011001	143	1100011
32	00110010	40	100000	66	01100110	102	1000010				
33	00110011	41	100001	67	01100111	103	1000011				

Potências de Dois

2^n	n	2^{-n}
1	0	1,0
2	1	0,5
4	2	0,25
8	3	0,125
16	4	0,062 5
32	5	0,031 25
64	6	0,015 625
128	7	0,007 812 5
256	8	0,003 906 25
512	9	0,001 953 125
1 024	10	0,000 976 562 5
2 048	11	0,000 488 281 25
4 096	12	0,000 244 140 625
8 192	13	0,000 122 070 312 5
16 384	14	0,000 061 035 156 25
32 768	15	0,000 030 517 578 125
65 536	16	0,000 015 258 789 062 5
131 072	17	0,000 007 629 394 531 25
262 144	18	0,000 003 814 697 265 625
524 288	19	0,000 001 907 348 632 812 5
1 048 576	20	0,000 000 953 674 316 406 25
2 097 152	21	0,000 000 476 837 158 203 125
4 194 304	22	0,000 000 238 418 579 101 562 5
8 388 608	23	0,000 000 119 209 289 550 781 25
16 777 216	24	0,000 000 059 604 644 775 390 625
33 554 432	25	0,000 000 029 802 322 387 695 312 5
67 108 864	26	0,000 000 014 901 161 193 847 656 25
134 217 728	27	0,000 000 007 450 580 596 923 828 125
268 435 456	28	0,000 000 003 725 290 298 461 914 062 5
536 870 912	29	0,000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0,000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0,000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0,000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0,000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0,000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0,000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0,000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0,000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0,000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0,000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0,000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0,000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0,000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0,000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0,000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0,000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0,000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0,000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0,000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0,000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0,000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0,000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0,000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0,000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0,000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0,000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0,000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0,000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0,000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0,000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0,000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0,000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0,000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0,000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0,000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0,000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0,000 000 000 000 000 000 013 552 527 156 088 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0,000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0,000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0,000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0,000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0,000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0,000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

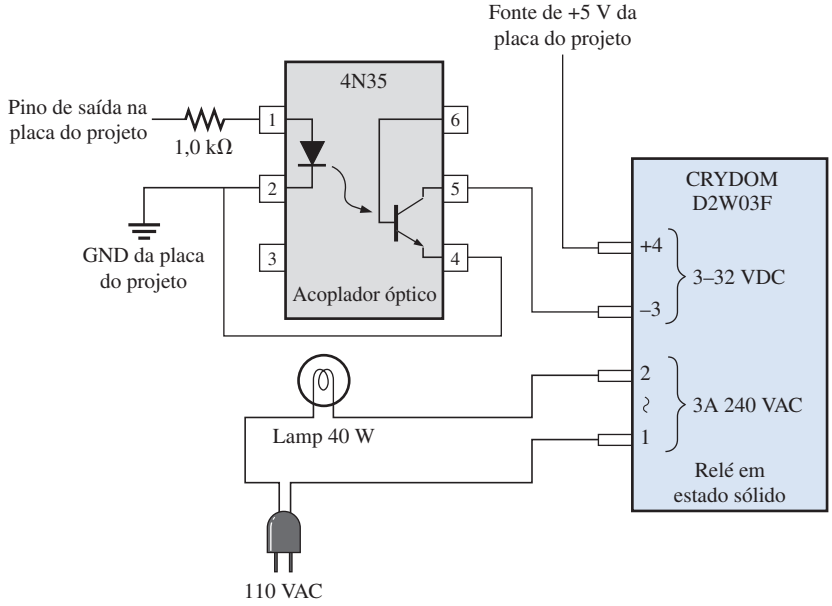
B: Interface para um Semáforo Luminoso

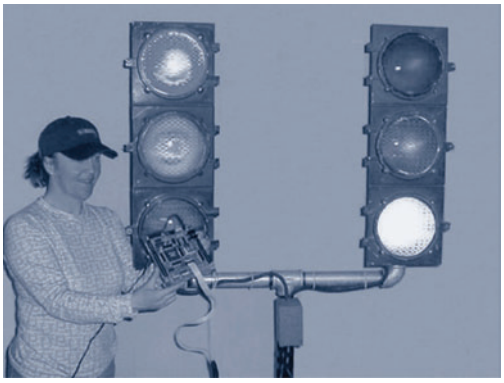


A placa de desenvolvimento com um CPLD programado e uma placa de interface que aciona o semáforo no laboratório. Cortesia de Dave Buchla.

▲ **FIGURA B–1**

Circuito de interface usado com o modelo de semáforo. Um circuito para o acionamento de cada lâmpada.





Estudande segurando a placa de desenvolvimento com a CPLD programada num semáforo real no laboratório. Os circuitos de interface estão dentro da caixa metálica montada no suporte das lâmpadas. Cortesia de Doug Joksch

▲ **FIGURA B–2**

Circuito de interface usado com o projeto proposto para o semáforo. Um circuito aciona uma lâmpada.

Glossário

Aceitador Um dispositivo de recepção em um barramento.

ADC flash Um conversor analógico-digital simultâneo.

Adição Booleana A operação OR na álgebra Booleana.

Adjacência Características das células em um mapa de Karnaugh na qual existe uma mudança de uma única variável de uma célula para outra próxima em um dos seus quatro lados.

Alfanumérico Consiste de numerais, letras e outros caracteres.

Álgebra Booleana A matemática dos circuitos lógicos.

Aliasing O efeito criado quando um sinal é amostrado com pelo menos duas vezes a freqüência do sinal. O *aliasing* cria freqüências indesejadas que interferem com a freqüência do sinal.

ALU Unidade lógica e aritmética; o principal elemento de processamento de um microprocessador que realiza operações lógicas e aritméticas.

Amostragem O processo de obtenção de um número suficiente de valores discretos em pontos da forma de onda que definem o formato da forma de onda.

Amplificador operacional (amp-op) Um dispositivo com duas entradas diferenciais que tem um ganho muito ALTO, impedância de entrada muito alta e impedância de entrada muito baixa.

Amplitude Na forma de onda de um pulso, a altura ou o valor máximo de um pulso quando medido a partir de um nível baixo.

Análise de defeito A técnica ou processo de identificação, isolação e correção sistemática de um defeito num circuito ou sistema.

Analógico Apresenta valores contínuos no tempo, em oposição ao que apresenta um conjunto de valores discretos.

AND negativa A operação dual de uma porta NOR quando as entradas são ativas em nível BAIXO.

AND Uma função lógica básica na qual uma saída verdadeira (nível ALTO) ocorre apenas quando todas as condições de entrada são verdadeiras (nível ALTO).

ANSI *American National Standards Institute*. (Instituto de padronização americano).

Antifusível Um tipo de conexão programável não volátil de PLD que pode ser deixada aberta ou pode ser colocada em curto-circuito uma vez conforme determinado pelo programa.

Arbitragem de barramento O processo que evita que duas fontes usem um barramento ao mesmo tempo.

Armazenamento A capacidade de um dispositivo digital de reter bits; o processo de retenção de dados digitais para uso posterior.

Arquitetura A unidade VHDL que descreve a operação interna de uma função lógica; o arranjo funcional interno dos elementos que dão a um dispositivo as suas características operacionais particulares.

Arranjo AND Um arranjo de portas AND consiste de uma matriz de interconexões programáveis.

Arranjo de interconexões programáveis (PIA) Um arranjo que consiste de condutores distribuídos no chip CPLD através do qual as conexões a partir das macrocélulas em cada LAB podem ser feitas.

Arranjo de memória Um arranjo de células de memória organizadas em linhas e colunas.

Arranjo de registrador Um conjunto de posições de armazenamento temporário dentro de um microprocessador para manter os dados e endereços que precisam serem acessados rapidamente pelo programa.

Arranjo Em um PLD, uma matriz formada por linhas de termos-produto e colunas de linhas de entrada com uma célula programável em cada junção. Em VHDL, um arranjo é um conjunto ordenado de itens individuais denominados elementos com um único nome identificador.

ASCII Código Padrão Americano para Troca de Informações; o código alfanumérico mais amplamente usado.

Assemblador Um programa que converte mnemônicos originário do inglês em código de máquina.

Assíncrono Que não ocorre no mesmo instante.

Astável Não tem estado estável. Um multivibrador astável oscila entre dois estados quase estáveis.

Barramento 488 da IEEE O mesmo que GPIB (barramento de interface de propósito geral); um padrão de barramento paralelo usado amplamente para interface de instrumentos de medição e teste.

Barramento de controle Um conjunto de vias condutoras que conectam a CPU a outras partes do computador para coordenar sua operação e se comunicar com dispositivos externos.

Barramento de dados Um conjunto bidirecional de vias condutoras através das quais os dados ou códigos de instrução são transferidos para o microprocessador ou os resultados das operações são enviadas para fora do microprocessador.

Barramento de endereço Um grupo de condutores que interconectam o processador e a memória, ou ainda outros dispositivos externos, nos quais os códigos de endereços são enviados.

Barramento ISA Barramento de arquitetura padrão industrial; um padrão de barramento paralelo interno.

Barramento local Um barramento interno que conecta o microprocessador à memória cache, à memória principal, ao coprocessador e ao controlador de barramento PCI.

Barramento PCI Barramento de interconexão de controle de periférico; um padrão de barramento paralelo interno.

Barramento Um conjunto de interconexões que faz a interface com um ou mais dispositivos baseado em especificações padronizadas.

Base Uma das três regiões de um transistor bipolar de junção.

BCD Decimal codificado em binário; um código digital no qual cada um dos dígitos decimais, de 0 a 9, é representado por um grupo de quatro bits.

Bed-of-nails Um método para teste automático de placas de circuito impresso no qual a placa é colocada em um acessório de fixação que se assemelha a uma "cama de unhas" que faz contato com os pontos de teste.

Bidirecional Tem duas direções. Num registrador de deslocamento bidirecional, o dado armazenado pode ser deslocado para a direita ou para a esquerda.

Biestável Tem dois estados estáveis. Flip-flops e latches são multivibradores biestáveis.

Binário Que apresenta dois valores ou estados; descreve um sistema de numeração que tem base dois e utiliza os dígitos 0 e 1.

BIOS Sistema de entrada/saída básico; um conjunto de programas em ROM que faz a interface de dispositivos I/O em um sistema de computador.

Bipolar Relativo a dois portadores de carga opostos dentro da estrutura de um transistor.

Bit de paridade Um bit anexado em cada grupo de bits de informação para tornar o número total de 1s ímpar ou par para cada grupo de bits.

Bit de sinal O bit mais à esquerda de um número binário que indica se o número é positivo (0) ou negativo (1).

Bit mais significativo (MSB) O bit mais à esquerda em um código ou número binário.

Bit menos significativo (LSB) Bit menos significativo; o bit mais à direita num código ou número inteiro binário.

Bit Um dígito binário, que pode ser 1 ou 0.

BIU Unidade de interface de barramento; a parte da CPU que faz a interface com os barramentos do sistema e busca instruções, lê operandos e escreve resultados.

BJT Transistor bipolar de junção; um dispositivo semicondutor usado para comutação ou amplificação. Um BJT tem duas junções, a junção base-emissor e a junção base-coletor.

Bloco de arranjo lógico (LAB) Um grupo de macrocélulas que podem ser interconectadas com outros LABs ou outros I/Os usando um arranjo de interconexões programáveis; também chamado de bloco funcional.

Borda anterior A primeira transição de um pulso.

Borda posterior A segunda transição de um pulso.

Boundary scan Um método de teste interno de um PLD baseado no padrão JTAG (padrão 1149.1 da IEEE).

Buffer Um circuito que evita a carga de uma entrada ou saída.

Busca Processo da CPU no qual uma instrução é obtida da memória; um estágio da operação pipeline do DSP na qual as instruções são obtidas a partir da memória de programa.

Byte Um grupo de oito bits.

Capacidade de palavras O número de palavras que uma memória pode armazenar.

Capacidade de processamento A velocidade média na qual um programa é executado.

Capacidade O número total de unidades de dados (bits, nibbles, bytes, words) que um memória pode armazenar.

Caractere Um símbolo, letra ou numeral.

Carga unitária Uma medida do fan-out. Uma entrada de porta representa uma unidade de carga para a saída de uma porta lógica dentro da mesma família de CIs.

Carga A entrada de dados em um registrador de deslocamento.

Carry antecipado Um método de adição binária por meio do qual os carries dos estágios somadores anteriores são antecipados, eliminando assim os atrasos de propagação do carry.

Carry ondulante Um método de adição binária no qual o carry de saída de cada somador se torna o carry de entrada do próximo somador de ordem maior.

Carry O dígito gerado quando a soma de dois dígitos binários excede a 1.

Cartucho jaz Um dispositivo de armazenamento magnético; discos rígidos encapsulados em um cartucho plástico rígido com capacidades de armazenamento de 1 GB ou 2 GB.

Cascata Conexão da saída de um dispositivo para a entrada de um dispositivo similar, possibilitando que um dispositivo acione o outro para expandir a capacidade operacional.

CCD Dispositivo de acoplamento de carga; um tipo de memória semicondutora que armazena dados na forma de pacotes de carga acessados serialmente.

CD-R CD gravável; um dispositivo de armazenamento óptico na forma de disco no qual os dados podem ser armazenados uma vez.

CD-ROM Um dispositivo de armazenamento óptico na forma de disco no qual os dados são previamente armazenados podendo serem apenas lidos.

CD-RW CD regravável; um disco óptico de armazenamento no qual os dados podem ser escritos e reescritos muitas vezes.

Célula Um elemento único de armazenamento numa memória.

Ciclo de trabalho A relação da largura do pulso com o período expresso em porcentagem.

Circuito integrado (CI) Um tipo de circuito no qual todos os componentes são integrados num único chip de material semicondutor de tamanho extremamente reduzido.

Circuito seqüencial Um circuito digital cujos estados lógicos seguem uma seqüência temporal especificada.

Circuito Um arranjo de componentes elétricos e/ou eletrônicos interconectados de forma a realizarem uma função específica.

CLB Bloco lógico configurável; uma unidade de lógica em um FPGA que é constituído de múltiplos módulos lógicos pequenos e uma interconexão programável local que é usada para conectar módulos lógicos dentro do CLB.

Clear Uma entrada assíncrona usada para resetar um flip-flop (faz a saída $Q = 0$).

Clock O sinal de temporização básico num sistema digital; uma forma de onda periódica na qual cada intervalo entre pul-

sos é igual ao tempo de um bit; a entrada de disparo de um flip-flop.

CMOS Semicondutor de óxido metálico complementar; uma classe de circuitos lógicos integrados implementados com um tipo de transistor de efeito de campo.

Codec Uma combinação de codificador e decodificador.

Codificador de prioridade Um codificador no qual apenas o dígito de entrada de valor mais alto é codificado e qualquer outra entrada ativa é ignorada.

Codificador Um circuito (dispositivo) digital que converte informação para uma forma codificada.

Código de máquina As instruções binárias básicas entendidas pelo processador.

Código de operação O código que representa uma instrução para um microprocessador em particular; um mnemônico.

Código Gray Um código digital sem peso caracterizado por uma alteração única de bit entre números de códigos adjacentes em uma seqüência.

Código Hamming Um tipo de código de correção de erro.

Código realocável Um programa que pode ser movido para qualquer lugar dentro do espaço de memória sem alterar o código base.

Código Um conjunto de bit arranjados em um padrão único e usado para representar informações tais como números, letras e outros símbolos; declarações de um programa em VHDL.

Coletor aberto Um tipo de saída para um circuito TTL na qual o coletor do transistor de saída é deixado desconectado internamente sendo possibilitado a conexão externa de uma carga que necessita de uma tensão ou corrente relativamente alta.

Coletor Uma das três regiões em um transistor bipolar.

Comparador Um circuito digital que compara a magnitude de duas quantidades e produz uma saída que indica a relação das quantidades.

Compilador Uma programa aplicativo de um pacote de software de desenvolvimento que controla o processo do fluxo do projeto e traduz o código fonte em código objeto num formato que pode ser testado logicamente ou transferido para o dispositivo de destino.

Complemento O inverso ou o oposto de um número. Em álgebra Booleana, a função inversa é expressa com uma barra sobre a variável. O complemento de 1 é 0 e vice-versa.

Componente Uma característica VHDL que pode ser usada para predefinir uma função lógica para ser usada uma ou mais vezes num ou mais programas.

Conexão em cascata Para conexão "fim-a-fim" como quando vários contadores são conectados a partir da saída de contagem final de um contador na entrada de habilitação do próximo.

Consulta O processo de verificação de uma série de dispositivos periféricos para determinar se algum deles necessita de um serviço da CPU.

Contador assíncrono Um tipo de contador no qual cada estágio é disparado a partir da saída do estágio anterior.

Contador crescente/decrescente Um contador que pode contar nas duas direções através de uma certa seqüência.

Contador de década Um contador digital que tem dez estados.

Contador em anel Um registrador no qual um certo padrão de 1s e 0s é continuamente recirculado.

Contador Johnson Um tipo de registrador no qual um padrão específico de 1s e 0s pré-armazenados é deslocado através dos estágios, criando uma seqüência única de padrão de bits.

Contador ondulante Um contador assíncrono.

Contador síncrono Um tipo de contador no qual cada estágio é disparado pelo mesmo pulso.

Contador Um circuito digital capaz de contar eventos, tal como pulsos, percorrendo uma seqüência de estados binários.

Contagem final O estado final na seqüência de um contador.

Contenção de barramento Uma condição adversa que poderia ocorrer se dois ou mais dispositivos tentam comunicar ao mesmo tempo em um barramento.

Contíguo Unido, interconectado.

Controlador Um dispositivo que pode especificar cada um dos outros dispositivos em um barramento com "falante" ou "ouvinte" para fins de transferência de dados.

Conversão analógico-digital (A/D) O processo de conversão de um sinal analógico para a forma digital.

Conversão digital-analógico (D/A) O processo de converter uma seqüência de códigos digitais no formato analógico.

Conversor analógico-digital (ADC) Um circuito usado para converter um sinal analógico para o formato digital.

Conversor digital-analógico (DAC) um circuito usado para converter a representação digital de um sinal analógico de volta para o formato analógico.

Corrida Uma condição de um circuito lógico no qual a diferença nos tempos de propagação através de dois ou mais percursos no circuito pode produzir uma saída errada.

CPLD Dispositivo lógico programável complexo que consiste basicamente de múltiplos arranjos SPLD com interconexões programáveis.

CPU Unidade central de processamento; a principal parte de um computador responsável pelo controle e processamento de dados; o núcleo de um DSP que processa as instruções do programa.

Cross-assembler Um programa que traduz um programa em linguagem assembly de um tipo de microprocessador para a linguagem assembly de um outro tipo de microprocessador.

Dados Informação na forma numérica, alfabética ou outra forma.

DAT Fita de áudio digital; um tipo de fita magnética.

DCE Equipamento de comunicação de dados.

Debug Um código dentro do DOS que permite várias operações em arquivos e inclui um assemblador primitivo; operação de eliminação de um problema em hardware ou software.

Década Caracterizado por dez estados ou valores.

Decimal Descreve um sistema de numeração com base dez.

Decodificação Um estágio da operação pipeline do DSP no qual as instruções são associadas a unidades funcionais e são decodificadas.

Decodificador Um circuito digital que converte informação codificada numa forma familiar ou não-codificada.

Decremento Diminuição do estado binário de um contador de uma unidade.

Demultiplexador (DEMUX) Um circuito que comuta dados digitais de uma linha de entrada para diversas linhas de saída numa seqüência temporal especificada.

Deslocamento Para movermos dados binários de estágio para estágio dentro de um registrador de deslocamento ou outro dispositivo de armazenamento ou mover dados em binário para dentro ou para forma de um dispositivo.

Detecção de erro O processo de detecção de erros de bit em um código digital.

Diagrama de estados Uma ilustração gráfica de uma seqüência de estados ou valores.

Diagrama de temporização Um gráfico de formas de onda digitais que mostra a relação temporal de duas ou mais formas de onda e como cada forma de onda varia em relação às outras.

Diferença O resultado de uma subtração.

Digital Relativo a dígitos ou quantidades discretas; tem um conjunto de valores discretos.

Dígito Um símbolo usado para expressar uma quantidade.

DIMM Módulo de memória com pinos em duas linhas.

Diodo Um dispositivo semicondutor que conduz corrente em apenas uma direção.

DIP *Dual in-line package*; um tipo de encapsulamento de CI cujos pinos têm que passar através de furos para o outro lado da placa de circuito impresso.

Disco magneto-óptico Um dispositivo de armazenamento que usa eletromagnetismo e um feixe laser para ler e escrever dados.

Disco rígido Um dispositivo de armazenamento magnético; tipicamente, uma pilha de um ou mais discos rígidos confinados num invólucro hermeticamente selado.

Disco zip Um tipo de armazenamento magnético; um disco flexível com uma capacidade de 100 MB encapsulado em um cartucho de plástico rígido tendo aproximadamente o tamanho de um disquete.

Disparo Um pulso usado para iniciar uma mudança de estado em um circuito lógico.

Dispositivo destino Um PLD montado num equipamento de programação ou placa de desenvolvimento no qual se faz a transferência (download) do projeto lógico na forma de software; o dispositivo lógico programável a ser programado.

Disquete Um dispositivo de armazenamento magnético; um disco flexível com um diâmetro de 3,5 polegadas e uma capacidade de armazenamento de 1,44 MB interno a um envelope de plástico rígido.

Dissipação de potência O produto da tensão de alimentação cc pela corrente de alimentação cc em um circuito eletrônico. A quantidade de energia requerida por um circuito.

Dividendo A quantidade pela qual está sendo dividida uma outra quantidade numa operação de divisão.

Divisor A quantidade que é dividida pelo valor do dividendo numa operação de divisão.

DLT Fita linear digital; um tipo de fita magnética.

DMA Acesso direto à memória; um método de fazer uma interface direta de um dispositivo periférico com a memória sem utilizar o controle da CPU.

Domínio Todas as variáveis na expressão Booleana.

"Don't care" Uma combinação de literais que não podem ocorrer e podem ser usadas como um 1 ou um 0 no mapa de Karnaugh para simplificação.

Download O processo no fluxo de um projeto no qual o projeto lógico é transferido do software para o hardware.

DRAM BEDO Memória de acesso aleatório com saída de dados estendida em rajada.

DRAM EDO Memória de acesso aleatório dinâmica com saída de dados estendidas.

DRAM FPM Memória de acesso aleatório dinâmica de modo de página rápida.

DRAM Memória de acesso aleatório dinâmica; um tipo e memória semicondutora de leitura/escrita que usa capacitores como elementos de armazenamento sendo volátil.

Drenagem de corrente A ação de um circuito lógico na qual ele recebe corrente na saída a partir da carga.

Dreno Um dos terminais de um transistor de efeito de campo.

DSP Processador de sinais digitais; um tipo especial de microprocessador que processa dados em tempo real.

DTE Equipamento terminal de dados.

DVD-ROM *Digital versatile disk-ROM*; também conhecido como disco digital de vídeo em ROM; um tipo de dispositivo óptico de armazenamento no qual os dados previamente armazenados com uma capacidade muito maior que um CD-ROM.

E^2CMOS CMOS apagável eletricamente (EECMOS); tecnologia de CI usada em células reprogramáveis de PLDs.

ECL Lógica por acoplamento de emissor; uma classe de circuitos lógicos que são implementados com transistores bipolares de junção não saturados.

EDIF Formato de intercâmbio de um projeto eletrônico; uma forma padrão de uma netlist.

EEPROM Memória apenas de leitura programável e apagável eletricamente; um tipo de conexão programável não volátil de PLD baseada em células de memória apenas de leitura programável e apagável eletricamente podendo ser ligada ou desligada repetidas vezes por programação.

Elemento lógico A menor seção de um circuito lógico de um FPGA que contém tipicamente uma LUT, circuito lógico associado e um flip-flop.

Emissor Uma das três regiões em um transistor bipolar de junção.

Emparelhamento de instrução O processo de combinar certas instruções independentes de forma que elas possam ser executadas simultaneamente por duas unidades de execução separadas.

Endereço base O endereço inicial de um segmento de memória.

Endereço de offset A distância em número de bytes de um endereço físico a partir do endereço base.

Endereço físico A posição real de um dado na memória.

Endereço A posição de uma determinada célula de armazenamento ou de um grupo de células numa memória.

Entity A unidade VHDL que descreve as entradas e saídas de uma função lógica.

Entrada O sinal enviado para um circuito ou a linha que chega ao circuito. O sinal que controla a operação de um circuito.

Entrada/saída (I/O) Um terminal de um dispositivo que pode ser usado como entrada ou como saída.

EPROM Memória apenas de leitura programável e apagável; um tipo de conexão programável não volátil de PLD baseada em células de memória apenas de leitura programável eletricamente podendo ser ligada ou desligada uma vez por programação.

Escrita O processo de armazenamento de dados numa memória.

Estágio Um elemento (flip-flop) de armazenamento em um registrador.

EU Unidade de execução; a parte de uma CPU que executa instruções; ela contém a unidade lógica e aritmética (ALU), os registradores gerais e os flags.

Execução Processo de uma CPU no qual as instruções são executadas; um estágio da operação pipeline do DSP na qual as instruções decodificadas são executadas.

EX-OR Uma operação lógica básica na qual um nível lógico ALTO ocorre quando as duas entradas estão em níveis lógicos opostos.

Expoente A parte de um número de ponto flutuante que representa o número de casas que a vírgula decimal (ou vírgula binária) é movida.

Expressão Booleana Uma expressão de variáveis e operadores usada para expressar a operação de um circuito lógico.

Falante Um instrumento capaz de transmitir dados numa interface GPIB (barramento de interface de propósito geral).

Fan-out O número de entradas de portas equivalentes da mesma série de uma família que uma porta lógica é capaz de acionar.

Ferramenta ajustador Uma ferramenta de software compilador que seleciona as melhores interconexões, designações de pinos e designação de células lógicas para acomodar um projeto em um dispositivo destino selecionado.

FET Transistor de efeito de campo.

FIFO Memória na qual o primeiro dado a entrar é o primeiro a sair.

Fila Uma memória de alta velocidade que armazena instruções ou dados.

FireWire O padrão IEEE-1394 de barramento serial.

Flag Um bit que indica o resultado de uma operação lógica ou aritmética ou é usado para alterar uma operação.

Flip-flop D Um tipo de multivibrador biestável no qual a saída assume o estado da entrada *D* na borda de disparo do pulso de clock.

Flip-flop disparado por borda Um tipo de flip-flop no qual os dados são inseridos e aparecem na saída na mesma borda do clock.

Flip-flop J-K Um tipo de flip-flop que pode operar nos modos SET, RESET, repouso e *toggle* (comutação).

Flip-flop S-R Um flip-flop SET-RESET.

Flip-flop Um circuito de armazenamento básico que pode armazenar apenas um bit de cada vez; um dispositivo biestável síncrono.

Fluxo do projeto O processo ou seqüência de operações realizadas para programar um dispositivo destino.

Flying probe Um método para o teste automatizado de placas de circuito impresso, no qual uma ou várias pontas de prova se movem de um local para outro para fazer contato com pontos de teste.

Folhas de dados Um documento que especifica os parâmetros e as condições de operação para um circuito integrado ou outro dispositivo.

Fonte Um dispositivo que envia dados em um barramento; um dos terminais de efeito de campo.

Fornecimento de corrente A ação de um circuito lógico na qual ele envia corrente de sua saída para a carga.

FPGA de plataforma Um FPGA que contém um núcleo rígido e um núcleo flexível embutidos no processador entre outras funções.

FPGA Arranjo de portas programáveis por ação de campo; um dispositivo lógico programável que usa a LUT como elemento lógico básico e geralmente emprega a tecnologia de processo baseada em anti-fusível ou SRAM.

Freqüência (*f*) O número de pulsos em um segundo para uma forma de onda periódica. A unidade de freqüência é o hertz.

Freqüência de Nyquist O maior sinal de freqüência que pode ser amostrado para uma freqüência de amostragem especificada; uma freqüência igual ou menor que metade da freqüência de amostragem.

Fusível Um tipo de conexão programável não volátil de PLD que pode ser deixada intacta (curto-circuito) ou aberta conforme determinado pelo programa.

GAL Lógica de arranjo genérico; um tipo reprogramável de SPLD que é similar a um dispositivo PAL exceto que o primeiro usa uma tecnologia de processo reprogramável, tal como a EEPROM (E^2PROM), em vez de fusíveis.

Geração de carry O processo de produção de um carry de saída em um somador-completo quando os dois bits de entrada são 1s.

Glitch Um spike de tensão ou corrente de curta duração, geralmente produzido não – intencionalmente e indesejável.

GPIB Barramento de interface de propósito geral baseada no padrão IEEE-488.

Habilitação Para ativar ou colocar no modo de operação; uma entrada de um circuito lógico que habilita a operação dele.

Handshaking O processo de troca de sinais pelo qual dois dispositivos digitais ou sistemas estabelecem entre si uma comunicação.

Hardware O circuito e os componentes físicos de um sistema de computador (com sentido oposto ao que é denominado de software).

HDL Linguagem de descrição de hardware; uma linguagem usada para descrever um circuito lógico usando software.

Hexadecimal Descreve um sistema de numeração com base 16.

Histerese Uma característica de um circuito com limiar de comutação, tal como o Schmitt trigger, onde o dispositivo liga e desliga em diferentes nível de entrada.

Hi-Z O estado de alta impedância de um circuito tristate no qual a saída é eficientemente desconectada do restante do circuito.

HPIB Barramento de interface Hewlet Packard; o mesmo que GPIB (barramento de interface de propósito geral).

I^2L Lógica de injeção integrada; um tecnologia de CI.

IEEE 1394 Um barramento serial para transferência de dados em alta velocidade; também conhecido como FireWire.

IEEE Institute of Electrical AND Electronics Engineers.

Implementação O processo de software onde as estruturas lógicas descritas pela netlist são mapeadas na estrutura de um dispositivo destino.

Imunidade a ruído A capacidade de um circuito lógico de rejeitar sinais indesejados.

Incremento Para aumentar o estado binário de um contador de uma unidade.

Inserção gráfica (esquemático) Um método de inserir um projeto lógico no software criando graficamente um diagrama lógico (esquemático) em uma tela de projeto.

Inserção via esquemático Um método de colocar um projeto lógico em um software usando símbolo esquemáticos.

Inserção via texto Um método de colocar um projeto lógico em um software usando um linguagem de descrição de hardware (HDL).

Instrução Um passo em um programa de computador; uma unidade de informação que diz à CPU o que fazer.

Inteiro Um número inteiro.

Interconexão local Um conjunto de linhas que permitem a interconexão entre oito elementos lógicos em um bloco de arranjo lógico sem o uso da interconexão de linhas e colunas.

Interfaceamento O processo de fazer com que dois ou mais dispositivos eletrônicos ou sistemas compatíveis operacionalmente um com o outro funcionem corretamente juntos.

Interrupção por software Uma instrução que invoca uma rotina de serviço de interrupção.

Interrupção Um sinal de computador ou instrução que faz com que o processamento atual seja paralisado temporariamente enquanto uma rotina de serviço é executada.

Inversão A conversão de um nível ALTO em um nível BAIXO ou vice-versa; também denominado de complementação.

Inversor Um circuito NOT; um circuito que comuta um nível ALTO para um nível BAIXO ou vice-versa.

IP ponteiro de instrução; um registrador especial interno à CPU que mantém o endereço de offset da próxima instrução a ser executada.

ISP In-system programming; um método de programação de SPLDs após serem instalados em uma placa de circuito impresso e estarem operando em um sistema.

JTAG *Joint Test Action Group*; uma interface padrão projetada pela IEEE (padrão 1149.1)

Junção A fronteira entras as regiões *n* e *p* em um BJT.

LAB Bloco de arranjo lógico; um arranjo lógico SPLD em um CPLD.

Largura de pulso (t_w) O intervalo de tempo entre os pontos de 50% das bordas de subida e descida de um pulso; a duração do pulso.

Latch Um circuito digital biestável usado para armazenar um bit.

LCCC *Leadless ceramic chip carrier*; um encapsulamento SMT que tem contatos metálicos moldados no seu corpo.

LCD Display de cristal líquido.

LED Diodo emissor de luz.

Lei associativa Na adição (operação OR) e na multiplicação (operação AND) de três ou mais variáveis, a ordem na qual as variáveis são agrupadas não faz diferença.

Lei comutativa Na adição (operação OR) ou multiplicação (operação AND) de duas variáveis, a ordem na qual as variáveis são submetidas à operações OR ou AND não faz diferença.

Lei distributiva A lei que diz que a operação OR entre diversas variáveis seguida da operação AND do resultado com uma única variável é equivalente a uma operação AND entre a variável única com cada uma das diversas variáveis seguida da operação OR entre os produtos.

Leitura O processo de recuperação de dados armazenados numa memória.

LIFO Memória na qual o primeiro dado a entrar é o último a sair; uma memória pilha.

Linguagem assembly Uma linguagem de programação que usa palavras originárias do inglês e que tem uma correspondência individual com a linguagem de máquina.

Linguagem de alto nível Um tipo de linguagem de computador próxima à linguagem humana que está um nível acima da linguagem assembly.

Linguagem de máquina Instruções de computador escritas em código binário que são interpretados por um computador; o nível mais baixo de linguagem de programação.

Literal Uma variável ou o complemento dela.

Lógica combinacional Uma combinação de portas lógicas interconectadas para produzir uma função Booleana específica sem capacidade de armazenamento ou memória; algumas vezes denominada de lógica combinatória.

Lógica positiva O sistema de representação de um binário 1 com um nível ALTO e um binário 0 com um nível BAIXO.

Lógica Em eletrônica digital, a capacidade dos circuitos das portas de tomar decisões, para os quais um nível ALTO representa uma sentença verdadeira e um nível BAIXO uma sentença falsa.

LSI Integração em larga escala; um nível de complexidade de CIs de função fixa no qual existem entre 100 e 10.000 portas equivalentes por chip.

LUT Tabela de busca; um tipo de memória que pode ser programada para produzir funções de soma-de-produtos.

Macrocélula Um arranjo lógico de soma-de-produtos com saídas combinacionais ou registradas; parte de um dispositivo PAL, GAL ou CPLD que geralmente consiste de uma ou mais portas OR e alguma lógica de saída associada. A interconexão de múltiplas macrocélulas forma um CPLD.

Magnitude O valor e uma grandeza.

Mantissa A magnitude de um número em ponto flutuante.

Mapa de Karnaugh Um arranjo de células que representam as combinações de literais numa expressão Booleana e usado para uma simplificação sistemática dessa expressão.

Máquina de estados Um sistema lógico que exibe uma seqüência de estados condicionados pela lógica interna e as entradas externas; qualquer circuito seqüencial que exibe uma seqüência específica de estados.

Margem de ruído A diferença entre a saída máxima em nível BAIXO de uma porta e a entrada máxima em nível BAIXO aceitável de uma porta equivalente; é também a diferença entre a saída mínima em nível ALTO de uma porta e a entrada mínima em nível ALTO de uma porta equivalente. A margem de ruído é expressa algumas vezes com uma porcentagem da tensão de alimentação cc.

Meio-somador Um circuito digital que soma dois bits e produz uma soma e um carry de saída. Ele não tem a capacidade de operar com carry de entrada.

Memória cache Uma memória de alta velocidade relativamente pequena que armazena as instruções ou dados usados mais recentemente a partir da memória principal que é maior porém mais lenta.

Memória dinâmica Um tipo de memória semicondutora que tem células de armazenamento capacitivo que podem perder os dados armazenados ao longo de um período de tempo tendo, portanto, que serem reavivados (operação de refresh).

Memória estática Uma memória semicondutora volátil que usa flip-flops como células de armazenamento e é capaz de reter os dados sem a necessidade e refresh.

Memória flash Uma memória semicondutora de acesso aleatório, de leitura/escrita e não-volátil na qual os dados são armazenados como carga na porta flutuante num certo tipo de FET.

MFLOPS Milhões de operações de ponto flutuante por segundo.

Microprocessador Um circuito integrado digital de alta densidade (larga escala de integração) que pode ser programado com uma série de instruções para realizar uma função específica sobre os dados.

Minimização O processo que resulta numa expressão Booleana de soma-de-produtos ou num produto-de-somas que contém o menor número de literais por termo.

Minuendo O número a partir do qual um outro número é subtraído.

MIPS Milhões de instruções por segundo.

MMACS Milhões de multiplicações/acumulações por segundo.

Mnemônico Uma instrução originária do inglês que é convertida por um assemblador em código de máquina para ser usada pelo processador.

Modem Um modulador/demodulador para interface de dispositivos digitais com sistemas de transmissão analógicos tal como a linha telefônica.

Modo real Operação de um processador Intel em uma forma de emular a memória de 1 MB do 8086.

Modulação delta Um método de conversão analógico-digital usando um processo de quantização de 1 bit.

Módulo O número de estados únicos através dos quais o contador passa.

Monoestável Tem apenas um estado estável. Um multivibrador monoestável que produz um único pulso em resposta a uma entrada de disparo.

Monotonicidade A característica de um DAC definida pela ausência de qualquer degrau inverso incorreto; um tipo de linearidade digital-analógico.

MOS Semicondutor de óxido metálico; um tipo de tecnologia de transistor.

MOSFET Transistor de efeito de campo de semicondutor de óxido metálico.

MSI Integração em média escala; um nível de complexidade de CIs de função fixa no qual existem entre 10 e 100 portas equivalentes por chip.

Multiplexador (MUX) Um circuito que comuta dados digitais de diversas linhas de entrada para uma única linha de saída numa seqüência temporal especificada.

Multiplicação Booleana A operação AND na lógica Booleana.

Multiplicador O número que multiplica o multiplicando.

Multiplicando O número que é multiplicado pelo outro número.

Multivibrador Uma classe de circuitos digitais na qual a saída é conectada de volta para a entrada (um arranjo denominado de realimentação) para produzir dois estados estáveis, um estado estável ou nenhum estado estável, dependendo da configuração.

Não-volátil Um termo que descreve uma memória que pode reter dados armazenados quando a alimentação é removida.

Netlist Uma lista detalhada de informações necessárias para descrever um circuito, tal como os tipos de elementos, entradas e saídas e todas as interconexões.

Nibble Um grupo de quatro bits.

NMOS Semicondutor de óxido metálico de canal *n*.

Nó Um ponto de conexão comum num circuito no qual a saída de uma porta está conectada a uma ou mais entradas de portas.

NOT Uma operação lógica básica que realiza inversões.

Notação de dependência Um sistema notacional para símbolos lógicos que especificam relações entre entradas e saídas, assim definindo totalmente uma dada função; uma parte integral do padrão 91-1984 da ANSI/IEEE.

Núcleo de DSP A unidade de processamento central de um DSP.

Núcleo flexível Uma parte de um circuito lógico em um FPGA; similar ao núcleo rígido exceto que o primeiro apresenta algumas características programáveis.

Núcleo rígido Uma parte fixa de um circuito lógico em um FPGA que é inserida pelo fabricante para prover uma função específica.

Numérico Relacionado a números.

Número em ponto flutuante Uma representação numérica baseada em notação científica na qual o número consiste de um expoente e uma mantissa.

Octal Descreve um sistema de numeração com base oito.

OLMC Macrocélula lógica de saída; a parte de um dispositivo GAL que pode ser programada para uma saída combinacional ou registrada; um bloco em um circuito lógico de um dispositivo GAL que contém uma porta OR fixa e um outro circuito lógico para manuseio de entradas e/ou saídas.

OR negativa Uma operação de porta NAND equivalente na qual o nível ALTO é a entrada ativa quando uma ou mais entradas estão em nível BAIXO.

OR Uma operação lógica básica na qual uma saída verdadeira (nível ALTO) ocorre quando uma ou mais das condições de entrada são verdadeiras (nível ALTO).

Oscilador Um circuito eletrônico que é baseado nos princípios da realimentação regenerativa e produz uma forma de onda de saída repetitiva; uma fonte de sinal.

OTP *One-time programable* (programável apenas uma vez)

Ouvinte Um instrumento capaz de receber dados em um GPIB (barramento de interface de propósito geral).

Overflow A condição que ocorre quando o número de bits em uma soma excede ao número de bits em cada um dos números somados.

PAL Lógica de arranjo programável; um tipo de SPLD programável apenas uma vez que consiste de um arranjo programável de portas AND que se conecta a um arranjo fixo de portas OR.

Paralelo Em sistemas digitais, dados que ocorrem simultaneamente em diversas linhas; a transferência ou processamento de vários bits simultaneamente.

Paridade ímpar Condição apresentada por um grupo de bits que possui uma quantidade ímpar de 1s.

Paridade par A condição apresentada por um grupo de bits que tem uma quantidade par de 1s.

Paridade Em relação aos códigos binários, a condição de paridade par ou impar do número de 1s num grupo de código.

Periférico Um dispositivo ou instrumento que provê comunicação com um computador ou provê serviços auxiliares ou funções para o computador.

Periódico Descreve uma forma de onda que se repete por um intervalo de tempo fixo.

Período (*T*) O tempo necessário para uma forma de onda periódica se repetir.

Período de latência O tempo gasto para que um setor desejado gire sob a cabeça de leitura/gravação uma vez que essa esteja posicionada na trilha desejada de um disco rígido magnético.

Peso O valor de um dígito em um número baseado em sua posição no número.

PIC Controlador de interrupção programável; gerencia as interrupções baseadas em uma prioridade.

Pipeline Quando se refere a memórias, é uma implementação que permite que uma operação de leitura ou escrita seja iniciada antes que a operação anterior seja completada; parte da arquitetura de um DSP a qual permite que múltiplas instruções sejam processadas simultaneamente.

PLA arranjo lógico programável; um SPLD com arranjos AND e OR programáveis.

PLCC *Plastic leaded chip carrier*; um encapsulamento SMT cujos terminais são dobrados para baixo do CI na forma de um J.

PLD Dispositivo lógico programável; um circuito integrado que pode ser programado com qualquer função lógica específica.

PMOS Um semicondutor de óxido metálico de canal *p*.

Polarização direta Uma condição de polaridade de tensão que permite a uma junção semicondutora *pn* em um transistor ou diodo conduzir corrente.

Polarização reversa Uma condição de polaridade de tensão que evita que uma junção *pn* de um transistor ou diodo conduza corrente.

Ponta de prova Um acessório usado para conectar uma tensão na entrada de um osciloscópio ou outro instrumento.

Ponteiro O conteúdo de um registrador (ou registradores) que contêm um endereço.

Porta AND Uma porta lógica que produz uma saída de nível ALTO apenas quando todas as entrada estiverem em nível ALTO.

Porta de I/O Uma interface física em um computador através da qual os dados são enviados ou recebidos pelos periféricos.

Porta de I/O Uma interface física em um computador através da qual os dados são transferidos ou recebidos de periféricos.

Porta EX-NOR Uma porta lógica que produz uma saída de nível BAIXO apenas quando suas duas entradas estiverem em níveis opostos.

Porta EX-OR Uma porta lógica que produz uma saída de nível ALTO apenas quando suas duas entradas estiverem em níveis opostos.

Porta NAND Uma porta lógica que produz uma saída de nível BAIXO apenas quando todas as entradas estão em nível ALTO.

Porta NOR Uma porta lógica na qual a saída é nível BAIXO quando uma ou mais entradas estiverem em nível ALTO.

Porta OR Uma porta lógica que produz uma saída de nível ALTO quando uma ou mais entradas estiverem em nível ALTO.

Porta universal Tanto a porta NAND quanto a porta NOR são portas universais. O termo universal se refere à propriedade de uma porta permitir que qualquer função lógica seja implementada com essa porta ou uma combinação desse tipo de porta.

Porta Um circuito lógico que realiza uma operação lógica especificada, tal como AND ou OR; um dos três terminais de um transistor de efeito de campo.

Pré-busca O processo de execução de instruções ao mesmo tempo em que outras instruções são "buscadas" eliminando o tempo de inatividade: também chamado de *pipelining*.

Preset Uma entrada assíncrona usada para setar um flip-flop (fazer a saída *Q* igual a 1).

1ª parcela Na adição, é o número que é somado com um outro número denominado de 2ª parcela.

Primitivo Um elemento lógico básico tal como uma porta lógica ou um flip-flop, pinos de entrada/saída, GND e V_{CC}.

Produto velocidade-potência Um parâmetro de desempenho que é o produto do tempo de atraso de propagação pela dissipação de potência em um circuito de entrada.

Produto O resultado de uma multiplicação.

Produto-de-somas Uma forma de expressão Booleana que é basicamente a operação AND de termo OR.

Programa fonte Um programa escrito em assembly ou linguagem de alto nível.

Programa objeto A tradução para linguagem de máquina de um programa fonte em linguagem de alto nível.

Programa Uma lista de instruções ordenadas que um computador segue para obter um resultado específico; software.

PROM Memória semicondutora apenas de leitura programável; um SPLD com um arranjo fixo programável e um arranjo OR programável; usada como um dispositivo de memória e normalmente não como um dispositivo lógico.

Propagação do carry O processo de ondulação de um carry de entrada para se tornar um carry de saída em um somador-completo quando qualquer um dos bits de entrada for 1, ou então ambos, e o carry de entrada for 1.

Propriedade intelectual (IP) Projetos próprios de um fabricante de dispositivos lógicos programáveis.

Pseudo-operação Uma instrução para o assemblador (em oposição ao processador).

Pulso Uma mudança súbita do sinal de um nível para outro e, após um tempo (denominado de largura de pulso), retorna subitamente para o nível original.

QIC *Quarter-inch cassette*; um tipo de fita magnética.

Quantização O processo por meio do qual um código binário é associado a cada valor amostrado durante a conversão analógico digital.

Quociente O resultado de uma divisão.

RAM Memória de acesso aleatório; uma memória semicondutora de leitura/escrita volátil.

Rastreamento de sinal Uma técnica de análise de defeito na qual formas de onda são observadas de forma passo-a-passo começando pela entrada do circuito em direção à saída ou vice versa. Em cada ponto a forma de onda observada é comparada com o sinal correto para aquele ponto.

Realimentação A tensão de saída, ou a parte dela, que é conectada de volta à entrada do circuito.

Reciclagem Sofrer transição (como em um contador) de um estado final de volta para o estado inicial.

Refresh Para renovar o conteúdo de uma memória dinâmica recarregando as células de armazenamento capacitivas.

Registrado A saída de uma macrocélula de um CPLD na qual a saída provém de um flip-flop.

Registrador de deslocamento universal Um registrador que tem capacidade de entrada e saída serial e paralela.

Registrador Um circuito digital capaz de armazenar e deslocar informação binária; tipicamente usado como um dispositivo de armazenamento temporário.

RESET O estado de um flip-flop ou latch quando a saída é 0; a ação que produz o estado de RESET.

Resistor de pull-up Um resistor com uma das extremidades conectada à fonte de alimentação cc usado para manter em nível ALTO um determinado ponto de um circuito lógico quando no estado inativo.

Resolução O número de bits usados em um ADC.

Resto O valor que sobra após uma divisão.

ROM Memória semicondutora apenas de leitura, acessada aleatoriamente; também referida como ROM de máscara.

Saída O sinal ou a linha que sai de um circuito.

Schottky Um tipo específico de tecnologia de circuito com lógica transistor-transistor.

SCSI Interface para pequenos sistemas de computadores; um padrão de barramento paralelo externo.

SDRAM Memória de acesso aleatório dinâmica síncrona.

Segmento Um bloco de 64 k de memória.

2ª parcela Na adição, o numero com o qual a 1ª parcela é somado.

Seletor de dados Um circuito que seleciona dados a partir de varias entradas uma de cada vez numa seqüência e os coloca na saída; também denominado de multiplexador.

Seqüência de bits Uma série de bits que descrevem um projeto final que é enviado a um dispositivo destino durante a programação.

Serial Que tem um elemento seguido de outro, como em uma transferência serial de bits; ocorre em seqüência em vez de simultaneamente.

SET O estado de um flip-flop ou latch quando a saída é 1; a ação que produz o estado de SET.

SIMM Módulo de memória com pinos em uma linha.

Simulação de temporização Um processo via software que usa informação dos atrasos de propagação e os dados da netlist para testar a operação lógica e a temporização para o pior caso de um projeto.

Simulação funcional Um processo feito por software que testa a operação funcional ou lógica de um projeto.

Sinal Um tipo de objeto VHDL que mantém dados.

Síncrono Tem uma relação de tempo fixo; que ocorre no mesmo instante.

Síntese O processo de software onde o projeto é traduzido em uma netlist.

SMT Tecnologia de montagem em superfície; uma técnica de encapsulamento de CI na qual os encapsulamentos são menores que os DIPs e são montados na superfície da placa de circuito impresso.

Software Programas de computador; programas que instruem um computador o que fazer para executar um determinado conjunto de tarefas.

SOIC *Small-outline integrated circuit*; um encapsulamento SMT que se assemelha a um pequeno DIP porem tendo terminais dobrados na foram de "asa de gaivota".

Soma O resultado quando dois ou mais números são somados juntos.

Soma-de-produtos Uma forma de expressão Booleana que é basicamente uma operação OR de termos AND.

Somador Um circuito lógico usado para somar dois números binários.

Somador-completo Um circuito digital que soma dois bits e um carry de entrada para produzir uma soma e um carry de saída.

SPLD Dispositivo lógico programável simples. Um arranjo de porta AND e OR que podem ser programadas para se obter funções lógicas programáveis. Os quatro tipos são PROM, PLA, PAL e GAL.

SRAM Memória de acesso aleatório estática; um tipo de conexão programável volátil de PLD baseada em células de memória de acesso aleatório estática e que pode ser ligada ou desligada repetidas vezes por programação.

SSI Integração em pequena escala; um nível de complexidade de CIs de função fixa na qual existem até 10 portas equivalentes por chip.

SSOP Shrink small-outline package.

String Uma seqüência contígua de bytes ou palavras.

Strobing Um processo do uso de um pulso para amostrar a ocorrência de um evento num instante especificado em relação ao evento.

Sub-rotina Uma série de instruções que podem ser montadas juntas e usadas repetidamente por um programa porém programada apenas uma vez.

Subtraendo O número que é subtraído do minuendo.

Subtrator Um circuito lógico usado para subtrair dois números binários.

Supressão de zero O processo de apagamento dos zeros mais significativos anteriores à primeira casa antes da vírgula e os menos significativos posteriores à vírgula.

Tabela-verdade Uma tabela que mostra as entradas e a(s) correspondente(s) saída(s) de um circuito lógico.

Tamanho da palavra O número de bits em uma palavra.

Tempo de acesso O tempo a partir da aplicação de um endereço de memória válido até o surgimento de dados válidos na saída.

Tempo de atraso de propagação O intervalo de tempo entre a ocorrência de uma transição de entrada e a correspondente transição de saída num circuito lógico.

Tempo de bit O intervalo de tempo ocupado por um único bit em uma seqüência de bits; o período do clock.

Tempo de busca O tempo para que a cabeça de leitura/gravação em um drive de disco rígido se posicione na trilha desejada para uma operação de leitura.

Tempo de descida O intervalo de tempo entre os pontos de 90% e 10% da borda negativa de um pulso.

Tempo de hold O intervalo de tempo necessário para os níveis de controle permanecer nas entradas de um flip-flop após a borda de disparo do clock para a ativação confiável do dispositivo.

Tempo de setup O intervalo de tempo necessário para os níveis de controle serem colocados nas entradas de um circuito digital, tal como um flip-flop, antes da borda de disparo de um pulso de clock.

Tempo de subida O tempo necessário para a borda positiva de um pulso passe de 10 para 90% do valor máximo.

Temporizador Um circuito que pode ser usado como um monoestável ou um oscilador; um circuito que produz uma saída com intervalo de tempo fixo.

Termo-produto O produto Booleano de dois ou mais literais equivalente a uma operação AND.

Termo-soma A soma Booleana de dois ou mais literais equivalentes a uma operação OR.

Toggle A ação de um flip-flop quando comuta de estado a cada pulso de clock.

Totem-pole Um tipo de saída em circuitos TTL.

Transistor Um dispositivo semicondutor que apresenta ganho de tensão e/ou corrente. Quando usado como dispositivo de chaveamento, se aproxima de uma chave aberta ou fechada.

Tristate Um tipo de saída em circuitos lógicos que apresenta três estados: ALTO, BAIXO e alta impedância; também conhecido como 3 estados.

TSSOP Thin shrink small-outline package.

TTL Lógica transistor-transistor; uma classe de circuitos lógicos integrados que usa transistores de junção bipolar.

TVSOP Thin very small-outline package.

ULSI Integração em escala ultra ampla; um nível de complexidade de um CI no qual existem mais de 100.000 portas equivalentes por chip.

Unidade de controle A parte interna a um microprocessador que prover os sinais de temporização e controle para a transferência bidirecional de dados e para a sincronização da execução de instruções.

USB Barramento serial universal; um padrão de barramento serial externo.

UV EPROM ROM programável e apagável por ultravioleta.

Variável Um símbolo usado para representar uma grandeza lógica que pode ter um valor 1 ou 0, geralmente designada por uma letra em itálico.

VHDL Uma linguagem de descrição de hardware padrão. Padrão 1076-1993 da IEEE.

VLSI Integração em escala muito ampla; um nível de complexidade de um CI no qual existem entre 10.000 e 100.000 portas equivalentes por chip.

Volátil A característica de um dispositivos lógico programável que perde os dados programados quando a alimentação é desligada.

Word Uma unidade completa de dado binário (palavra).

WORM *Write once-read many*; um tipo de dispositivo de armazenamento óptico.

Respostas para os Problemas de Número Ímpar

Capítulo 1

1. Os dados digitais podem ser transmitidos e armazenados de forma mais eficiente e confiável.

3. **(a)** 11010001 **(b)** 000101010

5. **(a)** 550 ns **(b)** 600 ns
(c) 2,7 μs **(d)** 10 V

7. 250 Hz

9. 50%

11. 8 μs; 1 μs

13. Porta AND

15. **(a)** Somador **(b)** multiplicador
(c) multiplexador **(d)** comparador

17. 01010000

19. No encapsulamento DIP os pinos passam através de furos na placa de circuito. No encapsulamento SMT os pinos são conectados (soldados) na superfície das ilhas.

21. ABEL e CUPL

23. **(a)** Inserção do projeto: O passo no fluxo do projeto de uma lógica programável onde a descrição do circuito é inserida na forma de esquema (forma gráfica) ou na forma de texto usando HDL.
(b) Simulação: O passo no fluxo de um projeto onde o projeto inserido é simulado baseado nas formas de onda de entrada definidas.
(c) Compilação: Um processo de um programa que controla o processo do fluxo do projeto e traduz o código fonte do projeto em código objeto para teste e download.
(d) Download: O processo no qual o projeto é transferido do software para o hardware.

25. 7 V

27. Sistema é um conjunto de circuitos interconectados para realizar uma função específica.

29. Inserindo um novo valor no teclado.

Capítulo 2

1. **(a)** 1 **(b)** 100 **(c)** 100,000

3. **(a)** 400; 70; 1 **(b)** 9000; 300; 50; 6
(c) 100.000; 20.000; 5000; 0; 0; 0

5. **(a)** 3 **(b)** 4 **(c)** 7 **(d)** 8 **(e)** 9
(f) 12 **(g)** 11 **(h)** 15

7. **(a)** 51,75 **(b)** 42,25 **(c)** 65,875
(d) 120,625 **(e)** 92,65625 **(f)** 113,0625
(g) 90,625 **(h)** 127,96875

9. **(a)** 5 bits **(b)** 6 bits **(c)** 6 bits
(d) 7 bits **(e)** 7 bits **(f)** 7 bits
(g) 8 bits **(h)** 8 bits

11. **(a)** 1010 **(b)** 10001 **(c)** 11000
(d) 110000 **(e)** 111101 **(f)** 1011101
(g) 1111101 **(h)** 10111010

13. **(a)** 1111 **(b)** 10101 **(c)** 11100
(d) 100010 **(e)** 101000 **(f)** 111011
(g) 1000001 **(h)** 1001001

15. **(a)** 100 **(b)** 100 **(c)** 1000
(d) 1101 **(e)** 1110 **(f)** 11000

17. **(a)** 1001 **(b)** 1000 **(c)** 100011
(d) 110110 **(e)** 10101001 **(f)** 10110110

19. **(a)** 010 **(b)** 001 **(c)** 0101
(d) 00101000 **(e)** 0001010 **(f)** 11110

21. **(a)** 00011101 **(b)** 11010101
(c) 01100100 **(d)** 11111011

23. **(a)** 00001100 **(b)** 10111100
(c) 01100101 **(d)** 10000011

25. **(a)** −102 **(b)** +116 **(c)** −64

27. **(a)** 0 10001101 11110000101011000000000
(b) 1 10001010 11000001100000000000000

29. **(a)** 00110000 **(b)** 00011101
(c) 11101011 **(d)** 100111110

31. **(a)** 11000101 **(b)** 11000000

33. 100111001010

35. **(a)** 00111000
(b) 01011001
(c) 101000010100
(d) 010111001000
(e) 0100000100000000
(f) 1111101100010111
(g) 1000101010011101

37. **(a)** 35 **(b)** 146 **(c)** 26 **(d)** 141
(e) 243 **(f)** 235 **(g)** 1474 **(h)** 1792

39. **(a)** 60_{16} **(b)** $10B_{16}$ **(c)** $1BA_{16}$

41. **(a)** 10 **(b)** 23 **(c)** 46 **(d)** 52 **(e)** 67
(f) 367 **(g)** 115 **(h)** 532 **(i)** 4085

43. **(a)** 001011 **(b)** 101111
(c) 001000001
(d) 011010001
(e) 101100000
(f) 100110101011
(g) 001011010111001
(h) 100101110000000
(i) 001000000010001011

45. **(a)** 00010000 **(b)** 00010011
(c) 00011000 **(d)** 00100001
(e) 00100101 **(f)** 00110110
(g) 01000100 **(h)** 01010111
(i) 01101001 **(j)** 10011000
(k) 000100100101 **(l)** 000101010110

47. **(a)** 000100000100 **(b)** 000100101000
(c) 000100110010 **(d)** 000101010000
(e) 000110000110 **(f)** 001000010000
(g) 001101011001 **(h)** 010101000111
(i) 0001000001010001

49. **(a)** 80 **(b)** 237 **(c)** 346 **(d)** 421
(e) 754 **(f)** 800 **(g)** 978 **(h)** 1683
(i) 9018 **(j)** 6667

51. **(a)** 00010100 **(b)** 00010010
(c) 00010111 **(d)** 00010110
(e) 01010010 **(f)** 000100001001
(g) 000110010101 **(h)** 0001001001101001

53. No código Gray apenas um bit muda a cada vez que um número passa para o próximo da seqüência.

55. **(a)** 1100 **(b)** 00011 **(c)** 10000011110

57. **(a)** CAN **(b)** J **(c)** =
(d) # **(e)** > **(f)** B

59. 48 65 6C 6C 6F 2E 20 48 6F 77 20 61 72 65 20 79 6F 75 3F

61. **(b)** Está incorreto.

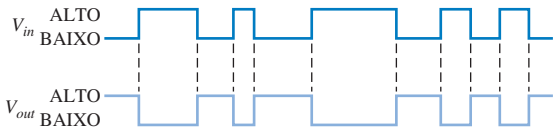
63. **(a)** 110100100 **(b)** 000001001
(c) 111111110

65. 001010001

67. **(a)** 110100010 **(b)** 100000101

CAPÍTULO 3

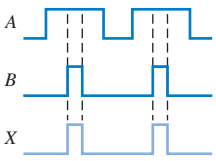
1. Veja a Figura P–1.



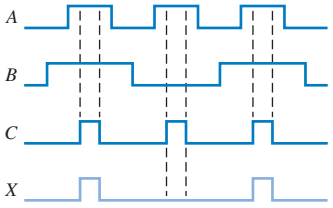
▲ **FIGURA P–1**

3. Veja a Figura P–2.

▶ **FIGURA P–2**

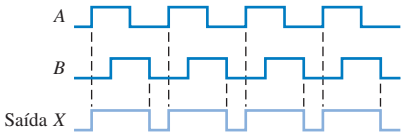


5. Veja a Figura P–3.



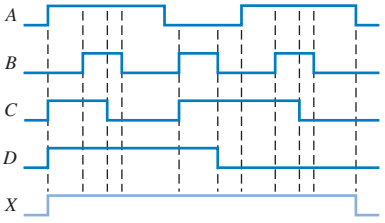
▲ **FIGURA P–3**

7. Veja a Figura P–4.



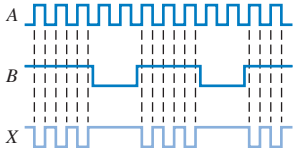
▲ **FIGURA P–4**

9. Veja a Figura P–5.



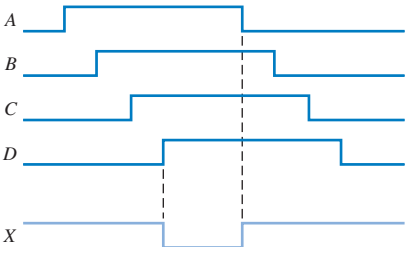
▲ **FIGURA P–5**

11. Veja a Figura P–6.



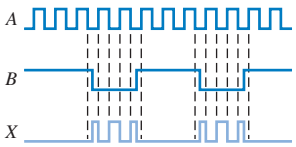
▲ **FIGURA P–6**

13. Veja a Figura P–7.



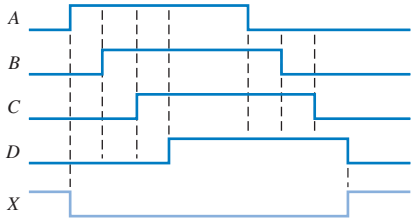
▲ **FIGURA P–7**

15. Veja a Figura P–8.



▲ **FIGURA P–8**

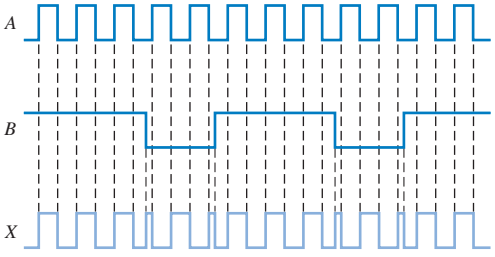
17. Veja a Figura P–9.



▲ **FIGURA P–9**

19. EX-OR $= A\overline{B} + \overline{A}B$; OR $= A + B$

21. Veja a Figura P–10.



▲ **FIGURA P–10**

23. $X_1 = \overline{A}B, X_2 = \overline{A}\,\overline{B}, X_3 = A\overline{B}$

25. CMOS

27. $t_{PLH} = 4{,}3$ ns; $t_{PHL} = 10{,}5$ ns

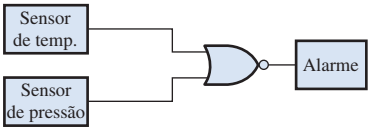
29. 20 mW

31. As portas nas partes (b), (c) e (e) da figura estão com defeito.

33. **(a)** Saída com defeito (travada em nível BAIXO ou aberta)
(b) A entrada no pino 4 ou a saída no pino 6 aberta internamente.

35. O sinal denominado Sinto de Segurança na entrada da porta AND está aberto.

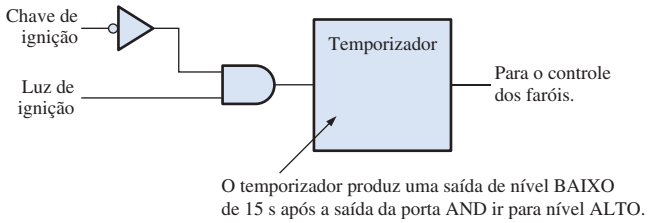
37. Veja a Figura P–11.



▲ **FIGURA P–11**

39. Deve-se acrescentar um inversor à entrada de habilitação da porta AND.

41. Veja a Figura P–12.



▲ **FIGURA P–12**

43. As entradas agora são ativas em nível BAIXO. Troque as portas OR por portas NAND (OR negativa) e acrescente dois inversores.

45. As entradas da porta AND estão em curto.

47. A saída da porta está aberta.

CAPÍTULO 4

1. $X = A + B + C + D$

3. $X = \overline{A} + \overline{B} + \overline{C}$

5. **(a)** $AB = 1$ quando $A = 1, B = 1$
(b) $A\overline{B}C = 1$ quando $A = 1, B = 0, C = 1$
(c) $A + B = 0$ quando $A = 0, B = 0$
(d) $\overline{A} + B + \overline{C} = 0$ quando $A = 1, B = 0, C = 1$
(e) $\overline{A} + \overline{B} + C = 0$ quando $A = 1, B = 1, C = 0$
(f) $\overline{A} + B = 0$ quando $A = 1, B = 0$
(g) $A\overline{B}\,\overline{C} = 1$ quando $A = 1, B = 0, C = 0$

7. **(a)** Comutativa **(b)** Comutativa
(c) Distributiva

9. **(a)** $\overline{A}B$ **(b)** $A + \overline{B}$
(c) $\overline{A}\,\overline{B}\,\overline{C}$ **(d)** $\overline{A} + \overline{B} + \overline{C}$
(e) $\overline{A} + \overline{B}\,\overline{C}$ **(f)** $\overline{A} + \overline{B} + \overline{C} + \overline{D}$
(g) $(\overline{A} + \overline{B})(\overline{C} + \overline{D})$ **(h)** $\overline{A}B + C\overline{D}$

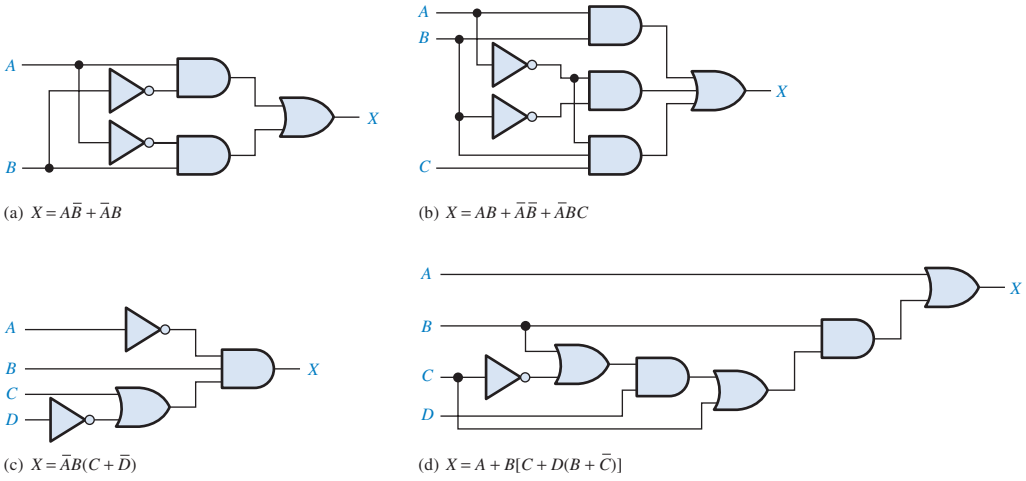
11. **(a)** $(\overline{A} + \overline{B} + \overline{C})(\overline{E} + \overline{F} + \overline{G})(\overline{H} + \overline{I} + \overline{J})(\overline{K} + \overline{L} + \overline{M})$
(b) $\overline{A}B\overline{C} + BC$
(c) $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}$

13. **(a)** $X = ABCD$ **(b)** $X = AB + C$
(c) $X = \overline{\overline{A}B}$ **(d)** $X = (A + B)C$

15. Veja a Figura P–13.

17. **(a)** A **(b)** AB **(c)** C
(d) A **(e)** $\overline{A}C + \overline{B}C$

19. **(a)** $BD + BE + \overline{D}F$ **(b)** $\overline{A}\,\overline{B}C + \overline{A}\,\overline{B}D$
(c) B **(d)** $AB + CD$
(e) ABC



▲ **FIGURA P–13**

21. (a) $A\overline{B} + AC + BC$ **(b)** $AC + \overline{B}C$ **(c)** $AB + AC$

23. (a) Domínio: A, B, C
Soma-de-produtos padrão: $A\overline{B}C + A\overline{B}\,\overline{C} + ABC + \overline{A}BC$

(b) Domínio: A, B, C
Soma-de-produtos padrão: $ABC + A\overline{B}C + \overline{A}\,\overline{B}C$

(c) Domínio: A, B, C
Soma-de-produtos padrão: $ABC + AB\overline{C} + A\overline{B}C$

25. (a) 101 + 100 + 111 + 011 **(b)** 111 + 101 + 001
(c) 111 + 110 + 101

27. (a) $(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)$
$(\overline{A} + \overline{B} + C)$

(b) $(A + B + C)(A + \overline{B} + C)(A + \overline{B} + \overline{C})$
$(\overline{A} + B + C)(\overline{A} + \overline{B} + C)$

(c) $(A + B + C)(A + B + \overline{C})(A + \overline{B} + C)$
$(A + \overline{B} + \overline{C})(\overline{A} + B + C)$

29. (a) Veja a Tabela P–1.

▼ **TABELA P–1**

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

29. (b) Veja a Tabela P–2.

▼ **TABELA P–2**

X	Y	Z	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

31. (a) Veja a Tabela P–3.

▼ **TABELA P–3**

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

31. (b) Veja a Tabela P–4.

▼ **TABELA P–4**

W	X	Y	Z	Q
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

33. (a) Veja a Tabela P–5.

▼ **TABELA P–5**

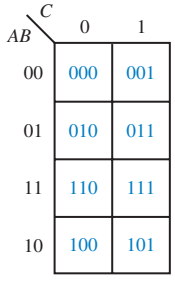
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

33. (b) Veja a Tabela P–6.

▼ **TABELA P–6**

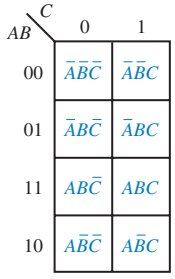
A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

35. Veja a Figura P–14.



▶ **FIGURA P–14**

37. Veja a Figura P–15.



▶ **FIGURA P–15**

39. **(a)** Sem simplificação **(b)** AC
(c) $\overline{D}\,\overline{F} + E\overline{F}$

41. **(a)** $AB + AC$
(b) $A + BC$
(c) $B\overline{C}D + A\overline{C}D + BC\overline{D} + AC\overline{D}$
(d) $A\overline{B} + CD$

43. $\overline{B} + C$

45. $\overline{A}\,\overline{B}\,\overline{C}D + C\overline{D} + BC + A\overline{D}$

47. **(a)** $(A + \overline{B} + C + \overline{D})(\overline{A} + B + \overline{C} + D)$
$(\overline{A} + \overline{B} + \overline{C} + \overline{D})$
(b) $(W + \overline{Z})(W + X)(\overline{Y} + \overline{Z})(X + \overline{Y})$

49. $(A + C + D)(A + \overline{B} + C)(\overline{A} + B + \overline{D})$
$(B + \overline{C} + \overline{D})(\overline{A} + \overline{B} + \overline{C} + D)$

51. $X = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E} + \overline{A}BC\overline{D}E + AB\overline{C}\,\overline{D}\,\overline{E} +$
$\overline{A}\,\overline{B}DE + \overline{A}BD\overline{E} + \overline{B}\,\overline{C}DE + AB\overline{C}D$

53.
```
entity AND_OR is
    port (A, B, C, D, E, F, G, H, I: in bit; X: out bit);
end entity AND_OR;
architecture Logic of AND_OR is
begin
    X <= (A and B and C) or (D and E and F) or
     (G and H and I);
end architecture Logic;
```

55. Um display de LEDs. Porque os LEDs emitem luz e os LCDs não.

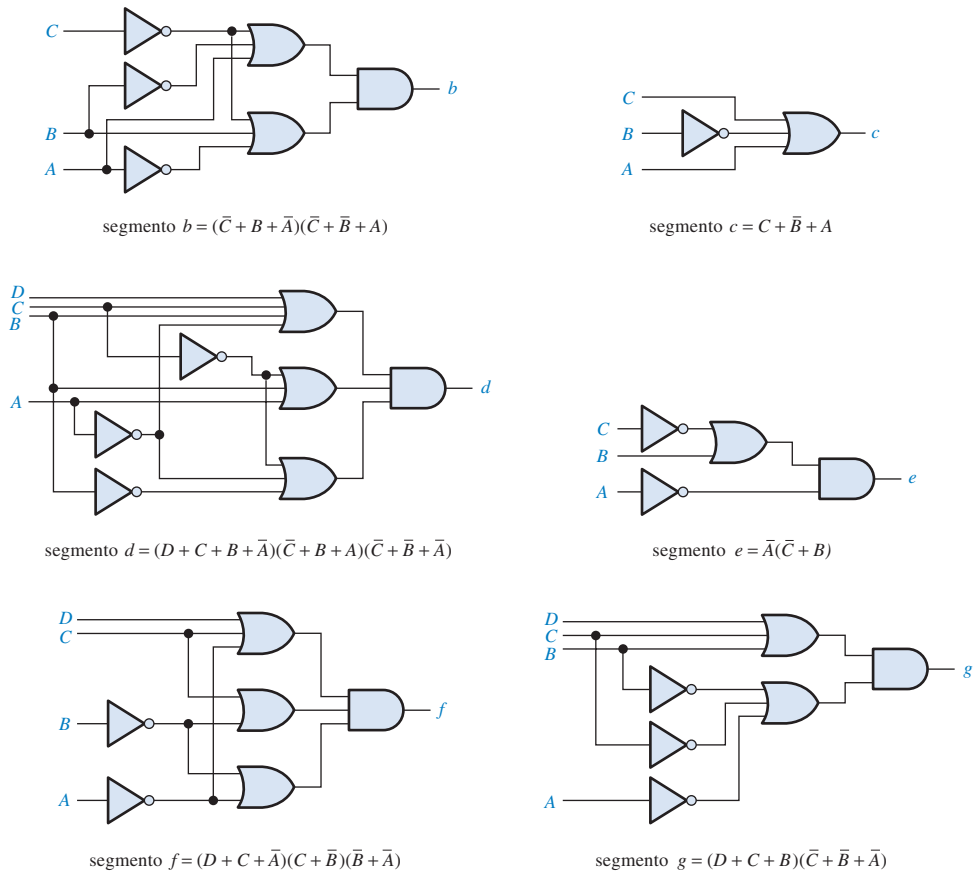
57. Um inversor a menos e seis portas a menos.

59. Acrescente um inversor na saída da porta OR em cada um dos segmentos.

61. Veja a Figura P–16.

63. Saída do inversor aberta.

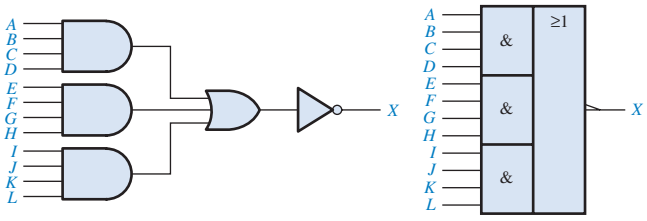
65. Saída da porta OR do segmento *b* aberta.



▶ **FIGURA P–16**

CAPÍTULO 5

1. Veja a Figura P–17.



▲ FIGURA P–17

3. **(a)** $X = ABB$ **(b)** $X = AB + B$
(c) $X = \overline{A} + B$ **(d)** $X = (A + B) + AB$
(e) $X = \overline{\overline{A}BC}$ **(f)** $X = (A + B)(\overline{B} + C)$

5. (a)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

5. (b)

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

5. (c)

A	B	X
0	0	1
0	1	1
1	0	0
1	1	1

5. (d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

5. (e)

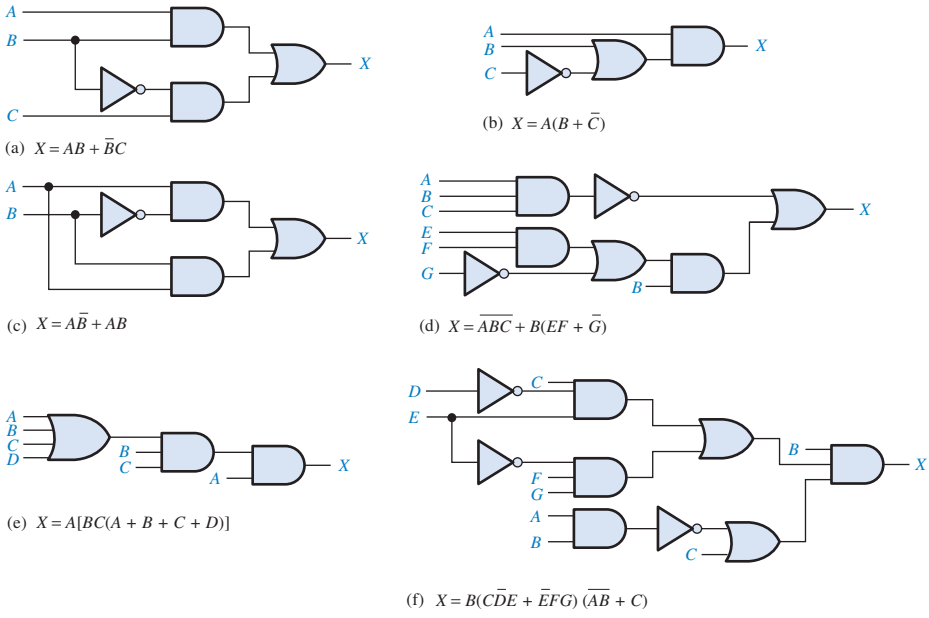
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

5. (f)

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

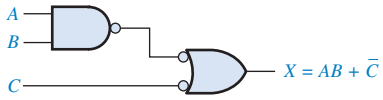
7. $X = \overline{A\overline{B} + \overline{A}B} = (\overline{A} + B)(A + \overline{B})$

9. Veja a Figura P–18.



▲ FIGURA P–18

11. Veja a Figura P–19.



▲ **FIGURA P–19**

13. $X = AB$

15. **(a)** Sem simplificação

(b) Sem simplificação

(c) $X = A$

(d) $X = \overline{A} + \overline{B} + \overline{C} + EF + \overline{G}$

(e) $X = ABC$

(f) $X = BC\overline{D}E + \overline{A}B\overline{E}FG + BC\overline{E}FG$

17. **(a)** $X = AC + AD + BC + BD$

(b) $X = \overline{A}CD + \overline{B}CD$

(c) $X = ABD + CD + E$

(d) $X = \overline{A} + B + D$

(e) $X = ABD + \overline{C}D + \overline{E}$

(f) $X = \overline{A}\,\overline{C} + \overline{A}\,\overline{D} + \overline{B}\,\overline{C} + \overline{B}\,\overline{D} + \overline{E}\,\overline{G} + \overline{E}\,\overline{H} + \overline{F}\,\overline{G} + \overline{F}\,\overline{H}$

19. Veja a Figura P–20.

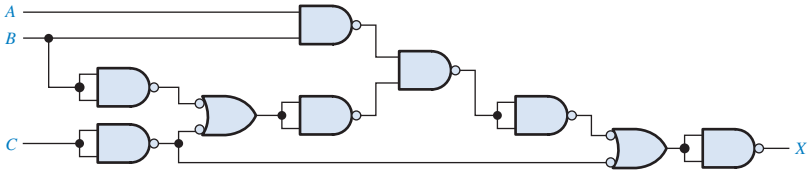
21. Veja a Figura P–21.

23. Veja a Figura P–22.

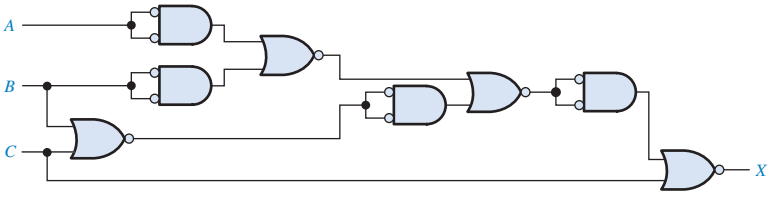
25. Veja a Figura P–23 na página 852.

27. $X = A + \overline{B}$; Veja a Figura P–24.

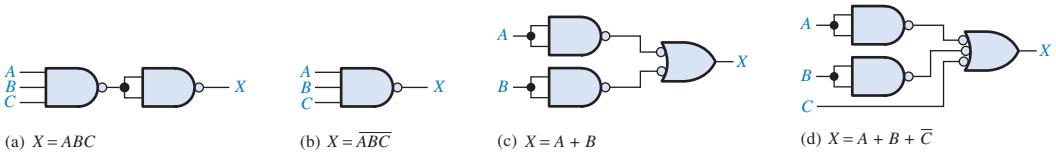
29. $X = A\overline{B}\,\overline{C}$; Veja a Figura P–25.

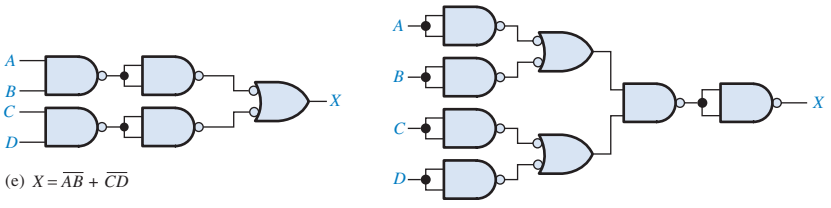


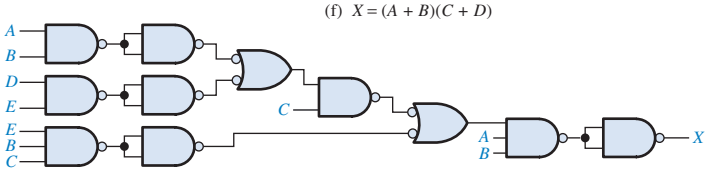
▶ **FIGURA P–20**



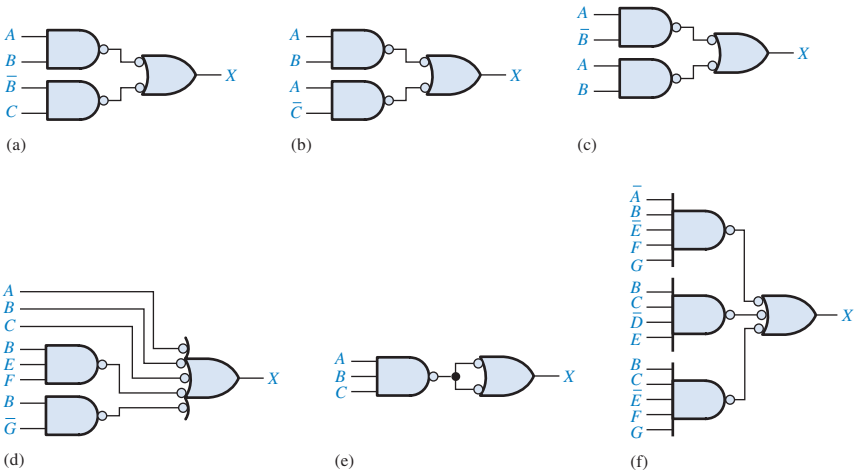
▶ **FIGURA P–21**



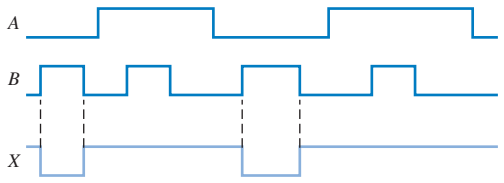




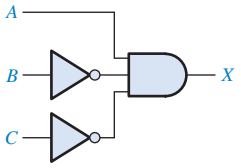
▶ **FIGURA P–22**



▶ **FIGURA P–23**



▲ **FIGURA P–24**



▲ **FIGURA P–25**

31. A largura do pulso de saída é maior que o mínimo especificado.

33. (e)

```
entity Circuit 5_52e is
    port (A, B, C: in bit; X: out bit);
end entity Circuit5_52e;
architecture LogicFunction of Circuit5_52e is
begin
    X <= (not A and B) or B or (B and not C) or
         (not A and not C) or (B and not C) or not C;
end architecture LogicFunction;
```

(f)

```
entity Circuit5_52f is
    port (A, B, C: in bit; X: out bit);
end entity Circuit5_52f;
architecture LogicFunction of Circuit5_52f is
begin
    X <= (A or B) and (not B or C);
end architecture LogicFunction;
```

35. As portas são numeradas de cima para baixo e da esquerda para a direita: G1, G2, G3, etc. A entradas são identificadas como IN1, IN2, IN3, etc. e a saída como OUT.

```
entity Circuit5_53f is
    port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8: in bit;
    OUT: out bit);
end entity Circuit5_53f;
architecture LogicFunction of Circuit5_53f is
component NAND_gate is
    port (A, B: in bit; X: out bit);
end component NAND_gate;
    signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT,
    G6OUT: bit;
begin
    G1: NAND_gate port map (A => IN1, B => IN2, X =>
    G1OUT);
    G2: NAND_gate port map (A => IN3, B => IN4, X =>
    G2OUT);
    G3: NAND_gate port map (A => IN5, B => IN6, X =>
    G3OUT);
    G4: NAND_gate port map (A => IN7, B => IN8, X =>
    G4OUT);
    G5: NAND_gate port map (A => G1OUT, B =>
    G2OUT, X => G5OUT);
    G6: NAND_gate port map (A => G3OUT, B =>
    G4OUT, X => G6OUT);
    G7: NAND_gate port map (A => G5OUT, B =>
    G6OUT, X => OUT);
end architecture LogicFunction;
```

37. -- Abordagem de fluxo de dados

```
entity Fig5_64 is
    port (A, B, C, D, E: in bit; X: out bit);
end entity Fig5_64;
architecture DataFlow of Fig5_64 is
begin
    X <= (A and B and C) or (D and not E);
end architecture DataFlow;
--Abordagem estrutural
entity Fig5_64 is
    port (IN1, IN2, IN3, IN4, IN5: in bit; OUT: out bit);
end entity Fig5_64;
architecture Structure of Fig5_64 is
component AND_gate is
    port (A, B: in bit; X: out bit);
end component AND_gate;
component OR_gate is
    port (A, B: in bit; X: out bit);
end component OR_gate;
component Inverter is
    port (A: in bit; X: out bit);
end component Inverter;
    signal G1OUT, G2OUT, G3OUT, INVOUT: bit;
begin
    G1: AND_gate port map (A => IN1, B => IN2,
    X => G1OUT);
    G2: AND_gate port map (A => G1OUT, B => IN3,
    X => G2OUT);
    INV: Inverter port map (A => IN5, X => INVOUT);
    G3: AND_gate port map (A => IN4, B => INVOUT,
    X => G3OUT);
    G4: OR_gate port map (A => G2OUT, B => G3OUT,
    X => OUT);
end architecture Structure;
```

39. Veja a Tabela P–7.

41. As portas AND são identificadas de cima para baixo como G1, G2, G3, G4. A porta OR é G5 e os inversores são identificados de cima para BAIXO como G6 e G7. Substitua A1, A2, B1, B2 por IN1, IN2, IN3, IN4 respectivamente. Substitua X por OUT.

```
entity Circuit5_62 is
    port (IN1, IN2, IN3, IN4: in bit; OUT: out bit);
end entity Circuit5_62;
architecture Logic of Circuit5_62 is
component AND_gate is
    port (A, B: in bit; X: out bit);
end component AND_gate;
component OR_gate is
    port (A, B, C, D: in bit; X: out bit);
```

▼ **TABELA P–7**

ENTRADAS				SAÍDA
A	*B*	*C*	*D*	*X*
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

```
end component OR_gate;
component Inverter is
    port (A: in bit; X: out bit);
end component Inverter;
    signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT,
    G6OUT, G7OUT: bit;
begin
    G1: AND_gate port map (A => IN1, B => IN2,
    X => G1OUT);
    G2: AND_gate port map (A => IN2, B => G6OUT,
    X => G2OUT);
    G3: AND_gate port map (A => G6OUT, B => G7OUT,
    X => G3OUT);
    G4: AND_gate port map (A => G7OUT, B => IN1,
    X => G4OUT);
    G5: OR_gate port map (A => G1OUT, B => G2OUT,
    C => G3OUT, D => G4OUT, X => OUT);
    G6: Inverter port map (A => IN3, X => G6OUT);
    G7: Inverter port map (A => IN4, X => G7OUT);
end architecture Logic;
```

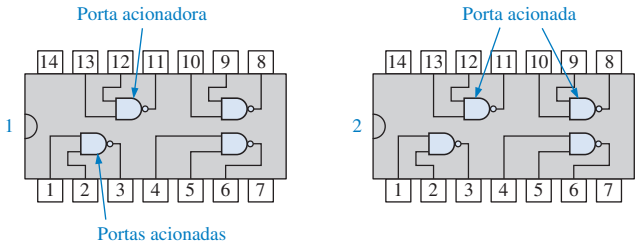
43. $X = ABC + D\overline{E}$. Como X é o mesmo que a saída G_3, G_1 ou G_2 falhou, com sua saída presa em nível BAIXO.

45. Veja a Figura P–26 na página 854.

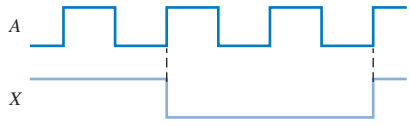
47. **(a)** Veja a Figura P–27.

(b) $X = E$

(c) $X = E$

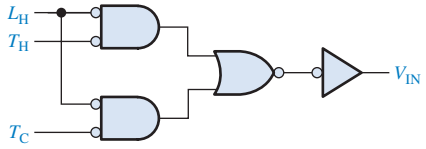


▲ **FIGURA P–26**



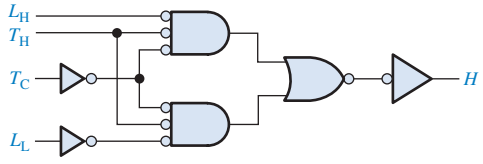
▲ **FIGURA P–27**

49. Veja a Figura P–28.

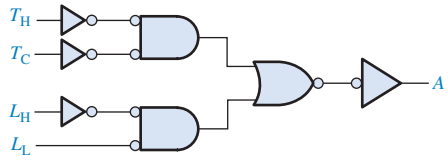


▲ **FIGURA P–28**

51. Veja a Figura P–29.



Circuito lógico do aquecedor



Circuito lógico do alarme

▲ **FIGURA P–29**

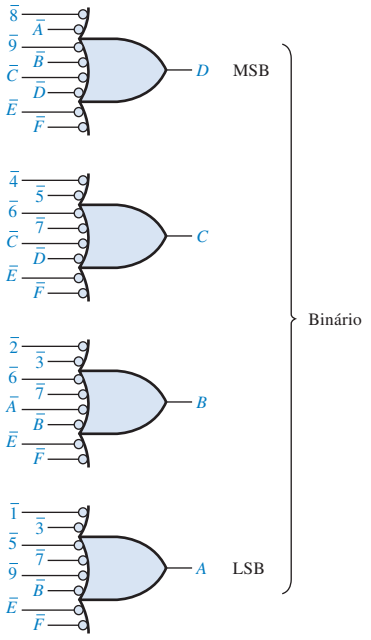
53. X = lâmpada ligada, A = chave na frente da porta, B = chave atrás da porta. Veja a Figura P–30.

▶ **FIGURA P–30**

55. Veja a Figura P–31. Os inversores (não mostrados) são usados para converter cada tecla acionada de nível ALTO para nível BAIXO.

57. O pino C da porta OR está aberto.

59. Sem defeito.



▲ **FIGURA P–31**

CAPÍTULO 6

1. **(a)** $A \oplus B = 0, \Sigma = 1, (A \oplus B)C_{ent.} = 0, AB = 1, C_{saída} = 1$
(b) $A \oplus B = 1, \Sigma = 0, (A \oplus B)C_{ent.} = 1, AB = 0, C_{saída} = 1$
(c) $A \oplus B = 1, \Sigma = 1, (A \oplus B)C_{ent.} = 0, AB = 0, C_{saída} = 0$

3. **(a)** $\Sigma = 1, C_{saída} = 0$;
(b) $\Sigma = 1, C_{saída} = 0$;
(c) $\Sigma = 0, C_{saída} = 1$;
(d) $\Sigma = 1, C_{saída} = 1$

5. 11100

7. $\Sigma_1 = 0110; \Sigma_2 = 1011; \Sigma_3 = 0110; \Sigma_4 = 0001; \Sigma_5 = 1000$

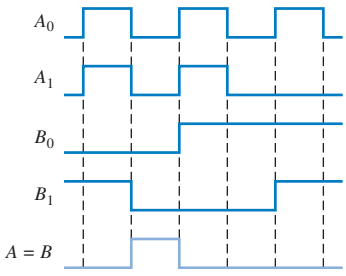
9. 225 ns

11. $A = B$ é nível ALTO quando $A_0 = B_0$ e $A_1 = B_1$; veja a Figura P–32.

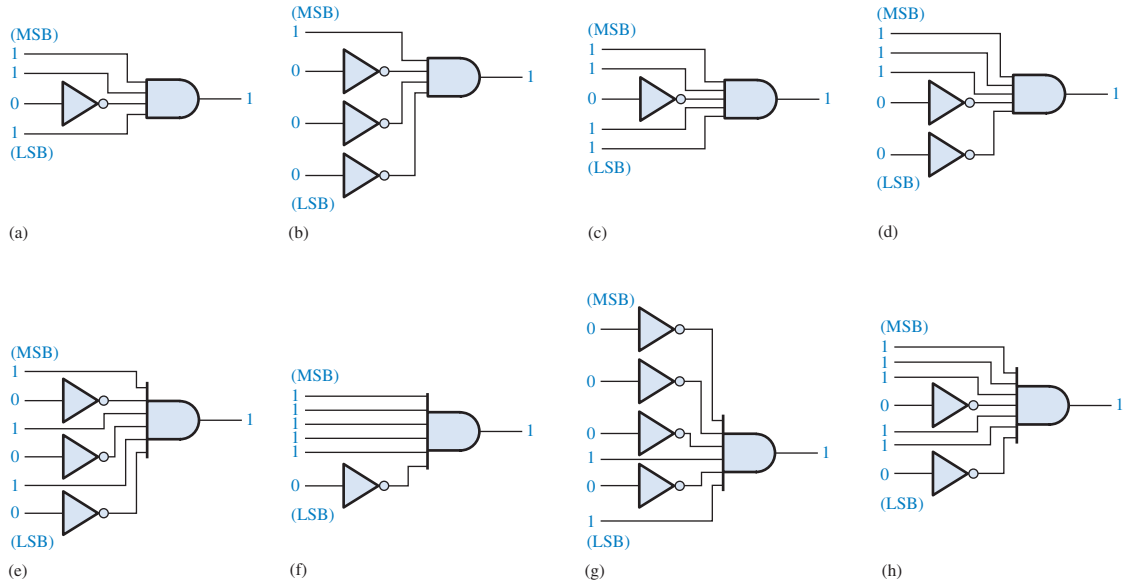
13. **(a)** $A > B = 1; A = B = 0; A < B = 0$
(b) $A < B = 1; A = B = 0; A > B = 0$
(c) $A = B = 1; A < B = 0; A > B = 0$

15. Veja a Figura P–33.

17. $X = A_3A_2\overline{A}_1\overline{A}_0 + \overline{A}_3\overline{A}_2\overline{A}_1A_0 + A_3\overline{A}_2A_1$

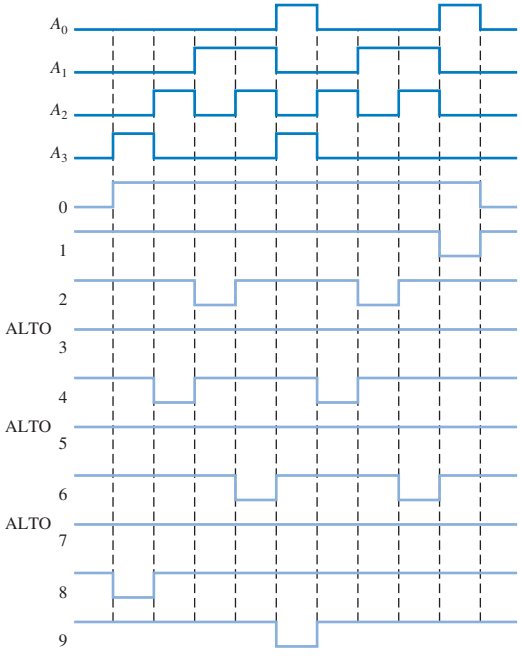


▲ **FIGURA P–32**



▲ **FIGURA P–33**

19. Veja a Figura P–34.



▲ **FIGURA P–34**

21. $A_3A_2A_1A_0 = 1011$, que é um código BCD inválido.

23. **(a)** $2 = 0010 = 0010_2$

(b) $8 = 1000 = 1000_2$

(c) $13 = 00010011 = 1101_2$

(d) $26 = 00100110 = 11010_2$

(e) $33 = 00110011 = 100001_2$

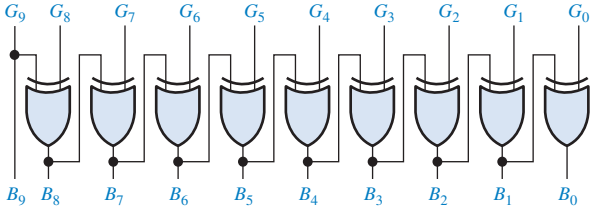
25. **(a)** 1010000000 Gray → 1100000000 binário

(b) 0011001100 Gray → 0010001000 binário

(c) 1111000111 Gray → 1010000101 binário

(d) 0000000001 Gray → 0000000001 binário

Veja a Figura P–35.



▲ **FIGURA P–35**

27. Veja a Figura P–36.

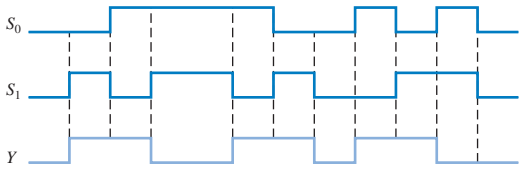
29. Veja a Figura P–37 na página 856.

31. Veja a Figura P–38.

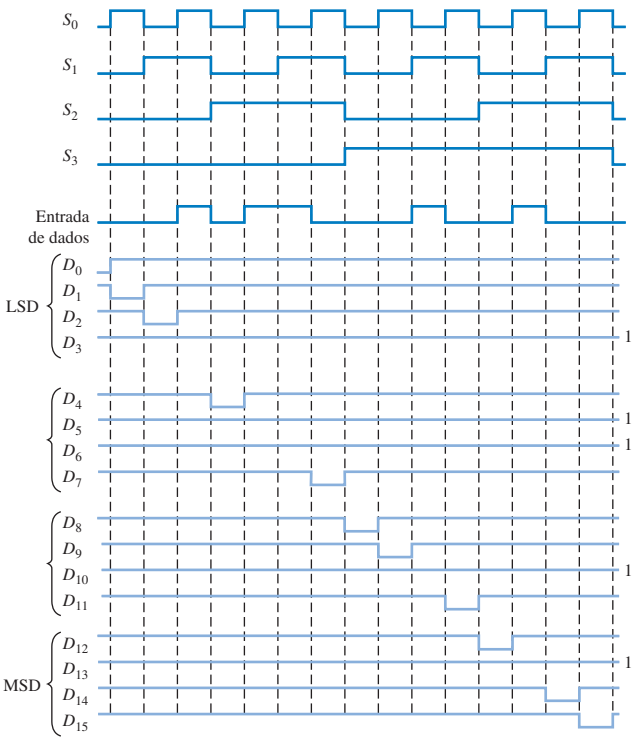
33. **(a)** OK

(b) segmento *g* "queimado"; saída G aberta

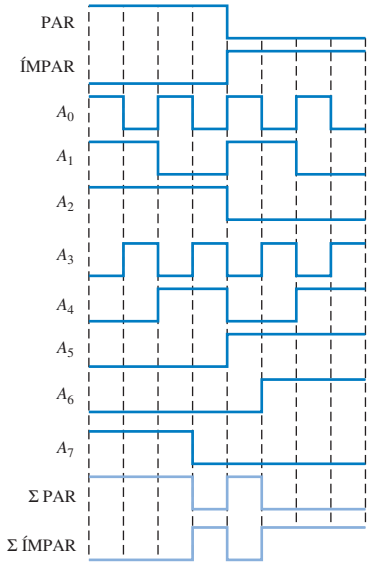
(c) a saída do segmento *b* presa em nível BAIXO.



▲ **FIGURA P–36**



▲ **FIGURA P–37**



▲ **FIGURA P–38**

35. (a) A entrada A_1 do somador superior está aberta: Todos os valores binários que correspondem aos números BCD 0, 1, 4, 5, 8 ou 9 terão um acréscimo de 2. Vemos isso primeiro com o valor BCD 0000 0000.

(b) A saída de carry do somador superior está aberta: Todos os valores que normalmente não envolvem um carry de saída terão um acréscimo de 32. Vemos isso primeiro com o valor BCD 0000 0000.

(c) A saída Σ_4 do somador superior está em curto-circuito com GND: Alguns valores binários acima de 15 terão uma diferença de 16. O primeiro valor BCD que indica isso é 0001 1000.

(d) A saída Σ_3 do somador inferior está em curto-circuito com GND: os outros 16 valores a partir do 16 terão uma diferença de 16. o primeiro valor BCD que indica isso é 0001 0110.

37. 1. Coloque um nível BAIXO no pino 7 (*Enable* – Habilitação).

2. Aplique um nível ALTO em D_0 e nível BAIXO de D_1 a D_7.

3. Coloque uma seqüência binária nas entradas de seleção e verifique as saídas e de acordo com a Tabela P–8.

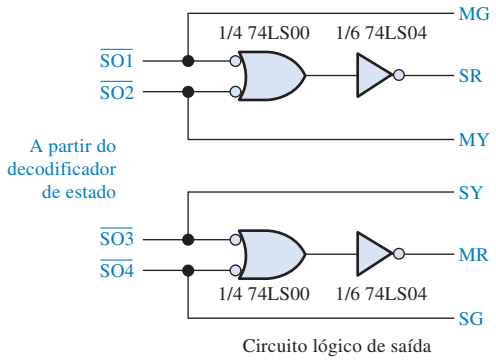
▼ **TABELA P–8**

S_2	S_1	S_0	Y	$\overline{Y}$
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

4. Repita a seqüência binária nas entradas de seleção para cada conjunto de dados de entrada listados na Tabela P–9. Um nível ALTO na saída Y deve ocorrer apenas para as combinações correspondentes das entradas de seleção mostradas.

39. Aplique um nível ALTO de cada vez na Entrada de dados, D_0 a D_7, com as entradas restante em nível BAIXO. Para cada nível ALTO aplicado a uma entrada de dados, coloque a seqüência das oito combinações binárias na entradas de seleção ($S_2S_1S_0$) e verifique se há nível ALTO na saída de dados correspondente e nível BAIXO nas outras saídas de dados.

41. Veja a Figura P–39.



▲ **FIGURA P–39**

▼ **TABELA P–9**

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y	$\overline{Y}$	S_2	S_1	S_0
L	H	L	L	L	L	L	L	1	0	0	0	1
L	L	H	L	L	L	L	L	1	0	0	1	0
L	L	L	H	L	L	L	L	1	0	0	1	1
L	L	L	L	H	L	L	L	1	0	1	0	0
L	L	L	L	L	H	L	L	1	0	1	0	1
L	L	L	L	L	L	H	L	1	0	1	1	0
L	L	L	L	L	L	L	H	1	0	1	1	1

43. $\Sigma = \overline{A}\,\overline{B}C_{ent.} + \overline{A}B\overline{C}_{ent.} + A\overline{B}\,\overline{C}_{ent.} + ABC_{ent.}$

$C_{saída} = \overline{A}BC_{ent.} + A\overline{B}C_{ent.} + AB\overline{C}_{ent.} + ABC_{ent.}$

Veja a Figura P–40.

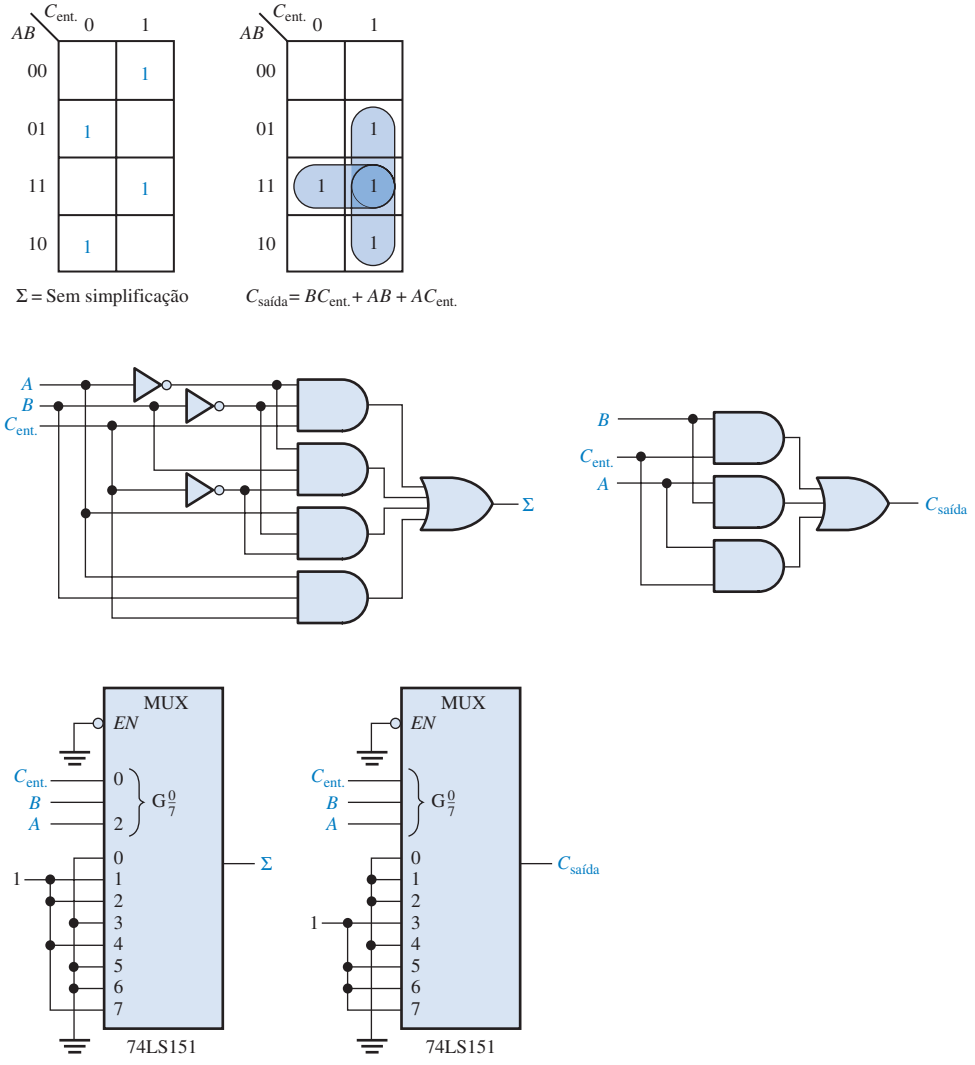
45. Veja o diagrama em bloco na Figura P–41 na página 858.

47. Veja a Figura P–42.

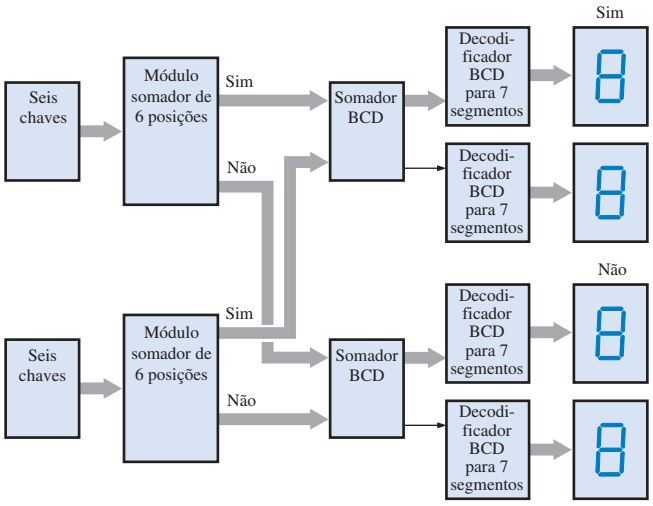
49. Veja a Figura P–43.

51. A saída de carry do somador LSB está aberta.

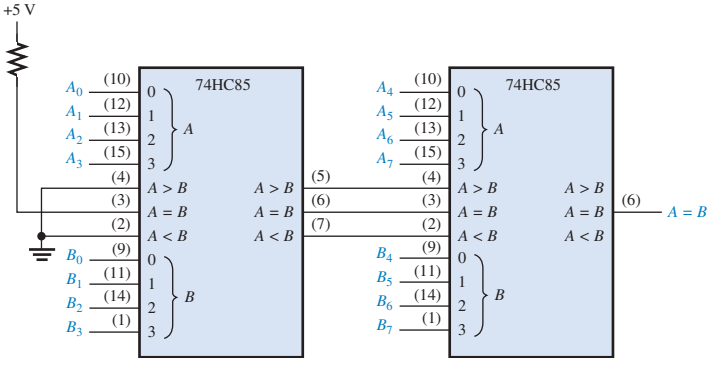
53. O pino 12 do 74148 superior está aberto.



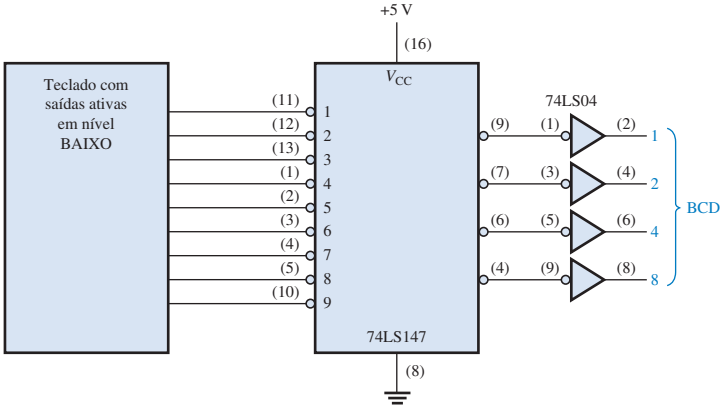
▶ **FIGURA P–40**



▶ **FIGURA P–41**



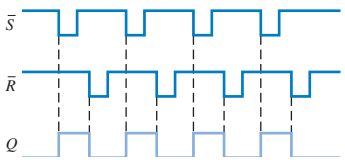
▶ **FIGURA P–42**



▶ **FIGURA P–43**

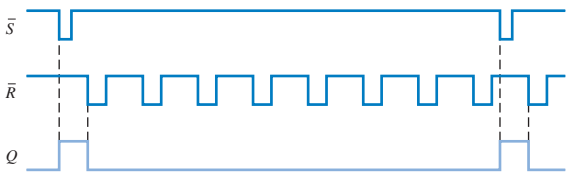
CAPÍTULO 7

1. Veja a Figura P–44.



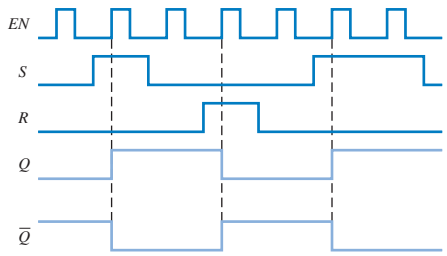
▲ **FIGURA P–44**

3. Veja a Figura P–45.



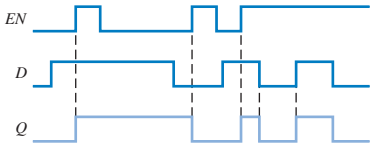
▲ **FIGURA P–45**

5. Veja a Figura P–46.



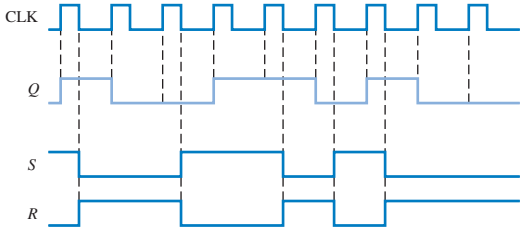
▲ **FIGURA P–46**

7. Veja a Figura P–47.



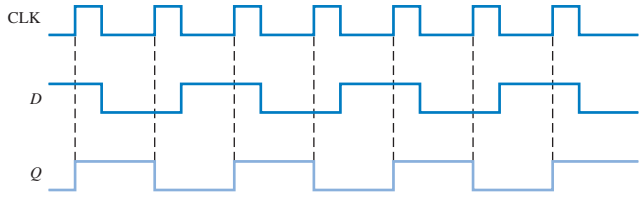
▲ **FIGURA P–47**

9. Veja a Figura P–48.



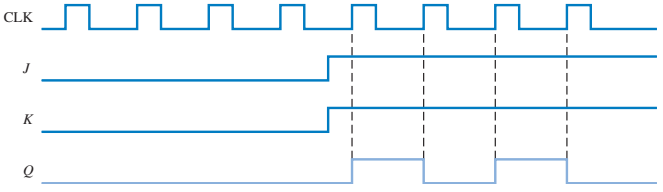
▲ **FIGURA P–48**

11. Veja a Figura P–49.



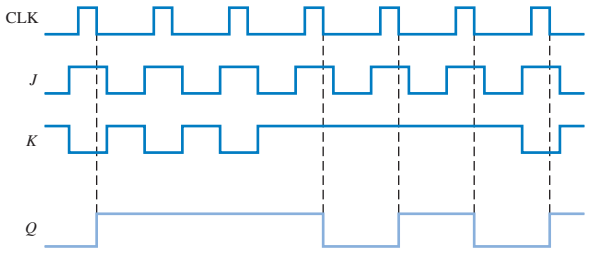
▲ **FIGURA P–49**

13. Veja a Figura P–50.



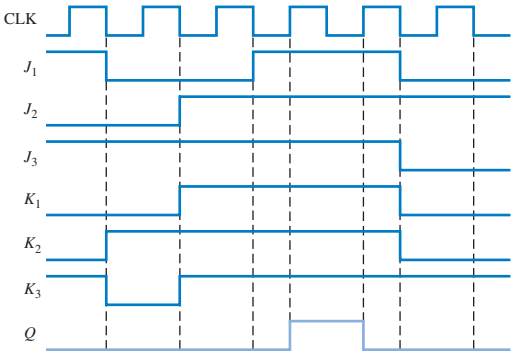
▲ **FIGURA P–50**

15. Veja a Figura P–51.



▲ **FIGURA P–51**

17. Veja a Figura P–52.



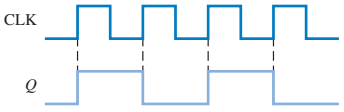
▲ **FIGURA P–52**

19. A corrente e a tensão de alimentação cc.

21. 14,9 MHz

23. 150 mA, 750 mW

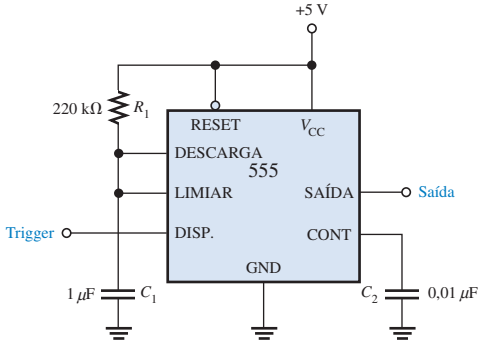
25. É um divisor por 2; veja a Figura P–53.



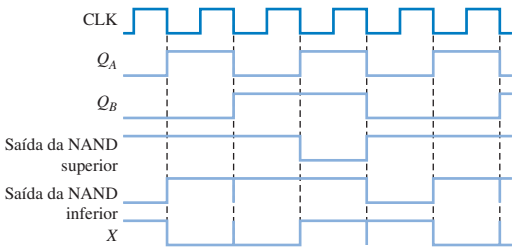
▲ **FIGURA P–53**

27. 4,62μs

29. $C_1 = 1\mu F$, $R_1 = 227\ k\Omega$ (use 220 kΩ). Veja a Figura P–54.

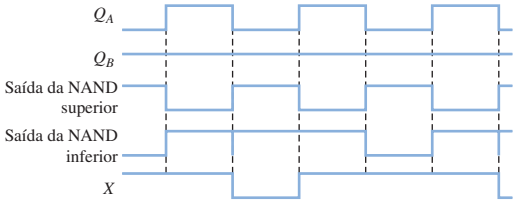


▲ **FIGURA P–54**



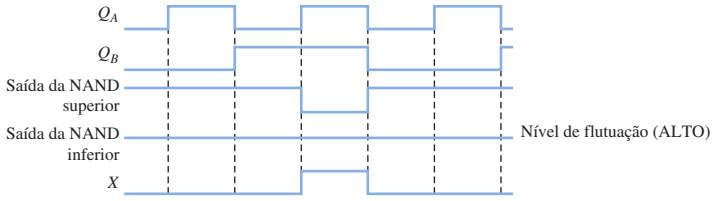
(a)

(b) O mesmo que em (a)



(c)

(d) X = BAIXO se $Q_B = 1$; $X = \bar{Q}_A$ se $Q_B = 0$



(e)

▲ **FIGURA P–55**

31. $R_1 = 18\ \text{k}\Omega$, $R_2 = 9{,}1\ \text{k}\Omega$.

33. Os fios do pino 6 ao pino 10 e o fio GND estão trocados no protoboard.

35. $\overline{CLR}$ em curto-circuito para GND.

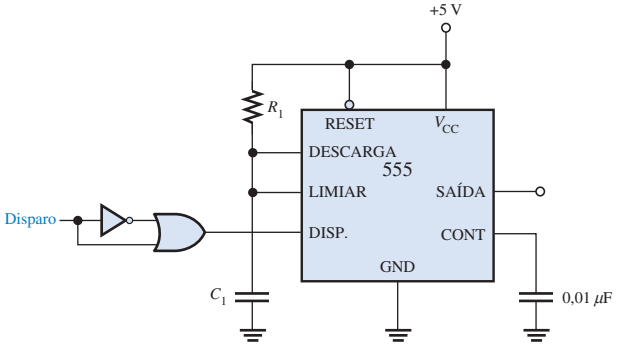
37. Veja a Figura P–55. Os atrasos não são mostrados.

39. Veja a Figura P–56.

4 s: $C_1 = 1\ \mu\text{F}$, $R_1 = 3{,}63\ \text{M}\Omega$ (use $3{,}9\ \text{M}\Omega$)

25 s: $C_1 = 2{,}2\ \mu\text{F}$, $R_1 = 10{,}3\ \text{M}\Omega$ (use $10\ \text{M}\Omega$)

41. Veja a Figura P–57.



▲ **FIGURA P–56**

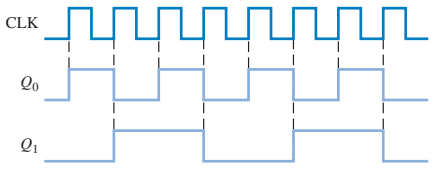
43. A saída $\overline{Q}$ de U1 está aberta.

45. A entrada $\overline{SET}$ de U1 está aberta.

47. A entrada K de U2 está aberta.

CAPÍTULO 8

1. Veja a Figura P–58.



▲ **FIGURA P–58**

3. O atraso no pior caso é 24 ns; ele ocorre quando todos os flip-flops mudam de estado de 011 para 100 ou de 111 para 000.

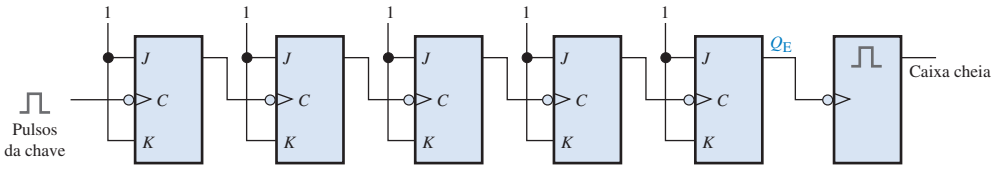
5. 8 ns

7. Inicialmente cada flip-flop é resetado.

Para CLK1:

$J_0 = K_0 = 1$ Portanto Q_0 vai para 1.

$J_1 = K_1 = 0$ Portanto Q_1 permanece em 0.



▲ **FIGURA P–57**

$J_2 = K_2 = 0$ Portanto Q_2 permanece em 0.
$J_3 = K_3 = 0$ Portanto Q_3 permanece em 0.

Para CLK2:

$J_0 = K_0 = 1$ Portanto Q_0 vai para 0.
$J_1 = K_1 = 1$ Portanto Q_1 vai para 1.
$J_2 = K_2 = 0$ Portanto Q_2 permanece em 0.
$J_3 = K_3 = 0$ Portanto Q_3 permanece em 0.

Para CLK13:

$J_0 = K_0 = 1$ Portanto Q_0 vai para 0.
$J_1 = K_1 = 0$ Portanto Q_1 permanece em 1.
$J_2 = K_2 = 0$ Portanto Q_2 permanece em 0.
$J_3 = K_3 = 0$ Portanto Q_3 permanece em 0.

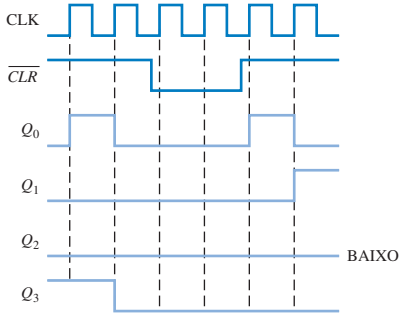
A continuação desse procedimento para os próximos sete pulsos de clock mostra que o contador avança numa seqüência BCD.

9. Veja a Figura P–59.

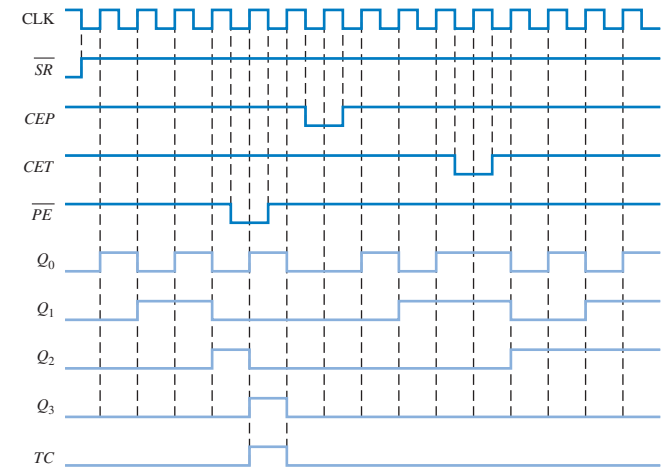
11. Veja a Figura P–60.

13. Veja a Figura P–61.

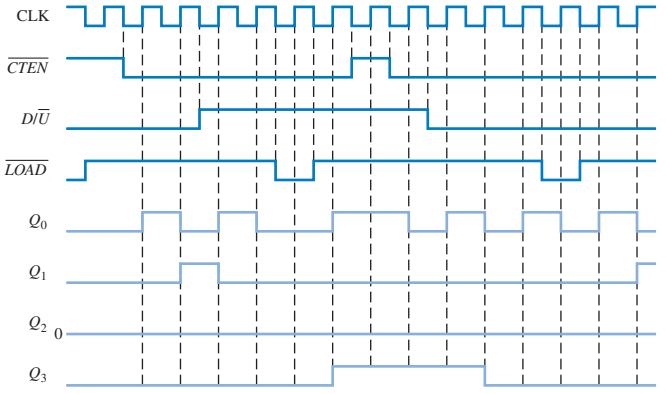
15. A seqüência é 0000, 1111, 1110, 1101, 1010, 0101. O contador trava nos estados 1010 e 0101 alternando entre eles.



▲ **FIGURA P–59**



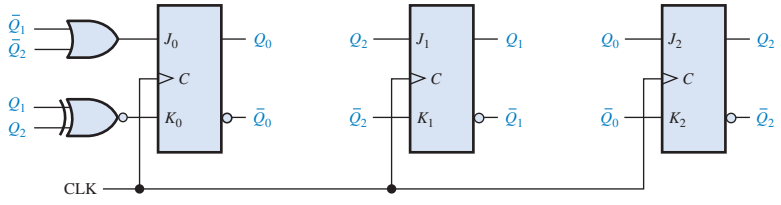
▲ **FIGURA P–60**



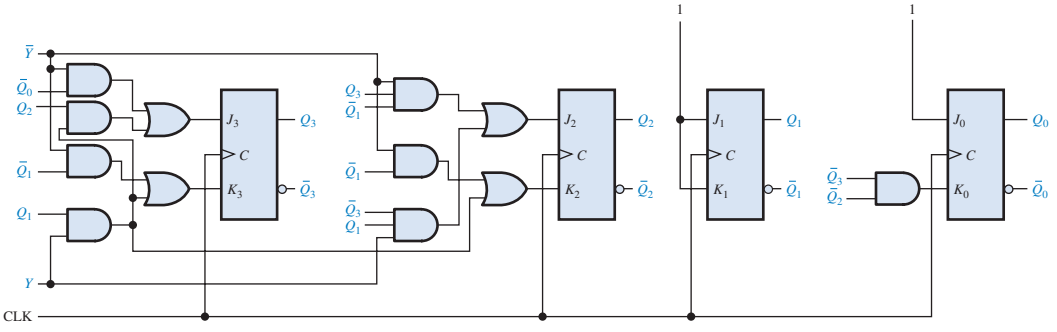
▲ **FIGURA P–61**

17. Veja a Figura P–62.

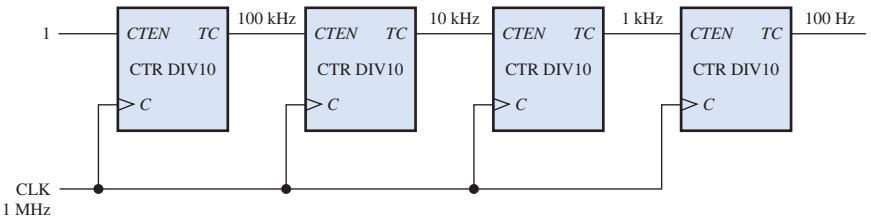
19. Veja a Figura P–63.



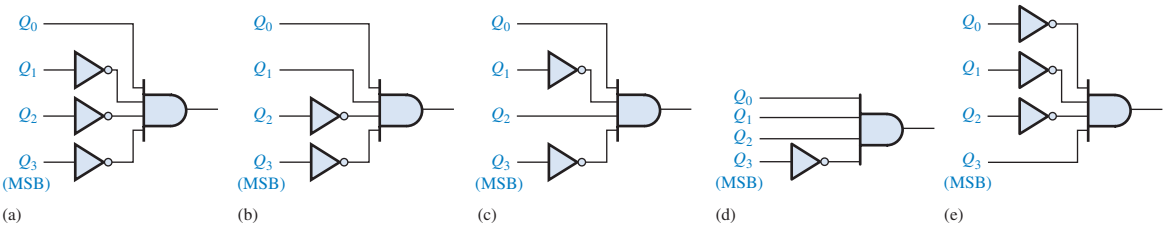
▶ **FIGURA P–62**



▲ **FIGURA P–63**



▶ **FIGURA P–64**



▲ **FIGURA P–65**

21. Veja a Figura P–64 para a divisão por 10.000. Acrescente mais um contador DIV10 para criar um divisor por 100.000.

23. Veja a Figura P–65.

25. CLK2, saída 0; CLK4, saídas 2, 0; CLK6, saída 4; CLK8, saídas 6, 4 e 0; CLK10, saída 8; CLK12, saídas 10, 8; CLK14, saída 12; CLK16, saídas 14, 12, 8

27. Um glitch na saída da porta AND ocorre na transição de 111 para 000. Este é eliminado fazendo a operação AND de $\overline{\text{CLK}}$ com as saídas do contador (strobe) ou usar o código Gray.

29. Dezenas das horas: 0001
Unidades das horas: 0010
Dezenas dos minutos: 0000
Unidades dos minutos: 0001
Dezenas dos segundos: 0000
Unidades dos segundos: 0010

31. 64

33. **(a)** Q_0 e Q_1 não mudam dos seus estados iniciais.
(b) operação normal, exceto Q_0 em flutuação
(c) a forma de onda de Q_0 é normal; Q_1 permanece no seu estado inicial.
(d) operação normal
(e) o contador não muda do seu estado inicial.

35. A entrada *K* do FF1 tem que ser conectada em GND em vez da entrada *J*. Verifique se há um erro de conexão.

37. A conexão de Q_0 com a porta AND está aberta, fazendo com que essa entrada se comporte como nível ALTO.

39. Veja a Tabela P–10.

41. A porta que decodifica o 6 interpreta a contagem 4 como sendo 6 (0110) e reseta o contador de volta para 0 (na realidade 0010 já que Q_1 está aberta). A seqüência aparente das dezenas do contador é 0010, 0011, 0010, 0011, 0110.

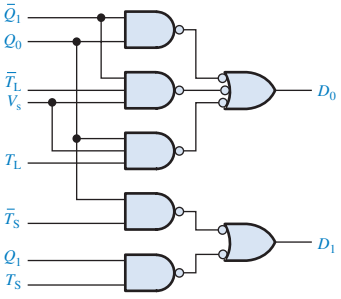
43. Veja a Figura P–66.

45. Aumentando em 2,4 vezes a constante de tempo $R_{EXT}C_{EXT}$ do monoestável de 25 s.

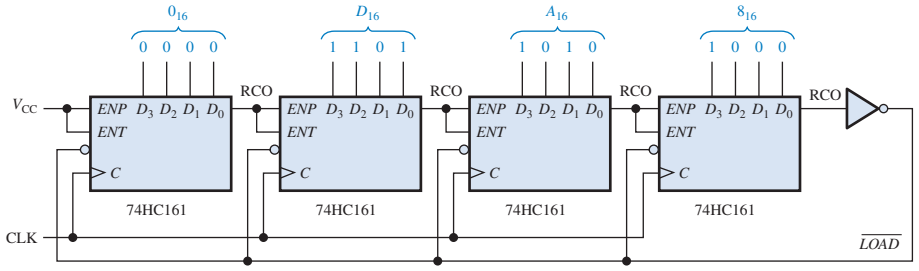
47. Veja a Figura P–67.

▼ **TABELA P–10**

ESTÁGIO	ABERTO	CONTAGEM CARREGADA	$f_{SAÍDA}$
1	0	63C1	250,006 Hz
1	1	63C2	250,012 Hz
1	2	63C4	250,025 Hz
1	3	63C8	250,050 Hz
2	0	63D0	250,100 Hz
2	1	63E0	250,200 Hz
2	2	63C0	250 Hz
2	3	63C0	250 Hz
3	0	63C0	250 Hz
3	1	63C0	250 Hz
3	2	67C0	256,568 Hz
3	3	6BC0	263,491 Hz
4	0	73C0	278,520 Hz
4	1	63C0	250 Hz
4	2	63C0	250 Hz
4	3	E3C0	1,383 kHz



▲ **FIGURA P–66**



▶ **FIGURA P–67**

49. Veja a Figura P–68.

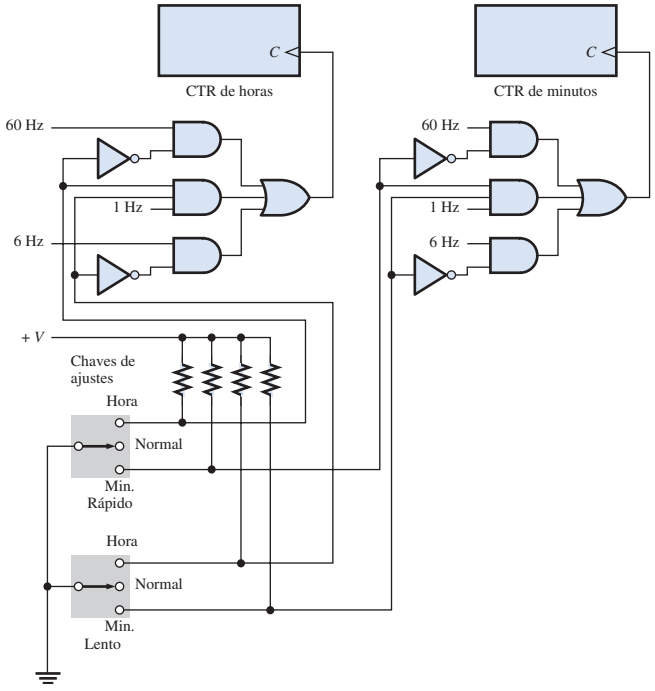
51. Veja a Figura P–69.

53. Veja a Figura P–70.

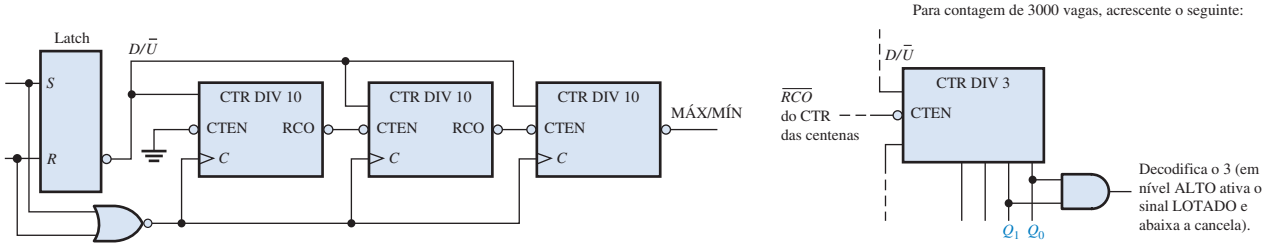
55. A saída Q de U3 está aberta.

57. O pino A de G3 está aberto.

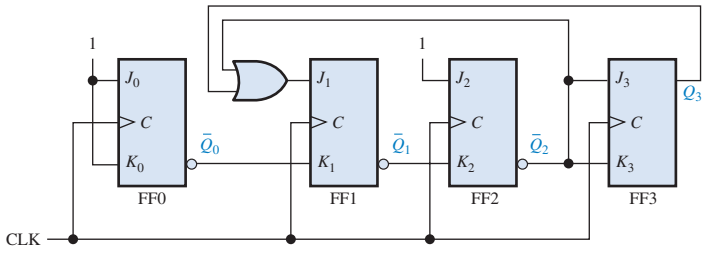
59. O pino 9 está aberto.



▶ **FIGURA P–68**



▲ **FIGURA P–69**



▶ **FIGURA P–70**

CAPÍTULO 9

1. Porque armazenam dados binários.

3. Veja a Figura P–71.

5. Inicialmente: 101001111000
CLK1: 010100111100
CLK2: 001010011110
CLK3: 000101001111
CLK4: 000010100111
CLK5: 100001010011
CLK6: 110000101001
CLK7: 111000010100
CLK8: 011100001010
CLK9: 001110000101
CLK10: 000111000010
CLK11: 100011100001
CLK12: 110001110000

7. Veja a Figura P–72.

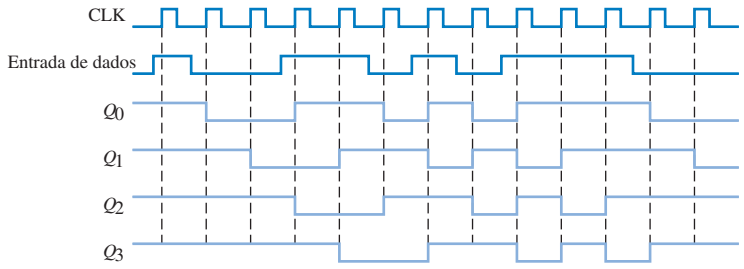
9. Veja a Figura P–73.

11. Veja a Figura P–74.

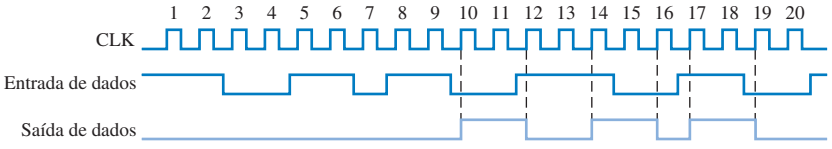
13. Veja a Figura P–75.

15. Veja a Figura P–76.

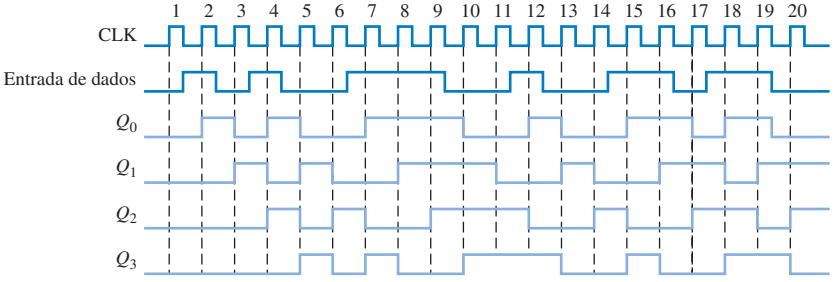
17. Veja a Figura P–77.



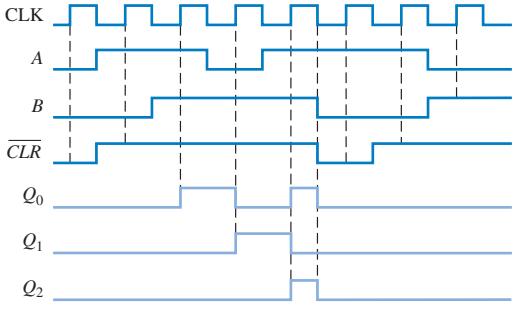
▶ FIGURA P–71



▶ FIGURA P–72

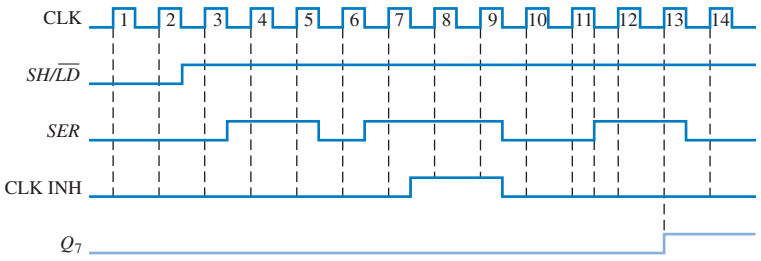


▶ FIGURA P–73

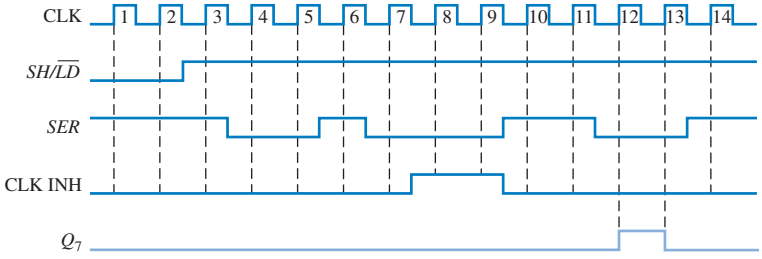


Q_3 a Q_7 permanece em nível BAIXO.

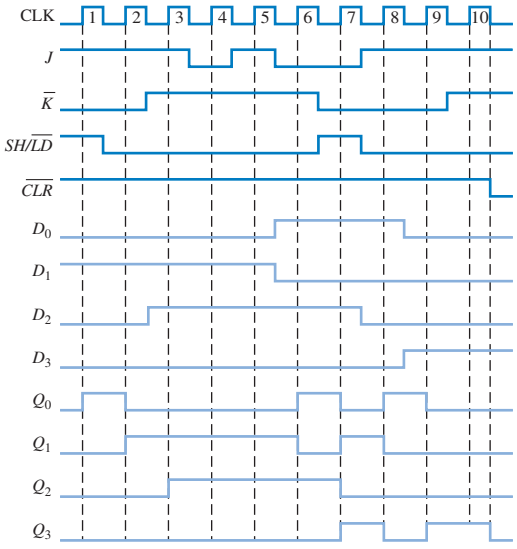
▶ FIGURA P–74



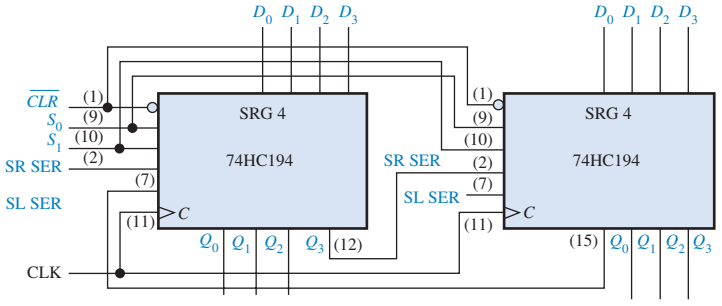
▶ **FIGURA P–75**



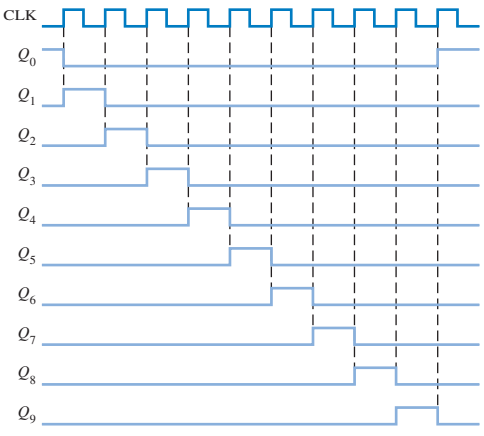
▶ **FIGURA P–76**



▲ **FIGURA P–77**



▲ **FIGURA P–78**



▲ **FIGURA P–79**

19. Inicialmente (76) 01001100

CLK1:	10011000	esquerda
CLK2:	01001100	direita
CLK3:	00100110	direita
CLK4:	00010011	direita
CLK5:	00100110	esquerda
CLK6:	01001100	esquerda
CLK7:	00100110	direita
CLK8:	01001100	esquerda
CLK9:	00100110	direita
CLK10:	01001100	esquerda
CLK11:	10011000	esquerda

21. Veja a Figura P–78.

23. **(a)** 3 **(b)** 5
(c) 7 **(d)** 8

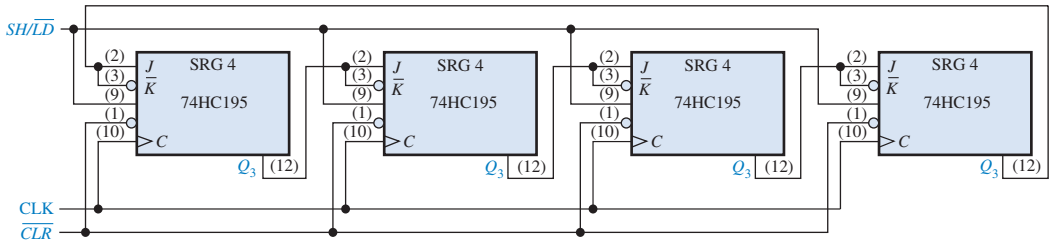
25. Veja a Figura P–79.

27. Veja a Figura P–80.

29. Um código incorreto pode ser gerado.

31. A entrada D_3 está aberta.

33. **(a)** A falta do sinal de tecla pressionada é por causa de um defeito na porta NAND (OR negativa) ou no monoestável; en-



▲ **FIGURA P–80**

trada de clock (C) aberta do registrador do código da tecla; entrada $SH/\overline{LD}$ do registrador de código da tecla aberta.

(b) Diodo na terceira linha aberto; saída Q_2 do contador em anel aberta.

(c) A entrada da porta NAND (OR negativa) conectada à primeira coluna está aberta ou em curto-circuito.

(d) A entrada "2" do codificador de coluna está aberta.

35. (a) O conteúdo do registrador de saída

(b) O conteúdo dos dois registradores não mudam.

(c) A saída do terceiro estágio do registrador de saída de dados permanece em nível ALTO.

(d) O gerador de clock é desabilitado após cada pulso pelo flip-flop sendo continuamente setado e em seguida resetado.

37. Registrador de deslocamento A: 1001
Registrador de deslocamento C: 00000100

39. Flip-flop de controle: 7476
Gerador de clock: 555
Registrador de entrada de dados: 74LS164
Registrador de saída de dados: 74LS199
Monoestável: 74121.

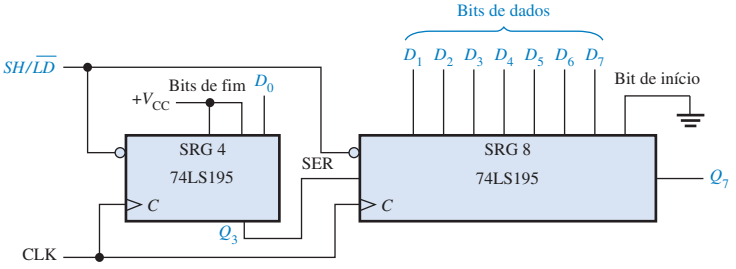
41. Veja a Figura P–81.

43. Veja a Figura P–82.

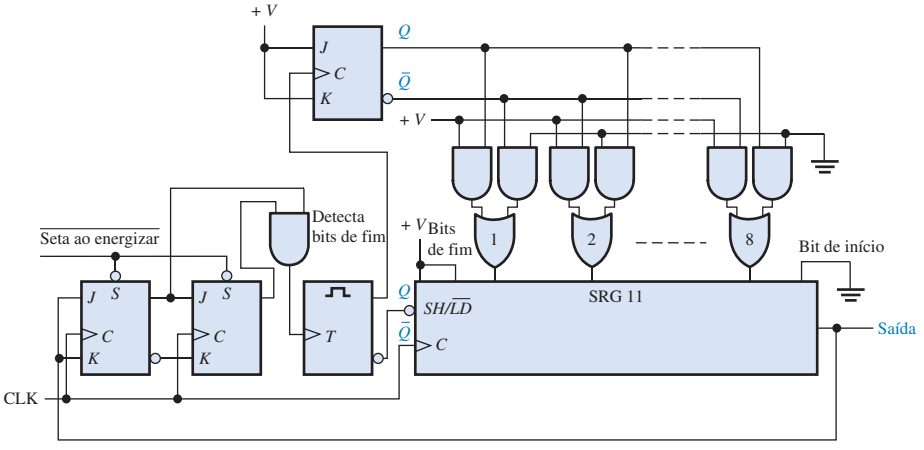
45. A entrada CLK de U3 está aberta.

47. O pino 14 está aberto.

49. A entrada CLK de U6 está aberta.



▶ **FIGURA P–81**



▶ **FIGURA P–82**

CAPÍTULO 10

1. **(a)** ROM **(b)** RAM

3. O *barramento de endereço* proporciona a transferência de códigos de endereço para a memória para o acesso de qualquer posição da memória em qualquer ordem numa operação de leitura ou escrita. O *barramento de dados* proporciona a transferência de dados entre o microprocessador e a memória ou dispositivos I/O.

5.

	Bit 0	Bit 1	Bit 2	Bit 3
Linha 0	1	0	0	0
Linha 1	0	0	0	0
Linha 2	0	0	1	0
Linha 3	0	0	0	0

7. 512 linhas × 128 colunas de 8 bits.

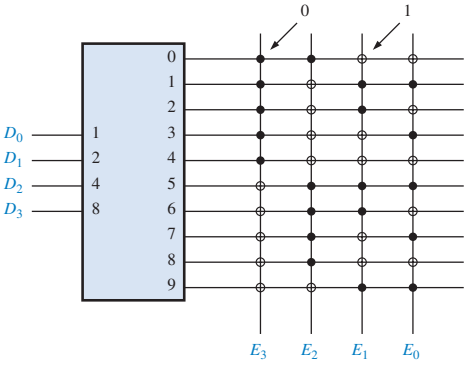
9. Uma SRAM armazena bits em flip-flops indefinidamente enquanto a tensão de alimentação estiver conectada. Uma DRAM armazena bits em capacitores que precisam ser reavivados (refresh) periodicamente para reter os dados.

11. Veja a Tabela P–11.

▼ **TABELA P–11**

ENTRADAS		SAÍDAS			
A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0

13. Veja a Figura P–83.



▲ **FIGURA P–83**

15. Fusíveis queimados: 1–17, 19–23, 25–31, 34, 37, 38, 40–47, 53, 55, 58, 61, 62, 63, 65, 67, 69

17. Use oito DRAM de 16k × 4 com dezesseis linhas de endereço. Duas dessas linhas são decodificadas para habilitar os chips de memória selecionadas. Quatro linhas de dados são conectadas em cada chip.

19. 8 bits, 64k de palavras; 4 bits, 256k de palavras.

21. Endereço inferior: $FC0_{16}$

Endereço superior: FFF_{16}

23. Um disco rígido é formado por trilhas e setores. Cada trilha é dividida em um número de setores sendo que cada setor de uma trilha tem um endereço físico. Os discos rígidos têm tipicamente de algumas centenas a alguns milhares de trilhas.

25. Uma fita magnética tem um tempo de acesso mais longo que um disco rígido porque os dados têm que ser acessados sequencialmente em vez de aleatoriamente.

27. O conteúdo do checksum está errado.

29. **(a)** ROM 2 **(b)** ROM 1 **(c)** Todos ROMs

31. 10

33. A PROM retém o código quando a alimentação é desligada. O código na PROM não pode ser alterado, a menos que seja uma EEPROM.

35. Para acomodar um código de entrada de 5 bits, o registrador de deslocamento C tem que ser carregado com cinco 0s em vez de quatro. O nível ALTO (1) em que ser movido uma posição para a esquerda nas entradas em paralelo.

CAPÍTULO 11

1. $X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}C$

3. **(a)** PAL16L/2 é um dispositivo lógico de arranjo programável com 16 entradas e duas saídas ativas em nível BAIXO.

(b) PAL12H6 é um dispositivo lógico de arranjo programável com 12 entradas e 6 saídas ativas em nível ALTO.

5. Uma CPLD consiste basicamente de múltiplas SPLDs que podem ser conectadas com um arranjo de interconexões programáveis.

7. **(a)** $\overline{A}BC\overline{D}$ **(b)** $ABC(\overline{D} + \overline{E}) = ABC\overline{D} + ABC\overline{E}$

9. $X = A\overline{B} + \overline{A}B$

11. $X_1 = A\overline{B}C\overline{D} + \overline{A}BCD + ABC\overline{D}$;
$X_2 = ABCD + AB\overline{C}D + \overline{A}BC\overline{D} + A\overline{B}CD$

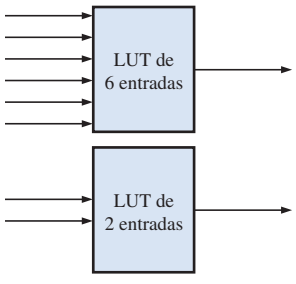
13. **(a)** Combinacional; 1 **(b)** Registrada; 0

15. **(a)** Registrada **(b)** GCK1 **(c)** 0 **(d)** 0

17. Saída de soma-de-produtos = $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$

19. LUT para circuitos lógicos combinacional, somador e registrador.

21. Veja a Figura P–84.

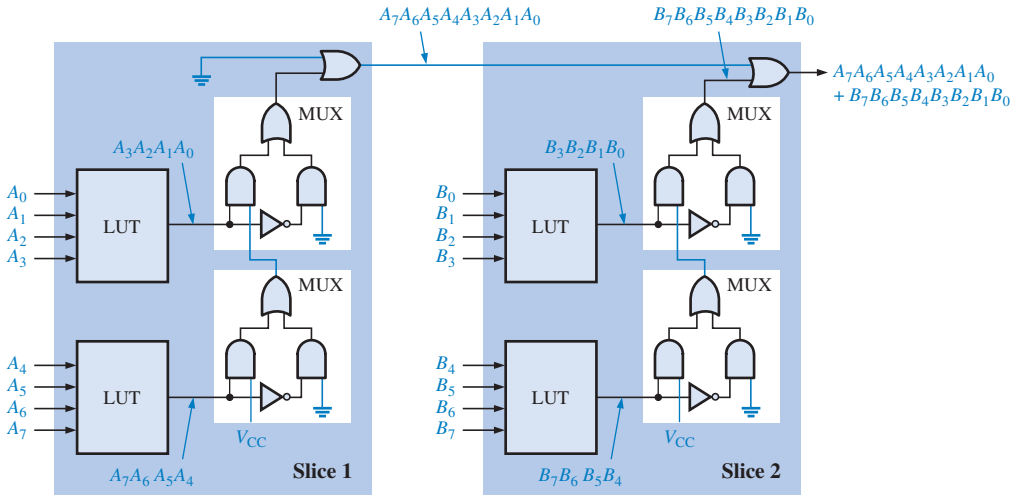


▲ **FIGURA P–84**

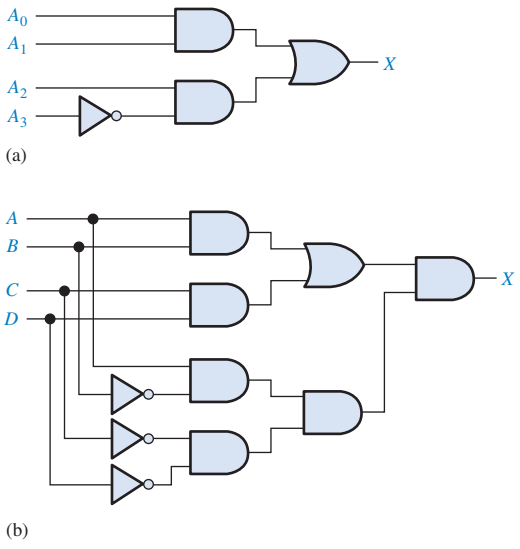
23. Veja a Figura P–85.

25. Um *slice*.

27. Veja a Figura P–86.



▶ **FIGURA P–85**



▲ **FIGURA P–86**

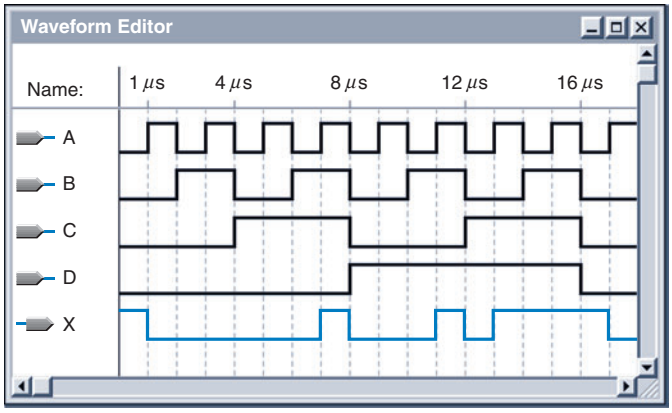
29. Veja a Figura P–87.

31. Entrada de deslocamento = 1, os dados são aplicados em SDI, passam através do MUX e são carregados no registrador de captura A na borda de subida do pulso de clock. A partir da saída do registrador de captura A, os dados vão para o MUX superior e são carregados no registrador de captura B na borda de descida do pulso de clock.

33. PDI/O = 0 e OE = 0. Os dados são aplicados no pino de entrada e vão através do MUX selecionado para a lógica programável interna.

35. 00001100101010001111011

0 00001100101010001111011
1 00001100101010001111011
3 00001100101010001111011
6 00001100101010001111011
12 00001100101010001111011
9 00001100101010001111011
2 00001100101010001111011
5 00001100101010001111011
10 00001100101010001111011
4 00001100101010001111011



▶ **FIGURA P–87**

8 00001100101000**1**1111011
1 00001100101000**1**111011
3 00001100101000**1**11011
7 00001100101000**1**1011
15 00001100101000**1**111011
14 00001100101000**1**11011
13 00001100101000**1**1011
11 00001100101000**1**1011

37. A entrada D para o circuito lógico está com defeito ou desconectada.

CAPÍTULO 12

1. CPU, memória, portas de I/O, barramentos.

3. Um barramento é um conjunto de conexões e especificações elétricas usado para a movimentação de informações em um computador.

5. ALU, decodificador de instrução, arranjo de registradores e unidade de controle.

7. Barramento de endereço, barramento de dados e barramento de controle.

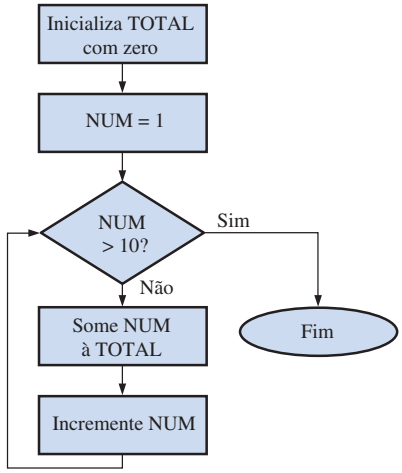
9. Busca, decodificação e execução.

11. CD, DS, SS, ES, FS, GS

13. AH e AL são registradores de 8 bits e representam as partes alta e baixa do registrador AX de 16 bits. O EAX é um registrador de 32 bits o qual inclui o registrador AX como os 16 bits menos significativos.

15. O emparelhamento permite que duas instruções sejam executadas ao mesmo tempo.

17. Veja a Figura P–88.



▲ FIGURA P–88

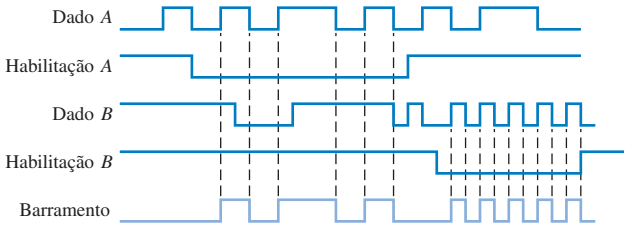
19. Quando a instrução **mov ax, [bx]** é executada, a palavra na memória apontada pelo registrador bx é copiada para o registrador ax.

21. Numa consulta em I/O, a CPU consulta cada dispositivo para saber se ele precisa de um serviço; em um sistema de interrupção ativada por I/O, o dispositivo periférico sinaliza para a CPU quando precisa de um serviço.

23. Uma instrução de programa que invoca uma rotina de serviço de interrupção.

25. a CPU não participa das operações num ciclo DMA.

27. Veja a Figura P–89.



▲ FIGURA P–89

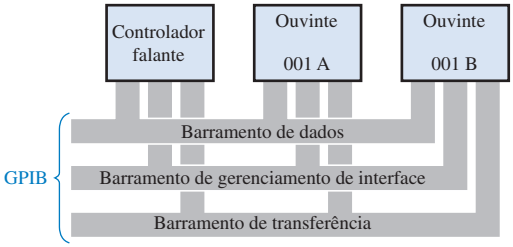
29. O barramento local é uma coleção de barramentos que fazem a interface direta com o processador. O barramento PCI é usado para dispositivos de expansão e é conectado ao barramento local através de um controlador e barramento.

31. O barramento PCI é um barramento de expansão de 33 ou 66 MHz e 32 ou 66 bits. O barramento ISA é um barramento de expansão de 8,33 MHz e 8 ou 16 bits.

33. DCE significa equipamento de comunicação de dados, tal como um modem. DTE significa equipamento terminal de dados, tal como um computador. Esses dois acrônimos são associados aos padrões RS-232/EIA-232.

35. seis

37. Um controlador está enviando dados para dois "ouvintes". Os dois primeiros bytes de dados (3F e 41) vão para o ouvinte com endereço 001A. O dois próximos bytes vão para o ouvinte com endereço 001B. Os sinais de handshaking ($\overline{\text{DAV}}$, $\overline{\text{NRFD}}$ e $\overline{\text{NDAC}}$) indicam que a transferência de dados foi feita com sucesso. Veja a Figura P–90.



▲ FIGURA P–90

CAPÍTULO 13

1. Um conversor analógico-digital converte um sinal analógico em um código digital.

3. Um conversor digital-analógico transforma um código digital para o sinal analógico correspondente.

5. Veja a Figura P–91.

7. 1000, 1110, 1011, 0100, 0001, 0111, 1110, 1011, 0100.

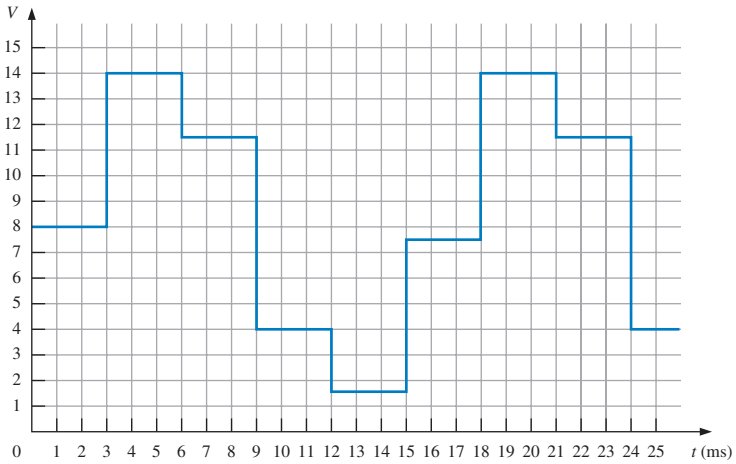
9. Veja a Figura P–92.

11. 330 kΩ

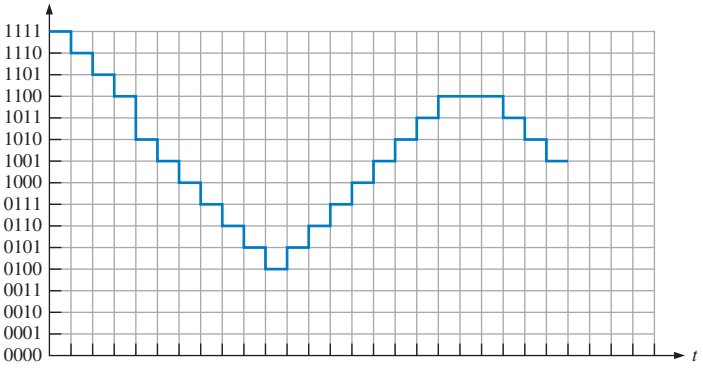
13. 000, 001, 100, 110, 101, 100, 011, 010, 001, 001, 011, 110, 111, 111, 111, 111, 111, 111, 111, 100

15. Veja a Tabela P–12.

17. 8000 MB/s



▶ FIGURA P–91



▶ FIGURA P–92

▼ TABELA P–12

SAR	COMENTÁRIO
1000	Maior que $V_{ent.}$, reseta MSB
0100	Menor que $V_{ent.}$, mantém I
0110	Igual a $V_{ent.}$, mantém (o estado final)

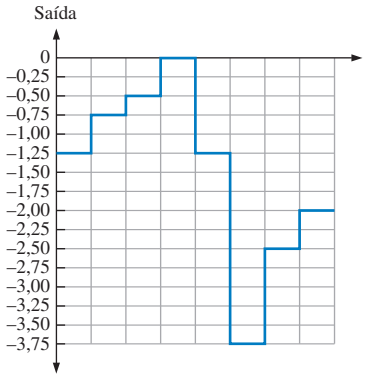
19. 1.000.000.000

21. Expedição de instrução (DP): pacotes de instruções são divididos em pacotes de execução e associados a unidades funcionais; Decodificação de instrução (DC): Instruções são decodificadas.

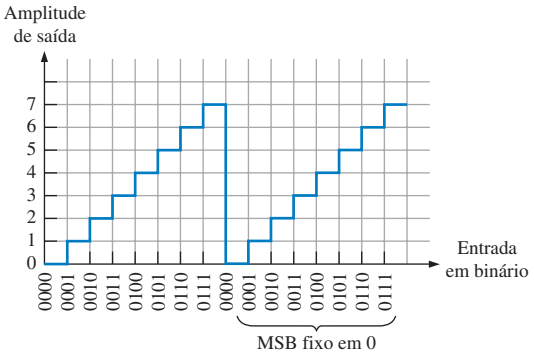
23. Veja a Figura P–93.

25. **(a)** 14,3% **(b)** 0,098% **(c)** 0,00038%

27. Veja a Figura P–94.



▲ FIGURA P–93



▲ FIGURA P–94

CAPÍTULO 14

1. Não. Porque $V_{OH(min)} < V_{IH(min)}$

3. 0,15 V no estado ALTO; 0,25 V no estado BAIXO.

5. Porta *C*.

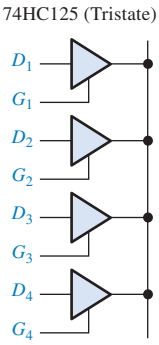
7. 12 ns

9. Porta *C*.

11. Sim, G_2

13. **(a)** ligado
(b) desligado
(c) desligado
(d) ligado

15. Veja a Figura P–95 para um possível circuito.



▲ **FIGURA P–95**

17. **(a)** ALTO **(b)** flutuação
(c) ALTO **(d)** Hi-Z

19. **(a)** BAIXO **(b)** BAIXO
(c) BAIXO

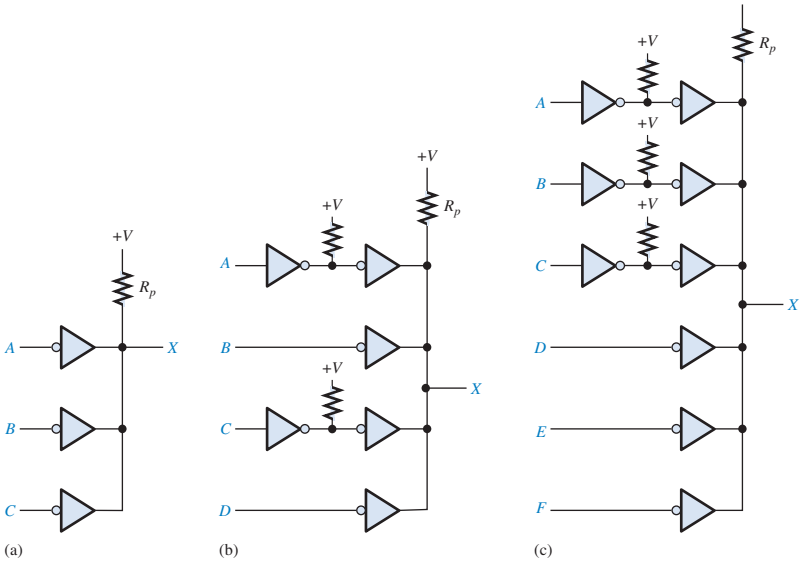
21. Veja a Figura P–96.

23. **(a)** $R_p = 198\ \Omega$
(b) $R_p = 198\ \Omega$
(c) $R_p = 198\ \Omega$

25. ALVC

27. **(a)** *A*, *B* para *X*: 9,9 ns
C, *D* para *X*: 6,6 ns
(b) *A* para X_1, X_2, X_3: 14 ns
B para X_1: 7 ns
C para X_2: 7 ns
D para X_3: 7 ns
(c) *A* para *X*: 11,1 ns
B para *X*: 11,1 ns
C para *X*: 7,4 ns
D para *X*: 7,4 ns

29. ECL opera com transistores BJTs não-saturados.



▶ **FIGURA P–96**

Índice