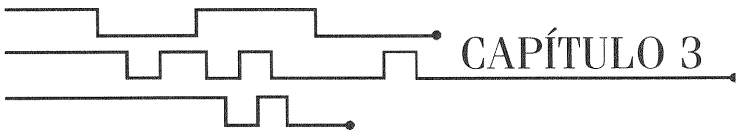
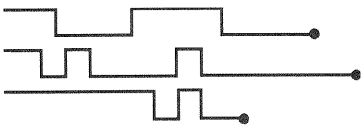


CAPÍTULO 3

# Portas Lógicas e Álgebra Booleana



## SUMÁRIO

## ■ OBJETIVOS

*Ao completar este capítulo, você deverá estar apto a:*

- Realizar as três operações lógicas básicas.
- Descrever a operação das portas AND, NAND, OR, NOR e NOT (INVERSOR), bem como construir as tabelas-verdade para as mesmas.
- Desenhar diagramas de tempo para as várias portas lógicas.
- Escrever expressões booleanas para as portas lógicas e para combinações das mesmas.
- Implementar circuitos lógicos utilizando portas básicas AND, OR e NOT.
- Estimar o potencial da álgebra booleana para simplificar circuitos lógicos complexos.
- Utilizar os teoremas de DeMorgan para simplificar expressões lógicas.
- Utilizar uma das portas lógicas universais (NAND ou NOR) para implementar um circuito representado por uma expressão booleana.
- Explicar as vantagens de construir um diagrama de circuito lógico usando a representação alternativa de símbolos para as portas lógicas *versus* a representação padrão.
- Descrever o conceito de sinais lógicos ativos em nível BAIXO e ativos em nível ALTO.
- Desenhar e interpretar circuitos lógicos que utilizam a representação de portas lógicas padrão IEEE/ANSI.

## ■ INTRODUÇÃO

Como foi mencionado no Cap. 1, circuitos digitais (lógicos) operam de modo binário onde cada tensão de saída ou entrada tem o valor 0 ou 1. As designações 0 e 1 representam intervalos de tensão predefinidos. Esta característica dos circuitos digitais nos permite utilizar a **álgebra booleana*** como uma ferramenta de análise e projeto de circuitos digitais. A álgebra booleana é uma ferramenta matemática relativamente simples que nos permite descrever a relação entre a(s) saída(s) de um circuito lógico e suas entradas através de uma equação (expressão booleana). Neste capítulo estudaremos os circuitos lógicos mais elementares, as *portas lógicas*, que são os blocos fundamentais a partir dos quais todos os outros circuitos lógicos e sistemas digitais são construídos. Veremos como a operação das diferentes portas lógicas e de circuitos mais complexos, formados pela combinação de portas lógicas, pode ser descrita e analisada utilizando a álgebra booleana. Também vislumbraremos como a álgebra booleana pode ser usada para simplificar a expressão booleana de um circuito, de modo a permitir que este circuito possa ser reconstruído utilizando um menor número de portas lógicas e/ou de conexões entre estas. Muito mais será dito sobre simplificação de circuitos no Cap. 4.

A álgebra booleana é também uma ferramenta valiosa para projetar um circuito que produzirá a relação desejada entre a entrada e a saída. Introduziremos a idéia básica neste capítulo e, depois, faremos uma cobertura mais completa deste tópico quando estudarmos o projeto de circuitos lógicos no Cap. 4.

Como a álgebra booleana expressa a operação de circuitos lógicos de forma algébrica, ela se apresenta como a forma ideal de descrever a operação de um circuito lógico para um programa de computador que necessite de informações sobre o circuito em questão. Este programa pode ser um procedimento de simplificação de circuitos, que recebe como entrada a equação em álgebra booleana, simplifica-a e fornece como saída uma versão simplificada do circuito lógico original. Uma outra aplicação possível para esse programa seria a geração de *fuse maps* (mapas de fusíveis) necessários para a programação de um dispositivo de lógica programável (*PLD — Programmable Logic Device*). O operador forneceria como entrada as equações booleanas que representariam a operação desejada do circuito e o programa as converteria em *fuse maps*. Estudaremos este processo em detalhe no Cap. 12.

Sem dúvida, a álgebra booleana é uma ferramenta muito valiosa para descrever, projetar e implementar circuitos digitais. O estudante que deseja atuar na área digital é estimulado a trabalhar bastante para compreender a lógica booleana e sentir-se à vontade com ela (acredite, ela é muito, muito mais fácil que a álgebra convencional). Faça *todos* os exemplos, exercícios e problemas, mesmo aqueles que seu professor não tiver recomendado. Quando eles acabarem, faça os seus próprios. O tempo gasto terá valido a pena quando você observar que suas habilidades aumentam e sua confiança cresce.

## 3-1 CONSTANTES E VARIÁVEIS BOOLEANAS

A álgebra booleana possui uma diferença fundamental em relação à álgebra convencional. Na álgebra booleana, constantes e variáveis possuem apenas dois valores permitidos, 0 ou 1. Uma variável booleana é uma quantidade que pode, em momentos diferentes, ser igual a 0 ou 1. Variáveis booleanas são geralmente utilizadas para representar o nível de tensão presente nas ligações ou nos terminais de entrada/saída do circuito. Por exemplo, em um certo sistema digital, o valor booleano 0 é dado para qualquer nível de tensão situado no intervalo entre 0 e 0,8 V, enquanto o valor booleano 1 é dado para qualquer nível de tensão situado no intervalo entre 2 a 5 V.*

Assim, 0 e 1 booleanos não são números de fato, mas, ao contrário, representam o estado do nível de tensão de

*A álgebra booleana foi desenvolvida pelo matemático George Boole (1815-1864) para o estudo da lógica. Ela foi apresentada em 1854 no trabalho intitulado *An Investigation of the Laws of Thought*. (*N. T.*)

*Níveis de tensão entre 0,8 e 2 V são indefinidos (não são 0 nem 1) e em circunstâncias normais não devem ocorrer.

TABELA 3-1

| Nível lógico 0 | Nível lógico 1 |
|---|---|
| Falso | Verdadeiro |
| Desligado | Ligado |
| Baixo | Alto |
| Não | Sim |
| Chave aberta | Chave fechada |

uma variável, ou, como é chamado, o seu **nível lógico.** Diz-se que o nível de tensão em um circuito digital está no nível lógico 0 ou no nível lógico 1, dependendo do seu valor numérico de fato. Em lógica digital, vários outros termos são usados como sinônimos de 0 e 1. Alguns dos mais comuns são mostrados na Tabela 3-1. Usaremos as designações 0/1 e BAIXO/ALTO na maioria das vezes.

Conforme dissemos na introdução, a álgebra booleana é um modo de expressar a relação entre as entradas e as saídas de um circuito lógico. As entradas são consideradas variáveis lógicas cujos níveis lógicos determinam, a qualquer momento, os níveis lógicos da saída. A partir de agora, utilizaremos letras para representar variáveis lógicas. Por exemplo, $A$ poderia representar uma certa entrada ou saída de um circuito digital, e em qualquer instante necessariamente teríamos ou $A = 0$ ou $A = 1$.

Como apenas dois valores são possíveis, a álgebra booleana é relativamente mais fácil de se trabalhar do que a álgebra convencional. Na álgebra booleana não existem frações, decimais, números negativos, raízes quadradas, raízes cúbicas, logaritmos, números imaginários e assim por diante. Na verdade, na álgebra booleana existem apenas *três* operações básicas: *OR* (OU), *AND* (E) e *NOT* (NÃO).

Essas operações básicas são chamadas *operações lógicas.* Circuitos digitais chamados *portas lógicas* podem ser construídos a partir de diodos, transistores e resistores conectados de um modo pelo qual a saída do circuito seja o resultado da operação lógica básica (*OR, AND, NOT*) realizada sobre suas entradas. Utilizaremos a álgebra, primeiramente para descrever e analisar essas portas lógicas básicas, e posteriormente para analisar e projetar combinações dessas portas lógicas conectadas como circuitos lógicos.

## 3-2 TABELAS-VERDADE

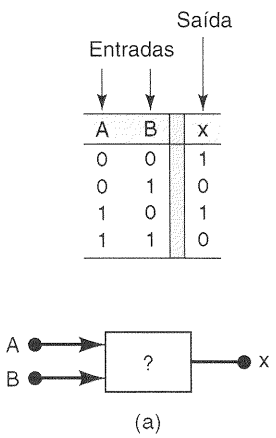
A **tabela-verdade** é uma maneira de descrever como a saída de um circuito lógico depende dos níveis lógicos presentes nas entradas do circuito. A Fig. 3-1(a) mostra a tabela-verdade para um tipo de circuito lógico de duas entradas. A tabela relaciona todas as combinações possíveis dos níveis lógicos presentes nas entradas $A$ e $B$ com o nível correspondente da saída $x$. A primeira linha da tabela mostra que quando $A$ e $B$ estão ambos em nível 0, a saída $x$ está no nível 1, ou, de modo equivalente, no estado 1. A segunda linha da tabela mostra que quando a entrada $B$ muda para o estado 1, de modo que $A = 0$ e $B = 1$, a saída $x$ torna-se 0. De maneira similar, a tabela mostra o que acontece com o estado da saída para qualquer conjunto de condições de entrada.

As Figs. 3-1(b) e (c) mostram exemplos de tabelas-verdade para circuitos de três e de quatro entradas. Novamente, cada tabela enumera todas as combinações possíveis dos níveis lógicos de entrada na esquerda, juntamente com o nível lógico resultante para a saída $x$ na direita. É claro que o valor real de $x$ dependerá do tipo de circuito lógico utilizado.

Observe que existem 4 linhas para um tabela-verdade de duas entradas, 8 linhas para uma tabela-verdade de três entradas, e 16 linhas para uma tabela de quatro entradas. O número de combinações de entrada será igual a $2^N$ para uma tabela-verdade de $N$ entradas. Note também que a lista de todas as combinações possíveis de entrada acompanha a seqüência de contagem binária, e, assim, torna-se bastante simples escrever todas as combinações possíveis sem esquecer nenhuma.

**Questões de Revisão**

1. Qual é o estado da saída para o circuito de quatros entradas representado na Fig. 3-1(c), quando todas as entradas forem iguais a 1?
2. Repita a questão 1 para as seguintes condições de entrada: $A = 1$, $B = 0$, $C = 1$, $D = 0$.
3. Quantas linhas deve ter uma tabela para representar um circuito de cinco entradas?




| A | B | x |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(a)

| A | B | C | x |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(b)

| A | B | C | D | x |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 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 |

(c)

**Fig. 3-1** Exemplos de tabelas-verdade para circuitos (a) de duas entradas, (b) de três entradas e (c) de quatro entradas.

# 3-3 OPERAÇÃO OR COM PORTAS OR

A **operação OR** é a primeira das três operações booleanas básicas a ser estudada. A tabela-verdade na Fig. 3-2(a) mostra o que acontece quando duas entradas lógicas, $A$ e $B$, são combinadas através da operação OR para produzir a saída $x$. A tabela mostra que $x$ é igual a 1 para todas as combinações dos níveis de entrada onde uma *ou* mais entradas são iguais a 1. O único caso onde $x$ é igual a 0 ocorre quando todas as entradas são iguais a 0.

A expressão booleana para a operação OR é dada por:

$$x = A + B$$

Nesta expressão, o sinal de + não representa a operação de adição ordinária, mas representa a operação OR. A operação OR é semelhante à adição ordinária, exceto para o caso em que $A$ e $B$ são ambos iguais a 1. Neste caso, a operação OR produz 1 + 1 = 1, e não 1 + 1 = 2, como seria no caso de uma adição. Na álgebra booleana, 1 é o valor máximo que pode ser obtido, e assim nunca poderemos ter um resultado maior do que 1. Essa afirmação continua sendo verdadeira quando combinamos três entradas utilizando a operação OR. Aqui teremos $x = A + B + C$. Se considerarmos o caso em que todas as três entradas são iguais a 1:

$$x = 1 + 1 + 1 = 1$$

Novamente, o resultado da operação OR, quando mais de uma entrada é igual a 1, é *sempre* igual a 1.

A expressão lógica $x = A + B$ é lida como "$x$ é igual a $A$ OR $B$". O mais importante a ser lembrado é que o sinal de +, que aparece na expressão, representa a operação OR que foi definida através da tabela-verdade na Fig. 3-2(a), e não a operação de adição ordinária.

## Porta OR

Em circuitos digitais, uma **porta OR*** é um circuito que possui duas ou mais entradas e cuja saída é igual à combinação das entradas através da operação OR. A Fig. 3-2(b) mostra o símbolo para uma porta OR de duas entradas. As entradas $A$ e $B$ são níveis lógicos de tensão, e a saída $x$ é um nível lógico de tensão cujo valor é o resultado da operação OR sobre as entradas $A$ e $B$, isto é, $x = A + B$. Em outras palavras, a porta OR funciona de tal modo que sua saída será ALTA (nível lógico 1) se $A$ *ou* $B$ *ou ambas* forem iguais a 1. A saída da porta OR será BAIXA (nível lógico 0) apenas se todas as entradas forem iguais a 0.

Esta mesma idéia pode ser estendida para um maior número de entradas. A Fig. 3-3 mostra uma porta OR de 3 entradas e sua tabela-verdade. O exame desta tabela-verdade mostra novamente que a saída será igual a 1 para todos os casos nos quais uma ou mais entradas são iguais a 1. Este princípio geral é o mesmo para portas OR com qualquer número de entradas.

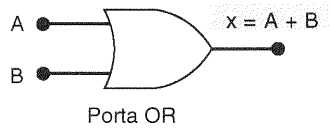
Usando a linguagem da álgebra booleana, a saída $x$ pode ser expressa como $x = A + B + C$, onde novamente devemos enfatizar que o sinal de + representa a operação OR. A saída de qualquer porta OR, então, pode ser expressa pela combinação das entradas através da operação OR. Utilizaremos este fato quando estivermos analisando circuitos lógicos.

*O termo porta [do inglês, *gate*] deriva da operação de habilitar/inibir a ser discutida no Cap. 4.

OR

| A | B | x = A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(a)




Porta OR

(b)

**Fig. 3-2** (a) Tabela-verdade que define a operação OR; (b) símbolo para uma porta OR de duas entradas.

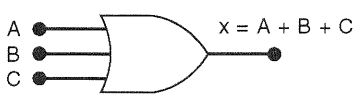
## Resumo da Operação OR

Os pontos mais importantes a serem lembrados no que se refere à operação OR e às portas OR são:

1. A operação OR produz 1 como resultado, quando *qualquer* uma das variáveis for igual a 1.
2. A operação OR produz 0 como resultado, quando todas as variáveis forem iguais a 0.
3. Na operação OR, 1 + 1 = 1, 1 + 1 + 1 = 1, e assim por diante.
4. A porta OR é um circuito lógico que realiza a operação OR sobre as entradas lógicas do circuito.

### EXEMPLO 3-1

Em muitos sistemas de controle industriais é necessário ativar uma função de saída sempre que uma das várias entradas for ativada. Por exemplo, em um processo químico, pode ser desejável que um alarme seja ativado toda vez que a temperatura do processo exceder um valor máximo *ou* sempre que a pressão estiver acima de um certo limite. A Fig. 3-4 mostra um diagrama de blocos desta situação. O circuito transdutor de temperatura produz uma tensão proporcional à temperatura do processo. Esta tensão, $V_T$, é comparada com uma tensão de referência de temperatura, $V_{TR}$, através de um circuito comparador. A saída do comparador está normalmente




| A | B | C | x = A + B + C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Fig. 3-3** Símbolo e a tabela-verdade para uma porta OR de três entradas.

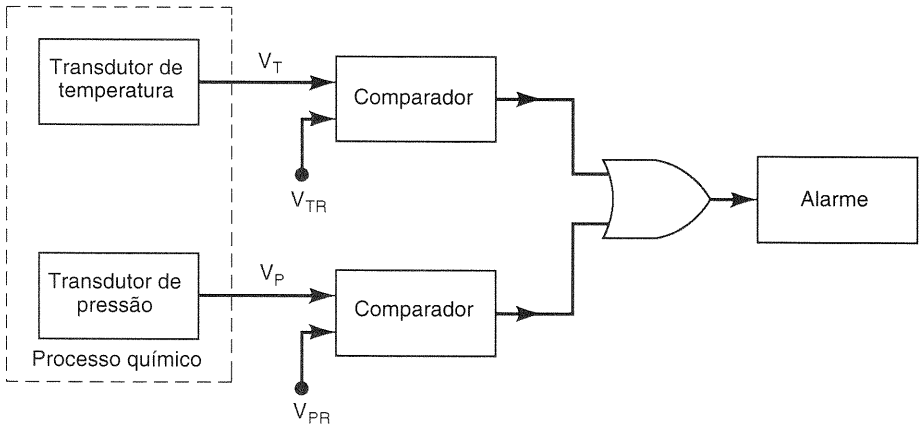



**Fig. 3-4** Exemplo de utilização da porta OR em um sistema de alarme.

com uma tensão baixa (nível lógico 0), mas esta muda para uma tensão alta (nível lógico 1) quando $V_T$ excede $V_{TR}$, indicando que a temperatura do processo é excessiva. Um arranjo similar é feito para a medição da pressão, de modo que a saída do respectivo comparador passa do nível baixo para o alto quando a pressão for excessiva.

Uma vez que desejamos que o alarme seja ativado quando *ou* a temperatura *ou* a pressão seja muito alta, podemos conectar as saídas dos comparadores a uma porta OR de duas entradas. A saída da porta OR será ALTA (1) para qualquer uma das condições de alarme, fazendo com que o mesmo seja ativado. Esta mesma idéia pode ser obviamente estendida para situações com mais do que duas variáveis de processo.

## EXEMPLO 3-2

Determine a saída da porta OR mostrada na Fig. 3-5. As entradas da porta OR são $A$ e $B$ que variam segundo o diagrama de tempo apresentado. Por exemplo, $A$ começa em BAIXO em $t_0$, passa para ALTO em $t_1$, e retorna a BAIXO em $t_3$, e assim por diante.

**Solução**

A saída da porta OR pode ser determinada observando-se que ela estará em ALTO sempre que *qualquer* uma das entradas estiver em nível alto. Quando $A$ passa para ALTO em $t_1$, SAÍDA passará para ALTO. A SAÍDA permanecerá em ALTO até $t_4$, quando ambas as entradas estarão em BAIXO. Observe que as mudanças nos níveis lógicos das entradas que ocorrem em $t_2$ e $t_3$ não têm efeito na SAÍDA, uma vez que uma das entradas permanece em nível ALTO enquanto a outra está mudando. Enquanto uma das entradas da porta OR estiver em ALTO, a saída permanecerá em ALTO, não importando o que estiver acontecendo nas outras entradas. Este mesmo raciocínio pode ser usado para determinar o restante do diagrama de tempo para SAÍDA.

## EXEMPLO 3-3A

Para o exemplo mostrado na Fig. 3-6, determine a forma de onda na saída da porta OR.

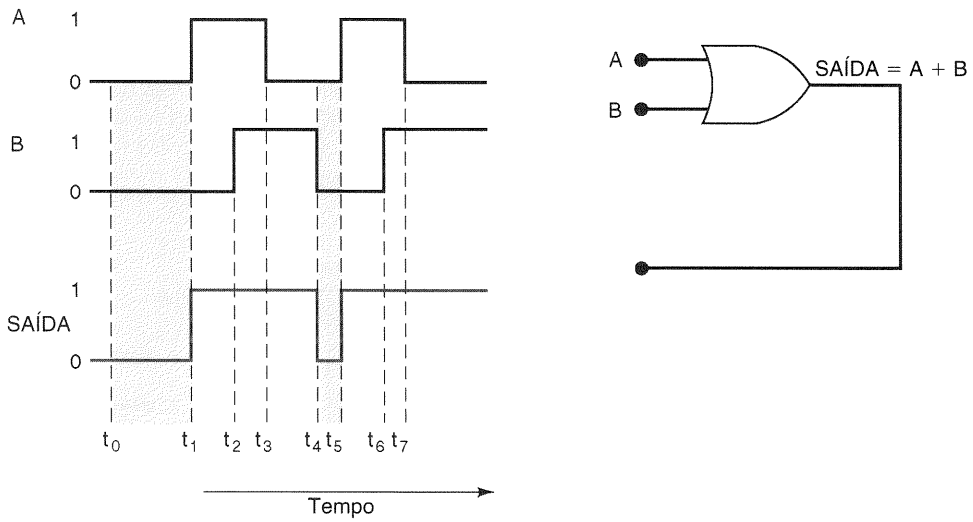



**Fig. 3-5** Exemplo 3-2.

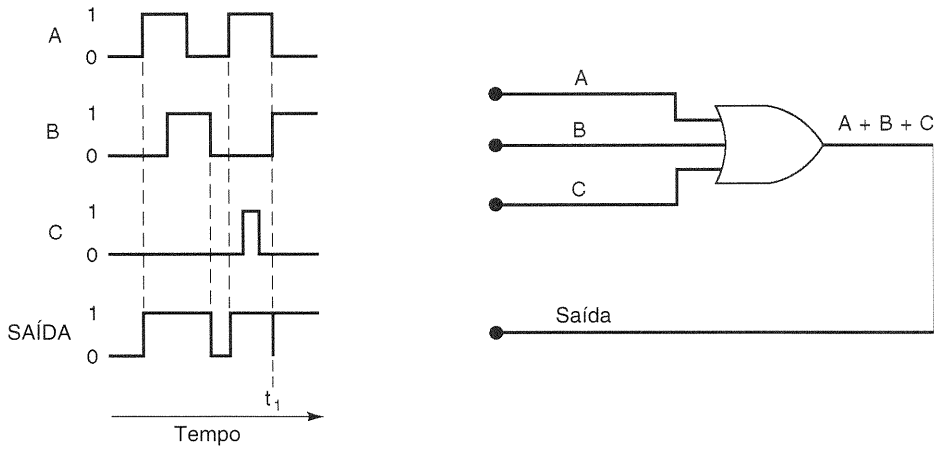



**Fig. 3-6** Exemplos 3-3A e 3-3B.

**Solução**

As três entradas da porta OR, *A*, *B* e *C*, estão variando, conforme as formas de onda mostradas no diagrama. A saída da porta OR será determinada observando que esta será alta sempre que *qualquer* uma das três entradas estiver em nível alto. Usando este raciocínio, a forma de onda da saída da porta OR é apresentada na figura. Devemos prestar bastante atenção no que acontece no instante $t_1$. O diagrama mostra que neste instante de tempo a entrada *A* está passando de alto para baixo, enquanto a entrada *B* está passando de baixo para alto. Como estas entradas estão fazendo suas transições aproximadamente no mesmo instante, e como essas transições duram um certo tempo, existe um pequeno intervalo em que ambas as entradas dessa porta OR estão na faixa indefinida entre 0 e 1. Quando isso ocorre, a saída da porta OR também possui um valor situado nesse intervalo indefinido, caracterizado por um pulso espúrio e estreito (*glitch* ou *spike*) na forma de onda da saída em $t_1$. A ocorrência do *glitch*, sua amplitude e largura irão depender da velocidade com que as transições acontecem.

**EXEMPLO 3-3B**

O que aconteceria ao *glitch* mostrado na Fig. 3-6 caso a entrada *C* permanecesse em nível ALTO enquanto *A* e *B* estivessem mudando de estado em $t_1$?

**Solução**

Com a entrada *C* em ALTO no instante $t_1$, a saída da porta OR permanecerá em ALTO independentemente do que estiver ocorrendo nas outras entradas, porque se qualquer uma das entradas estiver em ALTO a saída permanecerá em ALTO, e portanto o *glitch* não aparecerá na saída.

**Questões de Revisão**

1. Qual é a única combinação de valores das entradas que produz um nível BAIXO na saída de qualquer porta OR?
2. Escreva a expressão booleana para uma porta OR de seis entradas.
3. Se a entrada *A* mostrada na Fig. 3-6 fosse mantida permanentemente em nível 1, qual seria a forma de onda resultante na saída?

## 3-4 OPERAÇÃO AND COM PORTAS AND

A **operação AND** é a segunda operação booleana básica. A tabela-verdade que aparece na Fig. 3-7(a) mostra o que acontece quando duas entradas lógicas, *A* e *B*, são combinadas usando a operação AND para produzir a saída *x*. A tabela mostra que *x* está em nível lógico 1 somente quando tanto *A* como *B* estão em nível lógico 1. Para qualquer outro caso, onde uma das entradas é 0, a saída é 0.

A expressão booleana para a operação AND é

$$x = A \cdot B$$

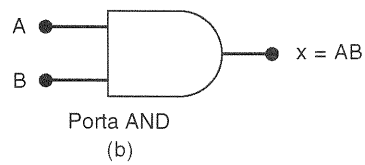
Nesta expressão, o sinal · expressa a operação AND, e não a multiplicação ordinária. Entretanto, a operação AND sobre variáveis booleanas opera da mesma maneira que a multiplicação ordinária, como pode ser visto através de um exame da tabela-verdade. Assim, podemos pensar nas duas operações como se fossem apenas uma. Essa característica pode ser de grande ajuda na análise de expressões lógicas que contenham operações AND.

A expressão $x = A \cdot B$ é lida como "$x = A$ AND $B$". O sinal · é geralmente omitido de modo que a expressão se torna apenas $x = AB$. A coisa mais importante a ser lembrada é que a operação AND produzirá 1 como resultado *ape-*

AND

| A | B | x = A · B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(a)




(b)

**Fig. 3-7** (a) Tabela-verdade para a operação AND; (b) símbolo da porta AND.

| A | B | C | x = ABC |
|---|---|---|---|
| 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 |

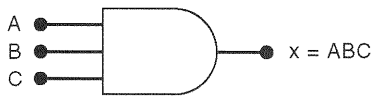



**Fig. 3-8.** Tabela-verdade e o símbolo para uma porta AND de três entradas.

*nas quando* todas as entradas (variáveis) forem iguais a 1, exatamente como na multiplicação. Este fato permanece verdadeiro para o caso de termos mais de duas entradas. Por exemplo, quando a operação AND é realizada sobre três entradas, temos $x = A \cdot B \cdot C = ABC$. O único momento em que $x$ pode ser igual a 1 é quando $A = B = C = 1$.

## Porta AND

O símbolo lógico para uma **porta AND** de duas entradas pode ser visto na Fig. 3-7(b). A saída da porta AND é igual ao produto das entradas lógicas, isto é, $x = AB$. Em outras palavras, a porta AND é um circuito que opera de tal maneira que sua saída está em ALTO apenas quando todas as entradas estão em ALTO. Para todos os outros casos, a saída da porta estará em BAIXO.

Esse mesmo modo de operação é característico em portas AND com mais de duas entradas. Por exemplo, uma porta AND de três entradas e a tabela-verdade correspondente podem ser vistas na Fig. 3-8. Mais uma vez, observe que a saída da porta é 1 apenas para o caso em que $A = B = C = 1$. A expressão para a saída é $x = ABC$. Para o caso de uma porta AND de quatro entradas, a expressão é $x = ABCD$, e assim por diante.

Observe a diferença entre os símbolos das portas AND e OR. Sempre que você vir o símbolo de uma porta AND em um diagrama de circuitos lógicos, isto lhe diz que a saída estará em ALTO *apenas* quando *todas* as entradas estiverem em ALTO. Sempre que você vir o símbolo de uma porta OR, isto significa que a saída estará em ALTO quando *qualquer* uma das entradas estiver em ALTO.

## Resumo da Operação AND

1. A operação AND é realizada exatamente do mesmo modo que a multiplicação ordinária de 0s e 1s.
2. A saída é igual a 1 quando todas as entradas forem iguais a 1.
3. A saida é 0 para o caso em que uma ou mais entradas são iguais a 0.
4. Uma porta AND é um circuito lógico que realiza a operação AND nas entradas do circuito.

### EXEMPLO 3-4

Determine a forma de onda da saída $x$ da porta AND mostrada na Fig. 3-9, dadas as formas de onda das entradas.

**Solução**

A saída da porta AND é determinada observando que ela estará em ALTO apenas quando todas as entradas estiverem em ALTO ao mesmo tempo. Para as formas de onda fornecidas, isto acontece apenas durante os intervalos $t_2$-$t_3$ e $t_6$-$t_7$. Em todos os outros intervalos, uma ou mais entradas estão em 0, produzindo portanto um nível BAIXO na saída. Observe que mudanças nos níveis de entrada que ocorrem enquanto uma das entradas está em nível BAIXO não têm efeito na saída.

### EXEMPLO 3-5A

Determine a forma de onda da saída para a porta AND mostrada na Fig. 3-10.

**Solução**

A saída $x$ será igual a 1 apenas quando $A$ e $B$ estiverem em ALTO ao mesmo tempo. A partir deste fato, podemos determinar a forma de onda de $x$ como está mostrado na figura.

Observe que a forma de onda de $x$ é igual a 0 toda vez que $B$ é igual a 0, independentemente do que acontece com

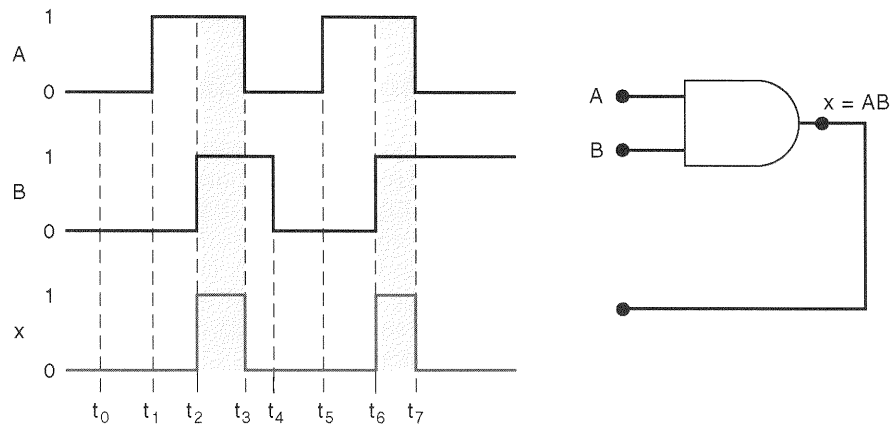



**Fig. 3-9** Exemplo 3-4.

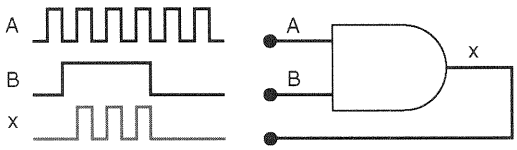



**Fig. 3-10** Exemplos 3-5A e 3-5B.

a entrada $A$. Também é importante notar que sempre que $B$ é igual a 1 a forma de onda de $x$ é igual à de $A$. Então podemos pensar na entrada $B$ como uma entrada de *controle*, cujo nível lógico determina se a forma de onda de $A$ chega ou não na saída $x$. Nesta situação, a porta AND é usada como um *circuito inibidor*. Podemos dizer que $B = 0$ é a condição de inibição que força que a saída seja igual a 0. Ao contrário, $B = 1$ é a condição de *habilitação*, que permite que $A$ chegue até a saída. Esta operação inibidora é uma importante aplicação das portas AND que encontraremos mais tarde.

**EXEMPLO 3-5B**

O que acontecerá com a forma de onda da saída $x$ na Fig. 3-10 se a entrada $B$ permanecer em nível 0?

**Solução**

Enquanto $B$ for mantido em BAIXO, a saída $x$ também permanecerá em BAIXO. Podemos chegar a esta conclusão de dois modos: o primeiro seria observar que com $B = 0$ temos $x = A \cdot B = A \cdot 0 = 0$, uma vez que o resultado da operação AND (multiplicação), quando uma das entradas é 0, é sempre 0. O segundo modo seria observar que uma porta AND necessita que todas as suas entradas estejam em ALTO para que a saída seja ALTO, e isto não acontece quando $B$ é mantido em BAIXO.

**Questões de Revisão**

1. Qual é a única combinação de entrada que irá produzir um nível ALTO na saída de uma porta AND de cinco entradas?
2. Qual é o nível lógico que deve ser aplicado na segunda entrada de uma porta AND de duas entradas para que o sinal aplicado na primeira entrada seja inibido (impedido) de chegar na saída?
3. *Falso* ou *verdadeiro*: A saída de uma porta AND sempre difere da saída de uma porta OR para as mesmas condições de entrada.

## 3-5 OPERAÇÃO NOT

A **operação NOT** é realizada, ao contrário das operações AND e OR, sobre uma única entrada. Por exemplo, se a

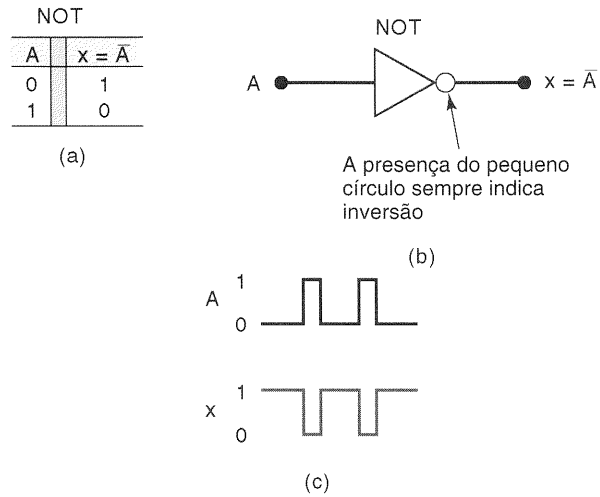



**Fig. 3-11** (a) Tabela-verdade; (b) símbolo para o INVERSOR (NOT); (c) formas de onda.

variável $A$ é sujeita à operação NOT, o resultado $x$ pode ser expresso como:

$$x = \overline{A}$$

onde a barra sobreposta representa a operação NOT. Esta expressão é lida como "$x$ é igual a NOT $A$" ou "$x$ é igual ao *inverso* de $A$" ou "$x$ é igual ao *complemento* de $A$".* Cada uma destas expressões é de uso comum, e todas indicam que o nível lógico de $x = \overline{A}$ é *oposto* ao valor lógico de $A$. A tabela-verdade mostrada na Fig. 3-11(a) esclarece esta afirmação para os dois casos possíveis, $A = 0$ e $A = 1$, isto é:

$$\overline{1} = 0 \quad \text{porque NOT 1 é 0}$$

e

$$\overline{0} = 1 \quad \text{porque NOT 0 é 1}$$

A operação NOT é também chamada de **inversão** ou **complemento;** estes termos serão usados de modo intercambiável durante o restante do livro. Apesar de sempre utilizarmos a barra sobreposta para representar inversão, é importante mencionar que um outro símbolo para representar a inversão é o apóstrofo ('), isto é:

$$A' = \overline{A}$$

Ambos os símbolos são reconhecidos como indicadores da operação de inversão.

### Circuito NOT (INVERSOR)

A Fig. 3-11(b) mostra o símbolo para a representação do **circuito NOT**, que é mais comumente chamado de **INVERSOR.** Este circuito tem *sempre* uma única entrada, e o nível lógico de sua saída é sempre oposto ao nível lógico da entrada. A Fig. 3-11(c) mostra como o INVERSOR age sobre o sinal de entrada. Ele inverte (complementa) o sinal de entrada em todos os pontos da forma de onda da entrada.

*Usa-se também dizer "$x$ é igual a $A$ barrado". (*N. T.*)

## Resumo das Operações Booleanas

As regras para as operações AND, OR e NOT podem ser resumidas como segue:

| *OR* | *AND* | *NOT* |
|---|---|---|
| $0 + 0 = 0$ | $0 \cdot 0 = 0$ | $\overline{0} = 1$ |
| $0 + 1 = 1$ | $0 \cdot 1 = 0$ | $\overline{1} = 0$ |
| $1 + 0 = 1$ | $1 \cdot 0 = 0$ | |
| $1 + 1 = 1$ | $1 \cdot 1 = 1$ | |

**Questões de Revisão**

1. A saída do INVERSOR da Fig. 3-11 é conectada à entrada de um segundo INVERSOR. Determine o nível lógico da saída do segundo INVERSOR para cada nível lógico da entrada $A$.
2. A saída da porta AND da Fig. 3-7 é conectada à entrada de um INVERSOR. Escreva a tabela-verdade que relaciona a saída $y$ do INVERSOR com cada combinação das entradas $A$ e $B$.

# 3-6 DESCREVENDO CIRCUITOS LÓGICOS ALGEBRICAMENTE

Qualquer circuito lógico, independentemente de sua complexidade, pode ser completamente descrito usando as operações booleanas previamente definidas, porque as portas AND, OR e NOT são os blocos básicos para a construção de sistemas digitais. Por exemplo, considere o circuito da Fig. 3-12. O circuito possui 3 entradas, $A$, $B$ e $C$, e uma única saída, $x$. Utilizando as expressões booleanas para cada porta, podemos facilmente determinar a expressão para a saída.

A expressão para a saída da porta AND é escrita como $A \cdot B$. Esta saída é conectada a uma porta OR, juntamente com $C$, que é a outra entrada do circuito. A porta OR opera sobre as entradas de modo que a saída seja o resultado de uma operação OR sobre as entradas. Assim, podemos expressar a saída da porta OR como $x = A \cdot B + C$ (esta última expressão também poderia ter sido escrita como $x = C + A \cdot B$, uma vez que a ordem dos termos não importa na operação OR).

Ocasionalmente, pode haver dúvida em relação a qual operação deve ser realizada primeiro. A expressão $A \cdot B + C$ pode ser interpretada de dois modos: (1) é feita a operação $A \cdot B$ OR $C$, ou (2) é feita a operação $A$ AND $B + C$. Para evitar essa confusão, fica definido que, caso uma expressão possua as operações AND e OR, as operações AND são realizadas primeiro, a não ser que existam *parênteses* na expressão, neste caso, a operação dentro dos parênteses é realizada primeiro. Esta é a mesma regra usada na álgebra comum para determinar a ordem das operações.

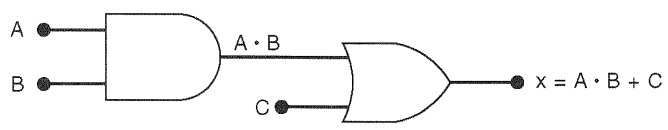



**Fig 3-12** Circuito lógico com sua expressão booleana.

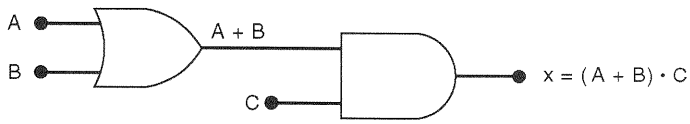



**Fig. 3-13** Circuito lógico cuja expressão requer parênteses.

A fim de dar mais um exemplo, considere o circuito da Fig. 3-13. A expressão para a saída da porta OR é simplesmente $A + B$. Esta saída serve como entrada de uma porta AND juntamente com uma outra entrada $C$. Portanto, podemos expressar a saída da porta AND como $x = (A + B) \cdot C$. Observe o uso de parênteses para indicar que $A$ OR $B$ é realizada primeiro, isto é, antes que se faça um AND desta soma OR com $C$. Sem os parênteses, poderíamos interpretar a expressão de forma *incorreta*, uma vez que $A + B \cdot C$ significa $A$ OR com o produto $B \cdot C$.

## Circuitos Contendo INVERSORes

Sempre que um INVERSOR é apresentado em um diagrama de circuitos lógicos, a expressão para a sua saída é simplesmente igual à expressão da entrada com um barra sobre ela. A Fig. 3-14 mostra dois exemplos usando INVERSORes. Na Fig. 3-14(a), a entrada é conectada a um inversor, e a saída do mesmo é igual a $\overline{A}$. A saída do INVERSOR é conectada a uma porta OR juntamente com $B$, de modo que a saída da porta OR é igual a $\overline{A} + B$. Observe que a barra está apenas sobre o $A$, indicando que $A$ é primeiramente invertido e depois é feita uma operação OR com $B$.

Na Fig. 3-14(b), a saída da porta OR é igual a $A + B$, e esta é conectada a um INVERSOR. A saída do INVERSOR é portanto igual a $\overline{(A + B)}$, uma vez que ele inverte a expressão de entrada *completa*. Observe que a barra cobre a expressão $(A + B)$ inteira. Isto é importante porque, como será mostrado mais adiante, as expressões $\overline{(A + B)}$ e $(\overline{A} + \overline{B})$ *não* são equivalentes. A expressão $\overline{(A + B)}$ significa que realizamos a operação $A$ OR $B$ e que depois o resultado desta operação é invertido, enquanto a expressão $(\overline{A} + \overline{B})$ indica que $A$ é invertido, $B$ é invertido e somente depois é feita uma operação OR com estes resultados.

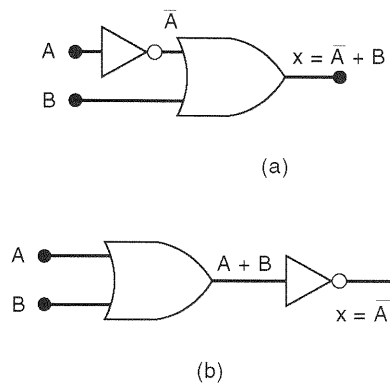



**Fig. 3-14** Circuitos que usam INVERSORes.

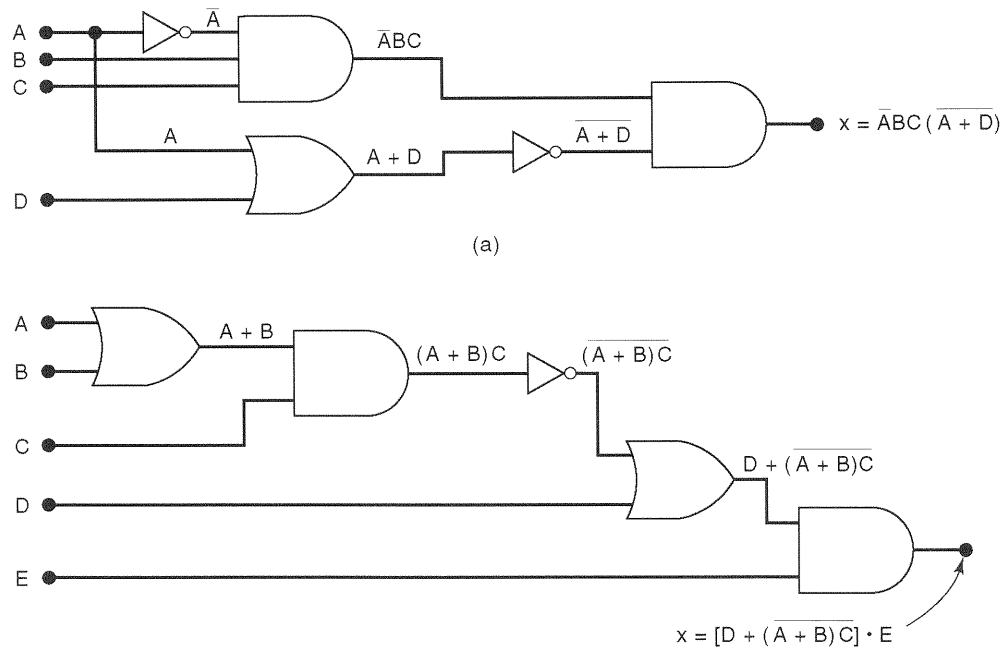



**Fig. 3-15** Mais exemplos.

A Fig. 3-15 mostra mais dois exemplos que devem ser estudados com cuidado. Observe especialmente o uso de *dois* conjuntos separados de parênteses na Fig. 3-15(b). Observe também que na Fig. 3-15(a) a variável de entrada $A$ está conectada como entrada em duas portas diferentes.

**Questões de Revisão**

**1.** Na Fig. 3-15(a), troque cada uma das portas AND por uma porta OR e troque a porta OR por uma porta AND. Agora escreva a expressão para a saída $x$.

## 3-7 DETERMINANDO O VALOR DA SAÍDA DE CIRCUITOS LÓGICOS

Uma vez obtida a expressão booleana para a saída do circuito, o nível lógico da saída pode ser determinado para qualquer conjunto de níveis lógicos das entradas. Por exemplo, suponha que desejamos saber o nível lógico da saída $x$ para o circuito mostrado na Fig. 3-15(a), para o caso em que $A = 0$, $B = 1$, $C = 1$ e $D = 1$. Como na álgebra ordinária, o valor de $x$ pode ser encontrado substituindo-se os valores das variáveis na expressão e fazendo as operações como se segue:

$$\begin{aligned} x &= \bar{A}BC(\overline{A + D}) \\ &= \bar{0} \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\ &= 1 \cdot 1 \cdot 1 \cdot (\overline{0 + 1}) \\ &= 1 \cdot 1 \cdot 1 \cdot (\bar{1}) \\ &= 1 \cdot 1 \cdot 1 \cdot 0 \\ &= 0 \end{aligned}$$

Como um outro exemplo, vamos avaliar a expressão para a saída do circuito da Fig. 3-15(b), para o caso em que $A = 0$, $B = 0$, $C = 1$, $D = 1$ e $E = 1$.

$$\begin{aligned} x &= [D + \overline{(A + B)C}] \cdot E \\ &= [1 + \overline{(0 + 0) \cdot 1}] \cdot 1 \\ &= [1 + \overline{0 \cdot 1}] \cdot 1 \\ &= [1 + \bar{0}] \cdot 1 \\ &= [1 + 1] \cdot 1 \\ &= 1 \cdot 1 \\ &= 1 \end{aligned}$$

De um modo geral, as seguintes regras devem ser obedecidas quando avaliamos expressões booleanas:

**1.** Primeiro, faça todas as inversões de termos simples, isto é, $\bar{0} = 1$ ou $\bar{1} = 0$.
**2.** A seguir, faça todas as operações que estão dentro dos parênteses.
**3.** Faça a operação AND antes da operação OR, a não ser que os parênteses indiquem o contrário.
**4.** Se a expressão tiver uma barra sobreposta, faça as operações da expressão primeiro e depois inverta o resultado.

Para praticar, determine os níveis lógicos das saídas dos circuitos da Fig. 3-15 para o caso em que todas as entradas são iguais a 1. As respostas são $x = 0$ e $x = 1$, respectivamente.

### Determinando o Nível da Saída a Partir de um Diagrama

O nível lógico da saída para um dado conjunto de níveis lógicos das entradas também pode ser determinado diretamente do diagrama do circuito, *sem* utilizar a expressão booleana. Esta técnica é freqüentemente usada por técnicos durante testes ou reparos de circuitos digitais, uma vez que ela mostra qual deveria ser a saída de cada porta, bem como qual deveria ser a saída final do sistema. Por exemplo, o circuito da Fig. 3-15(a) foi redesenhado na Fig. 3-16 com níveis de entrada iguais a $A = 0$, $B = 1$, $C = 1$, $D = 1$. O procedimento é o seguinte: a partir das entradas, devemos determinar

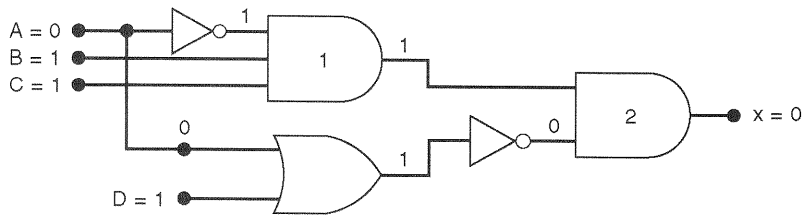



**Fig. 3-16** Determinando o nível lógico de saída a partir do diagrama do circuito.

para cada INVERSOR, ou porta, o valor de sua saída até que o valor da saída final do sistema seja encontrado.

Na Fig. 3-16, a porta AND número 1 tem *todas* as suas entradas em nível 1 porque o INVERSOR troca $A = 0$ para 1. Esta condição produz um nível lógico 1 na saída da porta AND, uma vez que $1 \cdot 1 \cdot 1 = 1$. A porta OR tem como entradas os níveis 0 e 1, o que produz um nível 1 na saída, uma vez que $1 + 0 = 1$. Este nível 1 é invertido para nível 0, e este, por sua vez, é aplicado como entrada da porta AND número 2, juntamente com a saída da porta AND número 1. Os níveis 0 e 1 nas entradas da porta AND número 2 vão gerar na saída um nível lógico 0 porque $0 \cdot 1 = 0$.

**EXEMPLO 3-6**

Determine a saída do circuito da Fig 3-16 para o caso em que todas as entradas estão em BAIXO.

**Solução**

Com $A = B = C = D = 0$, a saída da porta AND 1 estará no nível BAIXO. Este é colocado na entrada da porta AND 2, o que automaticamente gera um nível BAIXO na saída, independentemente dos níveis lógicos em outros pontos do circuito. Este exemplo mostra que nem sempre é necessário determinar os níveis lógicos em todos os pontos do circuito para determinar o nível lógico de sua saída.

**Questões de Revisão**

1. Use a expressão para $x$ para determinar a saída do circuito da Fig. 3-15(a), para as seguintes condições de entrada: $A = 0$, $B = 1$, $C = 1$ e $D = 0$.
2. Use a expressão para $x$ para determinar a saída do circuito da Fig. 3-15(b), para as seguintes condições de entrada: $A = B = E = 1$ e $C = D = 0$.
3. Determine as respostas das questões 1 e 2, encontrando os níveis lógicos presentes em cada entrada e saída das portas lógicas, como foi feito na Fig. 3-16.

## 3-8 IMPLEMENTANDO CIRCUITOS A PARTIR DE EXPRESSÕES BOOLEANAS

Se a operação de um circuito lógico é definida por meio de uma expressão booleana, então o diagrama do circuito lógico pode ser implementado diretamente desta expressão. Por exemplo, se necessitamos de um circuito que é defini-

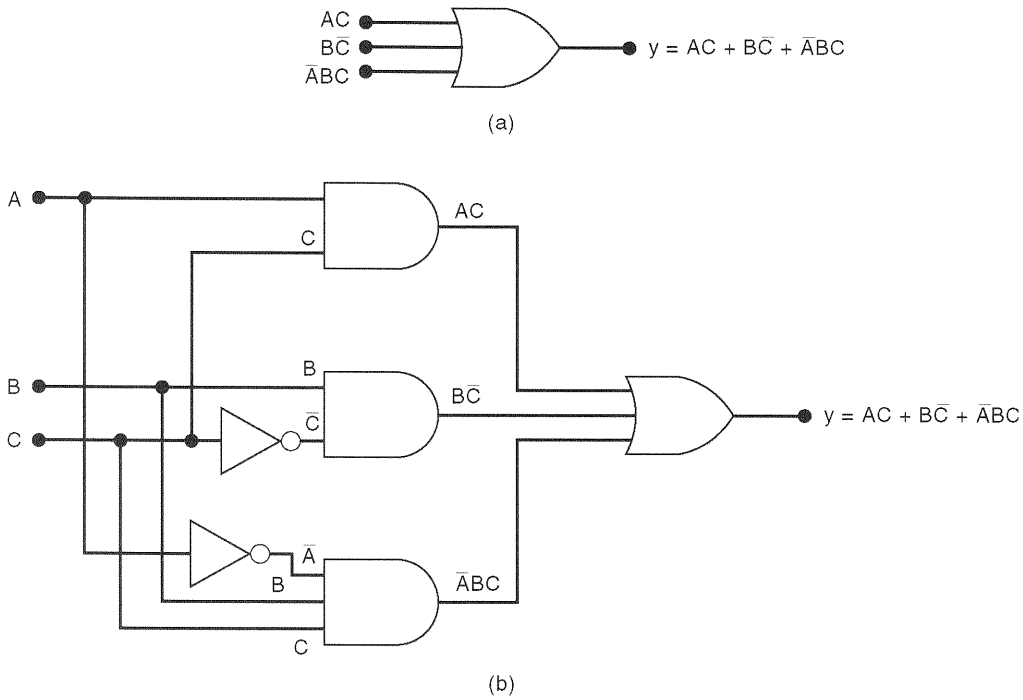



Fig. 3-17 Construindo um circuito lógico a partir de uma expressão booleana.

do pela expressão $x = A \cdot B \cdot C$, percebemos imediatamente que tudo de que precisamos é uma porta AND de três entradas. Se precisamos de um circuito definido pela expressão $x = A + \overline{B}$, poderíamos usar uma porta OR de duas entradas com um INVERSOR em uma de suas entradas. Esse mesmo raciocínio usado para esses casos simples pode ser estendido para circuitos mais complexos.

Suponha que desejamos implementar um circuito cuja saída pode ser definida pela expressão $y = AC + B\overline{C} + \overline{A}BC$. Esta expressão booleana possui três termos ($AC$, $B\overline{C}$, $\overline{A}BC$) sobre os quais é feita uma operação OR. Isto nos diz que necessitamos de uma porta OR de três entradas que são iguais a $AC$, $B\overline{C}$ e $\overline{A}BC$, respectivamente. Isto é mostrado na Fig. 3-17(a), onde uma porta OR de três entradas está desenhada com suas entradas $AC$, $B\overline{C}$ e $\overline{A}BC$.

Cada entrada da porta OR é um termo que expressa uma operação AND, o que significa que portas AND com entradas apropriadas devem ser usadas para gerar cada um desses termos. Isto é mostrado na Fig. 3-17(b) que é o diagrama do circuito final. Observe o uso de INVERSORes para produzir os termos $\overline{A}$ e $\overline{C}$ necessários à expressão.

Essa abordagem é bastante geral e pode ser sempre seguida, embora vamos ver mais tarde que existem outras técnicas melhores e mais eficientes que podem ser empregadas. Por enquanto, esse método direto de implementar circuitos lógicos deve ser utilizado para diminuir o número de coisas novas que devem ser aprendidas.

**EXEMPLO 3-7**

Desenhe o circuito que implementa a expressão $x = AB + \overline{B}C$.

**Solução**

Esta expressão indica que os termos $AB$ e $\overline{B}C$ são entradas de uma porta OR, e cada um destes termos pode ser gerado por uma porta AND. O resultado é mostrado na Fig. 3-18.

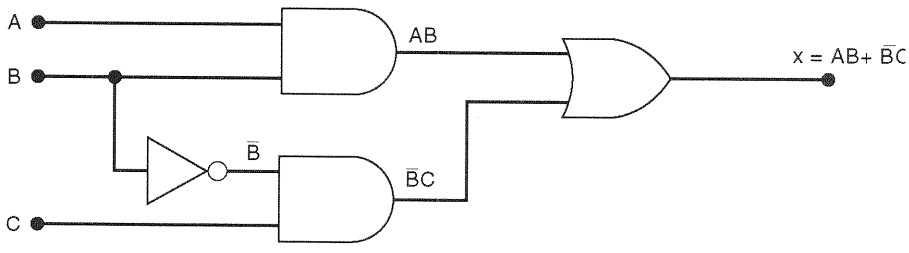



**Fig. 3-18** Exemplo 3-7.

**Questões de Revisão**

1. Desenhe o circuito que implementa a expressão $x = \overline{A}BC(\overline{A + D})$, usando portas lógicas com no máximo três entradas.
2. Desenhe o circuito para a expressão $y = AC + B\overline{C} + \overline{A}BC$.
3. Desenhe o circuito para $x = [D + (\overline{A + B})C] \cdot E$.

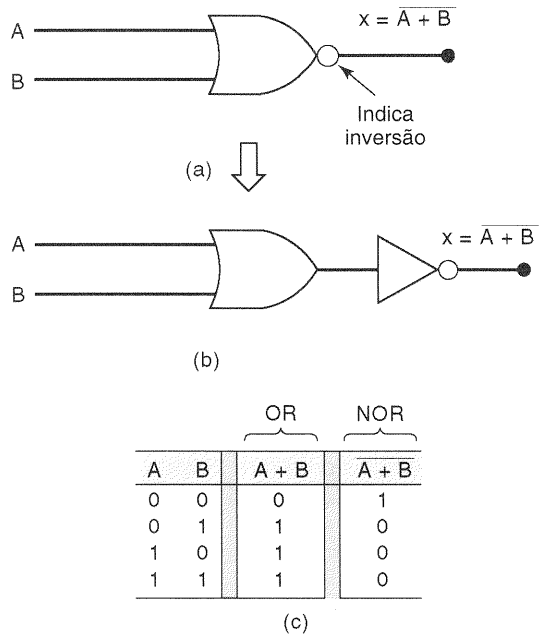
## 3-9 PORTAS NOR E PORTAS NAND

Existem dois outros tipos de portas lógicas, portas NOR e portas NAND, que são amplamente utilizadas em circuitos digitais. Estas portas, na verdade, combinam as operações básicas AND, OR e NOT. Este fato faz com que seja relativamente simples descrever o seu funcionamento utilizando as operações booleanas aprendidas anteriormente.

### Porta NOR

O símbolo para uma **porta NOR** de duas entradas pode ser visto na Fig. 3-19(a). Este símbolo é igual ao símbolo de uma porta OR, exceto pelo pequeno círculo que possui em sua saída. Este pequeno círculo representa a operação de inversão. Então, podemos dizer que uma porta NOR opera do mesmo modo que uma porta OR seguida de um INVERSOR, de modo que os circuitos mostrados na Fig. 3-19(a) e (b) são equivalentes e a expressão booleana para a saída de uma porta NOR é dada por $x = \overline{A + B}$.

A tabela-verdade, que pode ser vista na Fig. 3-19(c), mostra que a saída de uma porta NOR é exatamente o inverso da saída para uma porta OR, para todas as condições de entrada. Enquanto a saída de uma porta OR vai para o nível ALTO sempre que qualquer uma das entradas está em ALTO, a porta NOR vai para nível BAIXO sempre que qualquer uma das entradas está em ALTO. Este mesmo raciocínio pode ser estendido para portas NOR com mais de duas entradas.




| | | OR | NOR |
|---|---|---|---|
| A | B | A + B | $\overline{A + B}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

(c)

**Fig. 3-19** (a) Símbolo para porta NOR; (b) circuito equivalente; (c) tabela-verdade.

### EXEMPLO 3-8

Determine a forma de onda da saída de uma porta NOR para as formas de onda mostradas na Fig. 3-20.

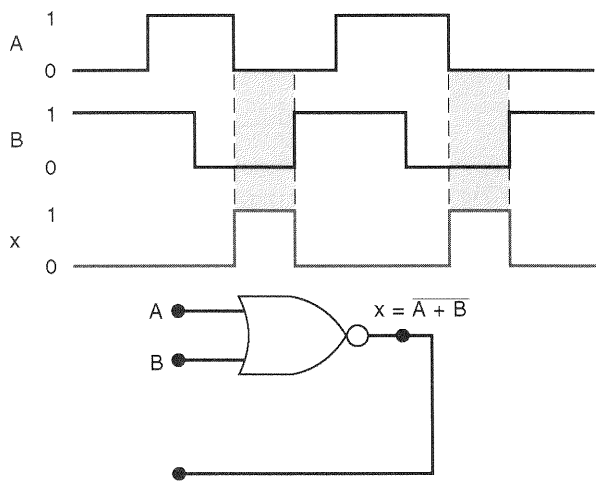



**Fig. 3-20** Exemplo 3-8.

### Solução

Existem diversas maneiras de determinar a forma de onda da saída de uma porta NOR. A primeira é encontrar a forma de onda da saída de uma porta OR e depois invertê-la, isto é, trocar todos os 1s por 0s e vice-versa. Uma outra utiliza o fato de que a saída de uma porta NOR estará em ALTO *apenas* quando todas as entradas estiverem em BAIXO. Então você pode examinar as formas de onda das entradas e encontrar os intervalos de tempo em que todas as entradas estão em BAIXO e fazer com que a saída esteja em ALTO nestes intervalos. A saída da porta NOR estará em BAIXO para todos os outros intervalos de tempo. A forma de onda resultante para a saída é mostrada na figura.

### EXEMPLO 3-9

Determine a expressão booleana para uma porta NOR de três entradas seguida de um INVERSOR.

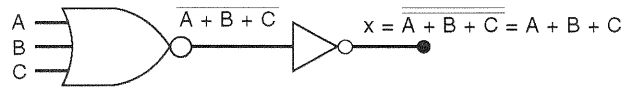



**Fig. 3-21** Exemplo 3-9.

### Solução

Observe a Fig. 3-21, onde o diagrama do circuito pode ser visto. A expressão para a saída da porta NOR é dada por $(\overline{A + B + C})$. Esta saída está conectada na entrada de um INVERSOR para produzir:

$$x = \overline{(\overline{A + B + C})}$$

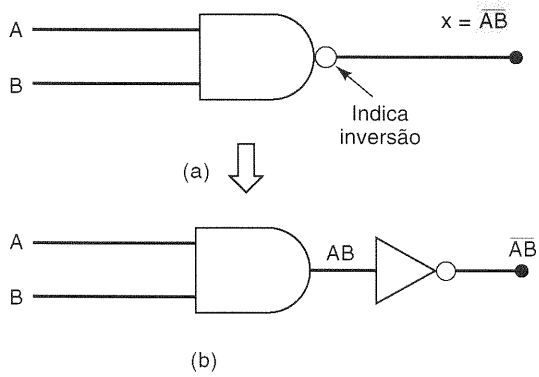
A presença de dois sinais de inversão indica que a expressão $(A + B + C)$ foi invertida e depois invertida mais uma vez. Deve estar claro que o resultado destas operações simplesmente não altera a expressão original $(A + B + C)$. Isto é,

$$x = \overline{(\overline{A + B + C})} = A + B + C$$

Sempre que duas barras de inversão estiverem sobre uma mesma variável ou expressão, elas se cancelam, como no exemplo anterior. Entretanto, em casos como $\overline{A} + \overline{B}$, as barras de inversão não se cancelam. Isto acontece porque as barras de inversão menores invertem as variáveis simples $A$ e $B$, respectivamente, enquanto as barras mais largas invertem toda a expressão $(\overline{A} + \overline{B})$. Então $\overline{\overline{A} + \overline{B}} \neq A + B$. De modo semelhante, $\overline{\overline{A} + \overline{B}} \neq AB$.

## Porta NAND

O símbolo para uma **porta NAND** de duas entradas pode ser visto na Fig. 3-22(a). Este símbolo é igual ao símbolo da porta AND, exceto pelo pequeno círculo em sua saída. Uma vez mais, este pequeno círculo representa uma operação de inversão. Então, podemos dizer que uma porta NAND funciona como uma porta AND seguida de um INVERSOR, e que portanto os circuitos das Fig. 3-22(a) e (b) são equivalentes e que a expressão booleana para a saída de uma porta NAND é $x = \overline{AB}$.




| | | AND | NAND |
|---|---|---|---|
| A | B | AB | $\overline{AB}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

(c)

**Fig. 3-22** (a) Símbolo para porta NAND; (b) circuito equivalente; (c) tabela-verdade.

A tabela-verdade vista na Fig. 3-22(c) mostra que a saída de uma porta NAND é exatamente o inverso da saída de uma porta AND para todas as condições possíveis de entrada. A saída de uma porta AND vai para ALTO quando todas as entradas estão em ALTO, enquanto a saída de uma porta NAND vai para BAIXO somente quando todas as entradas estão em ALTO. Portas NAND com mais de duas entradas também apresentam essa mesma característica.

### EXEMPLO 3-10

Determine a forma de onda da saída de uma porta NAND cujas formas de onda das entradas estão mostradas na Fig. 3-23.

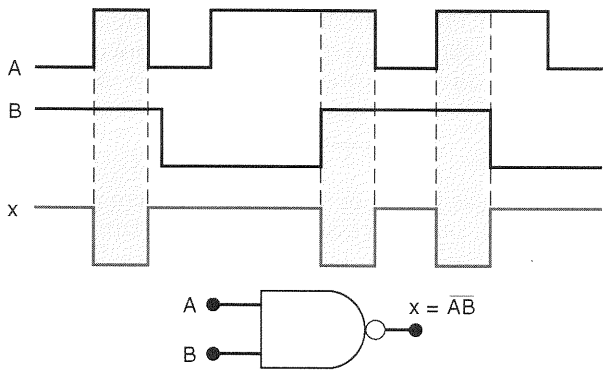



**Fig. 3-23** Exemplo 3-10.

### Solução

A forma de onda da saída pode ser determinada de várias maneiras. Uma delas é desenhar a forma de onda da saída para o caso de uma porta AND e depois inverter o resultado. Uma outra utiliza o fato de que a saída da porta NAND estará em BAIXO apenas quando todas as entradas estiverem em ALTO. Então, você pode encontrar os intervalos de tempo durante os quais todas as entradas estão em ALTO e fazer com que a saída esteja em BAIXO nesses intervalos. A saída estará em ALTO em todos os outros intervalos.

### EXEMPLO 3-11

Implemente um circuito lógico cuja expressão é $x = \overline{AB \cdot (\overline{C + D})}$ usando apenas portas NAND e NOR.

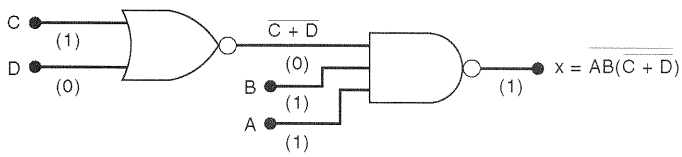



**Fig. 3-24** Exemplos 3-11 e 3-12.

### Solução

O termo $(\overline{C + D})$ é a expressão para a saída de uma porta NOR. Este termo, em conjunto com $A$ e $B$, é utilizado como entrada de uma operação AND cujo resultado final é invertido. Isto, obviamente, resulta em uma operação NAND. Assim, o circuito implementado é aquele que pode ser visto na Fig. 3-24. Observe que a porta NAND primeiro realiza uma operação AND sobre os termos $A$, $B$ e $(\overline{C+D})$ e depois inverte o resultado *inteiro*.

### EXEMPLO 3-12

Determine o nível lógico da saída na Fig. 3-24 quando $A = B = C = 1$ e $D = 0$.

### Solução

Podemos solucionar este problema de dois modos: no primeiro modo usamos a expressão booleana para $x$.

$$
\begin{aligned}
x &= \overline{AB(\overline{C + D})} \\
&= \overline{1 \cdot 1 \cdot (\overline{1 + 0})} \\
&= \overline{1 \cdot 1 \cdot (\overline{1})} \\
&= \overline{1 \cdot 1 \cdot 0} \\
&= \overline{0} = 1
\end{aligned}
$$

No segundo método, escrevemos os níveis lógicos de entrada no diagrama do circuito (mostrados entre parênteses na Fig. 3-24) e a partir desses níveis achamos os níveis lógicos da saída de cada porta até encontrarmos o resultado final. A porta NOR possui como entradas 0 e 1, o que faz a saída ser igual a 0 (em uma porta OR a saída seria 1). A porta NAND então tem como entradas os níveis lógicos 0, 1 e 1, o que faz com que a saída seja igual a 1 (em uma porta AND a saída seria igual a 0).

**Questões de Revisão**

1. Qual é o único conjunto de condições de entrada que vai gerar um nível ALTO na saída de uma porta NOR de três entradas?
2. Determine o nível da saída do circuito da Fig 3-24 para o caso em que $A = B = 1$ e $C = D = 0$.
3. Troque a porta NOR da Fig. 3-24 por uma porta NAND e troque também a porta NAND por uma porta NOR. Qual é a nova expressão booleana para $x$?

## 3-10 TEOREMAS DA ÁLGEBRA BOOLEANA

Vimos como a álgebra booleana pode ser usada para nos ajudar a analisar um circuito lógico e expressar sua operação matematicamente. Continuaremos nosso estudo da álgebra booleana investigando seus vários teoremas (regras), chamados **teoremas booleanos,** que podem nos ajudar a simplificar expressões e circuitos lógicos. O primeiro grupo de teoremas é mostrado na Fig. 3-25. Em cada um deles, $x$ é uma variável lógica que pode ser igual a 0 ou 1. Cada teorema está acompanhado por um circuito lógico que demonstra sua validade.

O teorema (1) mostra que o resultado de uma operação AND que tem como entradas uma variável qualquer $x$ e 0 deve ser igual a 0. Isto é fácil de lembrar porque a operação AND é como a multiplicação ordinária, onde sabemos que o resultado de multiplicar qualquer coisa por 0 é 0. Sabemos também que a saída de uma porta AND será 0 sempre que qualquer uma das entradas for 0, independentemente do nível lógico da outra entrada.

O teorema (2) é também óbvio, se fizermos mais uma vez a comparação da multiplicação ordinária com a operação AND.

O teorema (3) pode ser provado verificando o resultado para cada valor possível de entrada. Se $x = 0$, então $0 \cdot 0 = 0$; se $x = 1$, então $1 \cdot 1 = 1$. Portanto, $x \cdot x = x$.

O teorema (4) pode ser provado do mesmo modo. Entretanto, podemos raciocinar que em qualquer instante ou $x$ ou seu inverso $\overline{x}$ deve ser igual a 0 e, então, uma operação AND de $x$ com seu inverso será sempre igual a 0.

O teorema (5) é direto, uma vez que 0 *adicionado* a qualquer valor não altera esse valor, seja na adição ordinária ou na operação OR.

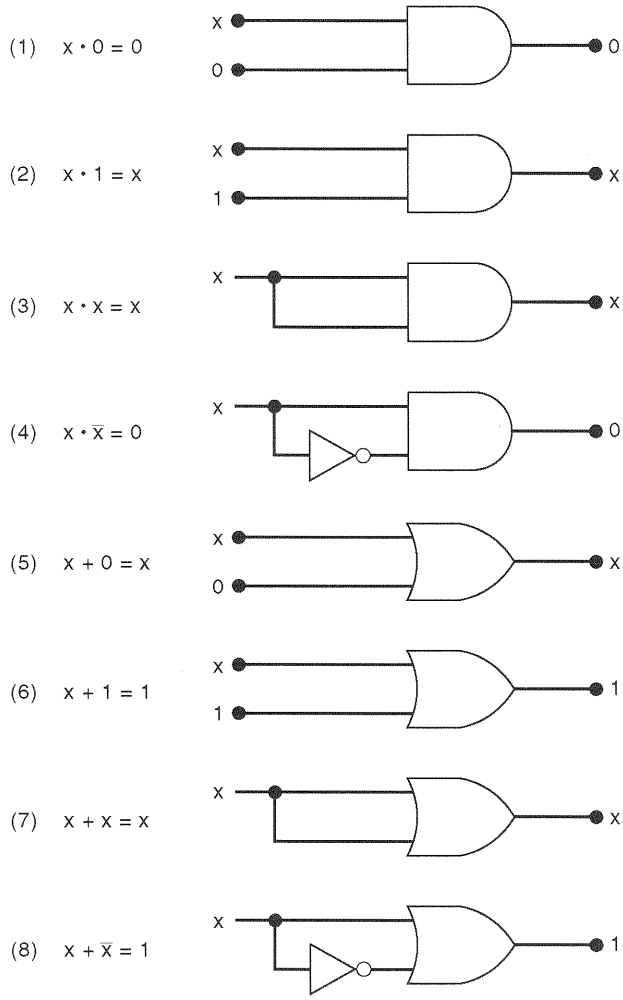



**Fig. 3-25** Teoremas de uma variável.

O teorema (6) afirma que o resultado de uma operação OR que possui como entradas uma variável qualquer $x$ e 1 será sempre igual a 1. Podemos fazer a verificação deste teorema para os dois valores possíveis de $x$: $0 + 1 = 1$ e $1 + 1 = 1$. De modo equivalente, podemos lembrar que a saída de uma porta OR de duas entradas será igual a 1 quando *qualquer* uma das entradas for igual a 1, não importando o valor da outra entrada.

O teorema (7) pode ser verificado para ambos os valores de $x$: $0 + 0 = 0$ e $1 + 1 = 1$.

O teorema (8) pode ser provado de modo similar, ou podemos raciocinar que em qualquer instante $x$ ou seu inverso $\overline{x}$ estará em nível lógico 1, então sempre teremos a operação OR de 0 e 1, cujo resultado será sempre 1.

Antes de introduzirmos mais teoremas, devemos enfatizar que quando os teoremas de (1) a (8) são aplicados, a variável $x$ pode, na verdade, representar uma expressão que contenha mais de uma variável. Por exemplo, se tivermos a expressão $A\overline{B}(\overline{A\overline{B}})$, podemos aplicar o teorema (4) se fizermos $x = A\overline{B}$. Então podemos dizer que $A\overline{B}(\overline{A\overline{B}}) = 0$. Este mesmo raciocínio pode ser aplicado para o uso de qualquer um destes teoremas.

## Teoremas com Mais de Uma Variável

Os teoremas apresentados a seguir envolvem o uso de mais de uma variável:

(9) $x + y = y + x$
(10) $x \cdot y = y \cdot x$
(11) $x + (y + z) = (x + y) + z = x + y + z$
(12) $x(yz) = (xy)z = xyz$
(13*a*) $x(y + z) = xy + xz$
(13*b*) $(w + x)(y + z) = wy + xy + wz + xz$
(14) $x + xy = x$
(15) $x + \overline{x}y = x + y$

Os teoremas (9) e (10) são conhecidos como *leis da comutatividade*. Estas leis determinam que a ordem na qual realizamos as operações AND e OR não é importante. O resultado é o mesmo.

Os teoremas (11) e (12) são conhecidos como *leis da associatividade*; elas afirmam que podemos agrupar as variáveis de expressões do tipo AND ou OR do modo que desejarmos.

O teorema (13) é a *lei da distributividade*, que afirma que uma expressão pode ser expandida multiplicando-se termo a termo, do mesmo modo que é feito na álgebra comum. Este teorema também afirma que podemos fatorar uma expressão. Caso tenhamos a soma de dois (ou mais) termos, cada um contendo uma variável comum, podemos fatorar essa variável como fazemos na álgebra comum. Por exemplo, na expressão $A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$, podemos fatorar a variável $\overline{B}$:

$$A\overline{B}C + \overline{A}\overline{B}\overline{C} = \overline{B}(AC + \overline{A}\overline{C})$$

Como um outro exemplo, considere a expressão $ABC + ABD$. Neste caso, estes dois termos têm as variáveis $A$ e $B$ em comum, e portanto $A \cdot B$ pode ser fatorado, como vemos a seguir.

$$ABC + ABD = AB(C + D)$$

Os teoremas (9) a (13) são fáceis de lembrar porque são idênticos àqueles utilizados na álgebra comum. Os teoremas (14) e (15), por outro lado, não possuem correspondentes na álgebra comum. Cada um deles pode ser demonstrado substituindo $x$ e $y$ na expressão por todos os diferentes casos possíveis, conforme demonstrado para o teorema (14):

**Caso 1.** Para $x = 0$, $y = 0,1$

$$\begin{aligned} x + xy &= x \\ 0 + 0 \cdot 0 &= 0 \\ 0 &= 0 \end{aligned}$$

**Caso 2.** Para $x = 0$, $y = 1$,

$$\begin{aligned} x + xy &= x \\ 0 + 0 \cdot 1 &= 0 \\ 0 + 0 &= 0 \\ 0 &= 0 \end{aligned}$$

**Caso 3.** Para $x = 1$, $y = 0$

$$\begin{aligned} x + xy &= x \\ 1 + 1 \cdot 0 &= 1 \\ 1 + 0 &= 1 \\ 1 &= 1 \end{aligned}$$

**Caso 4.** Para $x = 1$, $y = 1$,

$$\begin{aligned} x + xy &= x \\ 1 + 1 \cdot 1 &= 1 \\ 1 + 1 &= 1 \\ 1 &= 1 \end{aligned}$$

O teorema (14) também pode ser demonstrado através de fatoração e do uso dos teoremas (6) e (2).

$$\begin{aligned} x + xy &= x(1 + y) \\ &= x \cdot 1 \quad \text{[usando o teorema (6)]} \\ &= x \quad \text{[usando o teorema (2)]} \end{aligned}$$

Todos esses teoremas da álgebra booleana podem ser úteis na simplificação de uma expressão lógica, isto é, na redução do número de termos da expressão. Quando isto é feito, a expressão simplificada dá origem a um circuito que é menos complexo do que aquele que a expressão original produziria. Uma boa parte do próximo capítulo será dedicada ao processo de simplificação de circuitos. Por enquanto, os exemplos seguintes servem para ilustrar como os teoremas booleanos podem ser aplicados.

---

**EXEMPLO 3-13**

Simplifique a expressão $y = A\overline{B}D + A\overline{B}\,\overline{D}$.

**Solução**

Fatore as variáveis comuns $A\overline{B}$ utilizando o teorema (13):

$$y = A\overline{B}(D + \overline{D})$$

Pelo teorema (8), o termo entre parênteses é igual a 1 e portanto,

$$\begin{aligned} y &= A\overline{B} \cdot 1 \\ &= A\overline{B} \quad \text{[usando o teorema (2)]} \end{aligned}$$

---

**EXEMPLO 3-14**

Simplifique $z = (\overline{A} + B)(A + B)$.

**Solução**

A expressão pode ser expandida multiplicando-se os termos [teorema (13)]:

$$z = \overline{A} \cdot A + \overline{A} \cdot B + B \cdot A + B \cdot B$$

Pelo teorema (4), $\overline{A} \cdot A = 0$. Além disso, $B \cdot B = B$ [teorema (3)]:

$$z = 0 + \overline{A} \cdot B + B \cdot A + B = \overline{A}B + AB + B$$

Fatorando a variável $B$ [teorema (13)], temos

$$z = B(\overline{A} + A + 1)$$

Finalmente, utilizando os teoremas (2) e (6),

$$z = B$$

---

**EXEMPLO 3-15**

Simplifique $x = ACD + \overline{A}BCD$.

**Solução**

Fatorando as variáveis comuns $CD$, temos

$$x = CD(A + \overline{A}B)$$

Utilizando o teorema (15), podemos substituir $A + \overline{A}B$ por $A + B$, e então

$$\begin{aligned} x &= CD(A + B) \\ &= ACD + BCD \end{aligned}$$

**Questões de Revisão**

1. Use os teoremas (13) e (14) para simplificar $y = A\overline{C} + AB\overline{C}$.
2. Use os teoremas (13) e (8) para simplificar $y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$.

## 3-11 TEOREMAS DE DEMORGAN

Dois dos mais importantes teoremas da álgebra booleana são atribuídos a um grande matemático chamado DeMorgan. Os **teoremas de DeMorgan** são extremamente úteis para simplificar expressões nas quais o produto (AND) ou a soma (OR) das variáveis é invertido. Os dois teoremas são:

$$(16) \qquad \overline{(x + y)} = \overline{x} \cdot \overline{y}$$
$$(17) \qquad \overline{(x \cdot y)} = \overline{x} + \overline{y}$$

O teorema (16) diz que quando uma soma OR está invertida, esta é igual ao produto AND das variáveis invertidas. O teorema (17) diz que quando um produto AND de duas variáveis está invertido, este é igual a uma soma OR das variáveis invertidas. Cada um dos teoremas de DeMorgan pode ser prontamente demonstrado verificando-o para todas as combinações possíveis de valores para $x$ e $y$. Esta demonstração é deixada para ser feita como exercício ao final do capítulo.

Apesar de esses teoremas terem sido enunciados em termos de variáveis simples $x$ e $y$, eles são igualmente válidos para situações nas quais $x$ e/ou $y$ são expressões que contenham mais de uma variável. Por exemplo, a aplicação destes teoremas na expressão $\overline{(A\overline{B} + C)}$ pode ser vista a seguir:

$$\overline{(A\overline{B} + C)} = \overline{(A\overline{B})} \cdot \overline{C}$$

Note que aqui tratamos $A\overline{B}$ como $x$ e $C$ como $y$. O resultado pode depois ser simplificado já que temos um produto $A\overline{B}$ que é invertido. Usando o teorema (17), a expressão se torna

$$\overline{A\overline{B}} \cdot \overline{C} = (\overline{A} + \overline{\overline{B}}) \cdot \overline{C}$$

Observe que podemos substituir $\overline{\overline{B}}$ por $B$, e então finalmente temos

$$(\overline{A} + B) \cdot \overline{C} = \overline{A}\,\overline{C} + B\overline{C}$$

Este resultado final possui sinais de inversão apenas em variáveis simples.

**EXEMPLO 3-16**

Simplifique a expressão $z = \overline{(\overline{A}+C) \cdot (B+\overline{D})}$ para uma outra que contenha apenas variáveis simples invertidas.

**Solução**

Utilizando o teorema (17), podemos reescrever a expressão anterior como

$$z = \overline{(\overline{A} + C)} + \overline{(B + \overline{D})}$$

Podemos pensar nesse procedimento como partir o sinal de inversão ao meio e trocar sinais AND (·) por sinais OR (+). Agora o termo $\overline{(\overline{A} + C)}$ pode ser simplificado aplicando-se o teorema (16). Do mesmo modo, $\overline{(B+\overline{D})}$ pode ser simplificado como se segue:

$$\begin{aligned} z &= \overline{(\overline{A} + C)} + \overline{(B + \overline{D})} \\ &= (\overline{\overline{A}} \cdot \overline{C}) + \overline{B} \cdot \overline{\overline{D}} \end{aligned}$$

Nesta simplificação, partimos o sinal de inversão ao meio e trocamos os sinais (+) por (·). A seguir, cancelamos as inversões duplas e temos finalmente

$$z = A\overline{C} + \overline{B}D$$

O Exemplo 3-16 mostra que, quando se utilizam os teoremas de DeMorgan para simplificar uma expressão, o que fazemos é partir o sinal de inversão em qualquer ponto na expressão e então mudar o sinal do operador que estiver neste ponto (+ é trocado por · e vice-versa). Este procedimento pode ser continuado até que a expressão seja reduzida a uma outra na qual apenas variáveis simples encontram-se invertidas. Outros dois exemplos podem ser vistos a seguir:

***Exemplo 1***

$$\begin{aligned} z &= \overline{A + \overline{B} \cdot C} \\ &= \overline{A} \cdot \overline{(\overline{B} \cdot C)} \\ &= \overline{A} \cdot (\overline{\overline{B}} + \overline{C}) \\ &= \overline{A} \cdot (B + \overline{C}) \end{aligned}$$

***Exemplo 2***

$$\begin{aligned} \omega &= \overline{(A + BC) \cdot (D + EF)} \\ &= \overline{(A + BC)} + \overline{(D + EF)} \\ &= (\overline{A} \cdot \overline{BC}) + (\overline{D} \cdot \overline{EF}) \\ &= [\overline{A} \cdot (\overline{B} + \overline{C})] + [\overline{D} \cdot (\overline{E} + \overline{F})] \\ &= \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{D}\,\overline{E} + \overline{D}\,\overline{F} \end{aligned}$$

Os teoremas de DeMorgan podem ser facilmente estendidos para mais do que duas variáveis. Por exemplo, pode-se provar que:

$$\begin{aligned} \overline{x + y + z} &= \overline{x} \cdot \overline{y} \cdot \overline{z} \\ \overline{x \cdot y \cdot z} &= \overline{x} + \overline{y} + \overline{z} \end{aligned}$$

Aqui podemos ver que o grande sinal de inversão foi partido em *dois* pontos e, nesses pontos, o sinal do operador foi trocado por seu oposto. Esse raciocínio pode ser estendido para um número qualquer de variáveis. Mais uma vez, observe que as variáveis podem ser expressões em lugar de variáveis simples. Veja um outro exemplo:

$$\begin{aligned} x &= \overline{A\overline{B} \cdot \overline{CD} \cdot \overline{EF}} \\ &= \overline{\overline{A\overline{B}}} + \overline{\overline{CD}} + \overline{\overline{EF}} \\ &= AB + CD + EF \end{aligned}$$

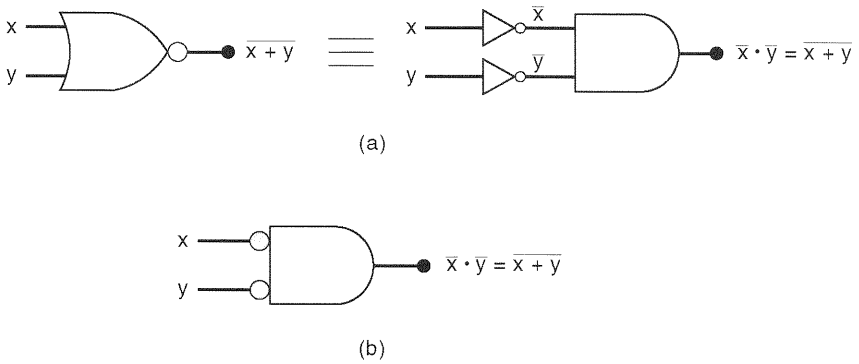



**Fig. 3-26** (a) Circuitos equivalentes obtidos pela aplicação do teorema (16); (b) símbolo alternativo para a função NOR.

## Implicações dos Teoremas de DeMorgan

Vamos examinar os teoremas (16) e (17) do ponto de vista de circuitos lógicos. Primeiro considere o teorema (16):

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$

O lado esquerdo da equação pode ser visto como a saída de uma porta NOR cujas entradas são $x$ e $y$. O lado direito da equação, por outro lado, pode ser visto como a saída de uma porta AND cujas entradas são as variáveis $x$ e $y$ invertidas. Estas duas representações são equivalentes e estão ilustradas na Fig. 3-26(a). Isto significa que uma porta AND com inversores em cada uma de suas entradas é equivalente a uma porta NOR. Na verdade, ambas as representações são usadas para representar a função NOR. Quando a porta AND com entradas invertidas é usada para representar a função NOR, esta é geralmente desenhada como mostrado na Fig. 3-26(b), onde os pequenos círculos nas entradas representam a operação de inversão.

Agora considere o teorema (17):

$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

O lado esquerdo da equação pode ser implementado através de uma porta NAND com entradas $x$ e $y$. O lado direito pode ser implementado por uma porta OR que tenha como entradas $x$ e $y$ invertidas. Estas duas representações equivalentes são mostradas na Fig.3-27(a). Uma porta OR com inversores em cada uma de suas entradas é equivalente a uma porta NAND. Na verdade, ambas as representações são usadas para representar a função NAND. Quando a porta OR com entradas invertidas é usada para representar a função NAND, esta é freqüentemente desenhada como mostra a Fig. 3-27(b), onde os círculos mais uma vez representam inversão.

### EXEMPLO 3-17

Determine a expressão lógica para a saída do circuito da Fig. 3-28 e simplifique-a usando os teoremas de DeMorgan.

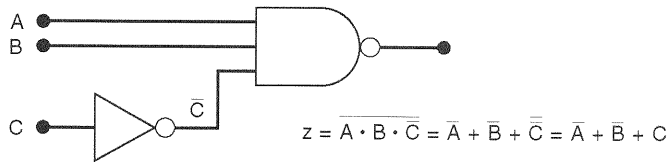



**Fig. 3-28** Exemplo 3-17.

### Solução

A expressão para $z$ é $z = \overline{ABC}$. Aplique o teorema de DeMorgan para partir o sinal de inversão como é mostrado a seguir:

$$z = \overline{A} + \overline{B} + \overline{\overline{C}}$$

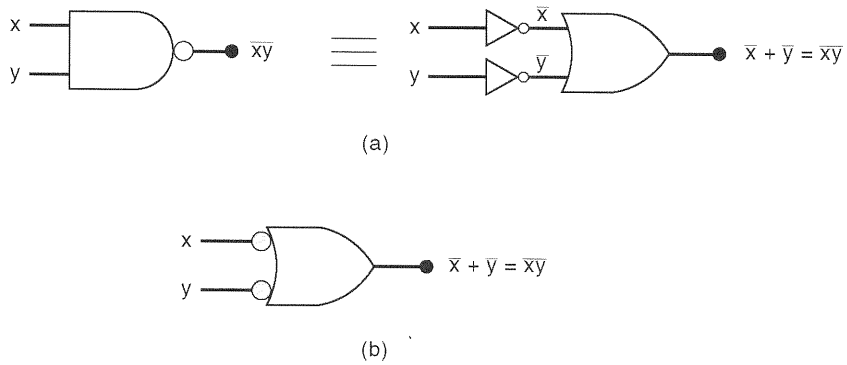



**Fig. 3-27** (a) Circuitos equivalentes obtidos pela aplicação do teorema (17); (b) símbolo alternativo para a função NAND.

Cancele a dupla inversão sobre $C$ para obter

$$z = \overline{A} + \overline{B} + C$$

**Questões de Revisão**

1. Use o teorema de DeMorgan para converter a expressão $z = \overline{(A+B)\cdot C}$ para uma outra que possua apenas inversões em variáveis simples.
2. Repita a questão 1 para a expressão $y = \overline{\overline{RST}+\overline{Q}}$.
3. Implemente um circuito cuja expressão para a saída é $z = \overline{A}\,\overline{B}C$ usando apenas uma porta NOR e um INVERSOR.
4. Use os teoremas de DeMorgan para converter $y = \overline{A+\overline{B}+\overline{CD}}$ para uma outra expressão que contenha apenas inversões em variáveis simples.

## 3-12 UNIVERSALIDADE DAS PORTAS NAND E NOR

Todas as expressões booleanas consistem em várias combinações das operações básicas OR, AND e NOT. Assim, qualquer expressão pode ser implementada usando combinações das portas AND, OR e INVERSORes. É possível, entretanto, implementar qualquer expressão lógica usando-se *apenas* portas NAND. Isto acontece porque portas NAND, em combinações apropriadas, podem ser usadas para representar cada uma das operações lógicas OR, AND e NOT. Isto pode ser visto na Fig. 3-29.

Em primeiro lugar, na Fig. 3-29(a), temos uma porta NAND de duas entradas onde estas se encontram propositadamente conectadas a uma mesma variável $A$. Nessa configuração, a porta NAND simplesmente age como um simples INVERSOR, uma vez que sua saída $x = \overline{A\cdot A} = \overline{A}$.

Na Fig. 3-29(b) temos duas portas NAND conectadas de tal modo que a operação AND seja realizada. A porta NAND número 2 é usada como um INVERSOR para que a expressão $\overline{AB}$ seja transformada em $\overline{\overline{AB}} = AB$, que é a função AND desejada.

A operação OR pode ser implementada usando portas NAND como está mostrado na Fig. 3-29(c). Neste caso, as portas NAND número 1 e 2 são usadas como INVERSORes para inverter as entradas, de modo que o resultado final seja $x = \overline{\overline{A}\cdot\overline{B}}$, que pode ser simplificado para $x = A + B$ através do teorema de DeMorgan.

De modo similar, pode ser mostrado que portas NOR podem ser combinadas para implementar qualquer uma das operações booleanas. Isto é ilustrado na Fig. 3-30. O item (a) mostra que uma porta NOR com as entradas conectadas juntas comporta-se como um INVERSOR, uma vez que sua saída é $x = \overline{A+A} = \overline{A}$.

Na Fig. 3-30(b), duas portas NOR são combinadas de modo a implementar a operação OR. A porta NOR número 2 é usada como um INVERSOR para modificar a expressão $\overline{A+B}$ em $\overline{\overline{A+B}} = A + B$, que é a operação OR desejada.

A operação AND pode ser implementada com portas NOR como pode ser visto na Fig. 3-30(c). Neste caso, as portas NOR 1 e 2 são usadas como INVERSORes para inverter as entradas, de modo que a saída $x$ seja igual a $x = \overline{\overline{A}+\overline{B}}$, que pode ser simplificado para $x = A \cdot B$, pelo uso do teorema de DeMorgan.

Uma vez que qualquer uma das operações booleanas pode ser implementada usando apenas portas NAND, qualquer circuito lógico pode ser construído usando apenas portas NAND. O mesmo é válido para portas NOR. Essa característica das portas NAND e NOR pode ser bastante útil no projeto de circutos lógicos, como mostra o exemplo a seguir.

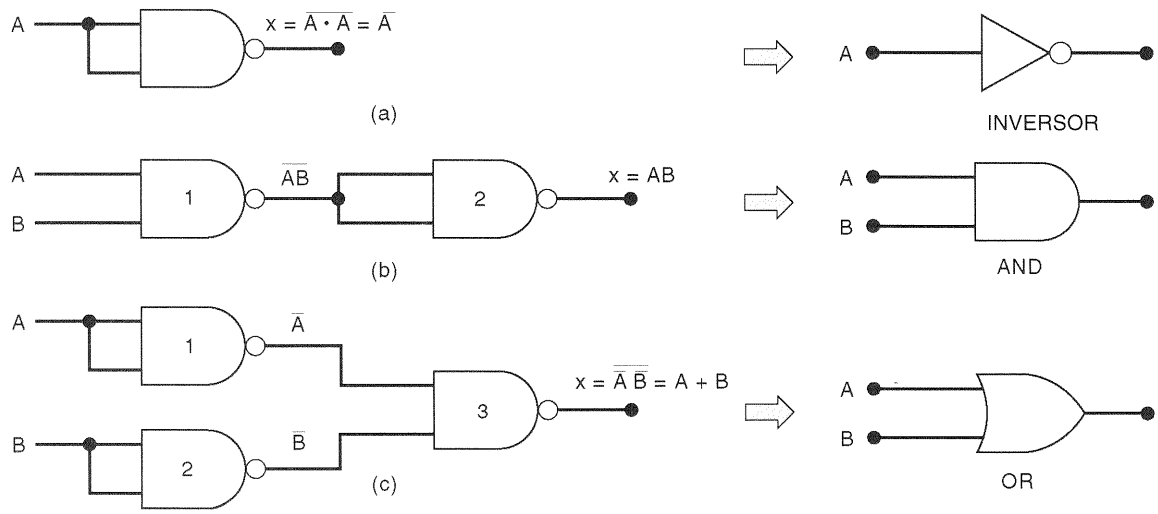



**Fig 3-29** Portas NAND podem ser usadas para implementar qualquer função booleana.

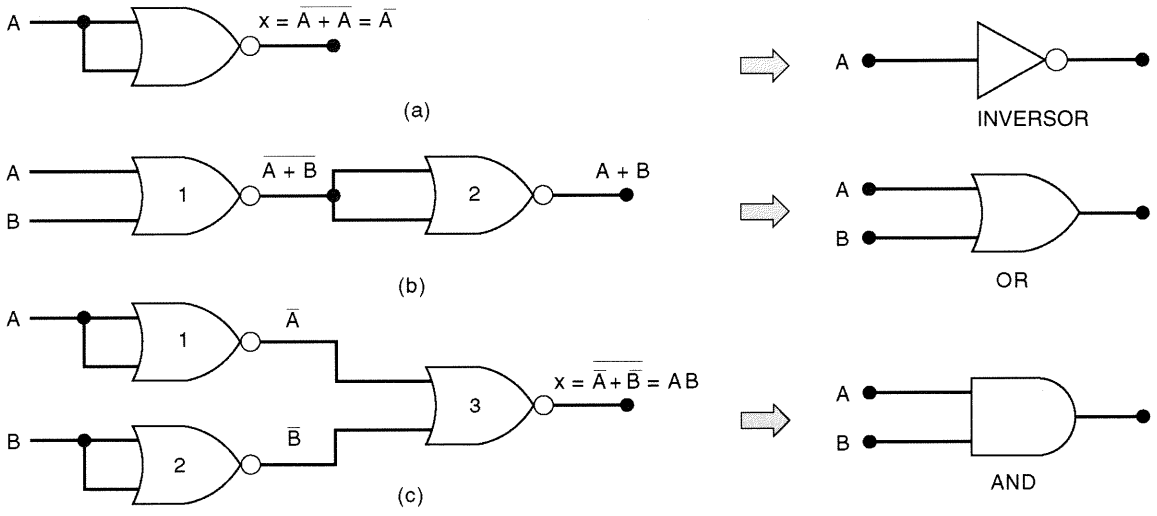



**Fig. 3-30** Portas NOR podem ser usadas para implementar qualquer função booleana.

## EXEMPLO 3-18

Em um determinado processo de fabricação, uma esteira de transporte deve ser desligada sempre que determinadas condições ocorrerem. Estas condições são monitoradas e são representadas pelo estado de quatro sinais lógicos como se segue: o sinal $A$ deve estar em nível ALTO sempre que a esteira de transporte estiver muito rápida; o sinal $B$ deve estar em nível ALTO sempre que o recipiente localizado no final da esteira estiver cheio; o sinal $C$ deve estar em nível ALTO sempre que a tensão na esteira estiver muito alta; o sinal $D$ deve estar em nível ALTO sempre que o comando manual estiver desabilitado.

Um circuito lógico é necessário para gerar um sinal $x$ que deve estar em ALTO sempre que as condições $A$ e $B$ existirem simultaneamente, ou sempre que as condições $C$ e $D$ existirem simultaneamente. Obviamente, a expressão lógica para $x$ deve ser igual a $x = AB + CD$. O circuito deve ser implementado com um número mínimo de CIs. Os circuitos integrados TTL mostrados na Fig. 3-31 estão disponíveis. Cada CI é *quádruplo*, o que significa que ele contém *quatro* portas idênticas em um chip.

### Solução

O método mais direto para se implementar a expressão dada usa duas portas AND e uma porta OR, como pode ser visto na Fig. 3-32(a). Esta implementação utiliza duas portas do CI 74LS08 e uma única porta do CI 74LS32. Os números entre parênteses, em cada entrada e saída, são os números dos pinos dos respectivos CIs. Estes números são sempre mostrados em qualquer diagrama de circuito lógico. Para os nossos propósitos, a maioria dos diagramas lógicos não mostrará o número dos pinos, a não ser que eles sejam necessários para descrever a operação do circuito.

Uma outra implementação pode ser obtida a partir do circuito da Fig. 3-32(a), se trocarmos cada uma das portas AND e OR pelas suas implementações com portas NAND equivalentes. O resultado desta operação é mostrado na Fig. 3-32(b).

À primeira vista, esse novo circuito parece necessitar de sete portas NAND. Entretanto, as portas NAND de número 3 e 5 estão conectadas como INVERSORES em série e podem ser eliminadas do circuito, uma vez que realizam uma dupla inversão da saída da porta NAND número 1. Do mesmo modo, as portas NAND 4 e 6 também podem ser eliminadas. O circuito final, após a eliminação dos INVERSORes duplos, está desenhado na Fig. 3-32(c).

Esse circuito é mais eficiente do que o da Fig. 3-32(a) porque utiliza três portas NAND de duas entradas que podem ser implementadas por um CI, o 74LS00.

### Questões de Revisão

1. Quantas maneiras diferentes temos agora para implementar a operação de inversão em um circuito lógico?
2. Implemente a expressão $x = (A + B)(C + D)$ usando portas AND e OR. Agora implemente esta expressão utilizando apenas portas NOR. Para isso, converta cada porta AND e OR que seja necessária pela sua implementação em portas NOR, como visto na Fig. 3-30. Qual circuito é mais eficiente?
3. Escreva a expressão para a saída do circuito da Fig. 3-32(c) e use o teorema de DeMorgan para mostrar que esta é equivalente à expressão dada para o circuito da Fig. 3-32(a).

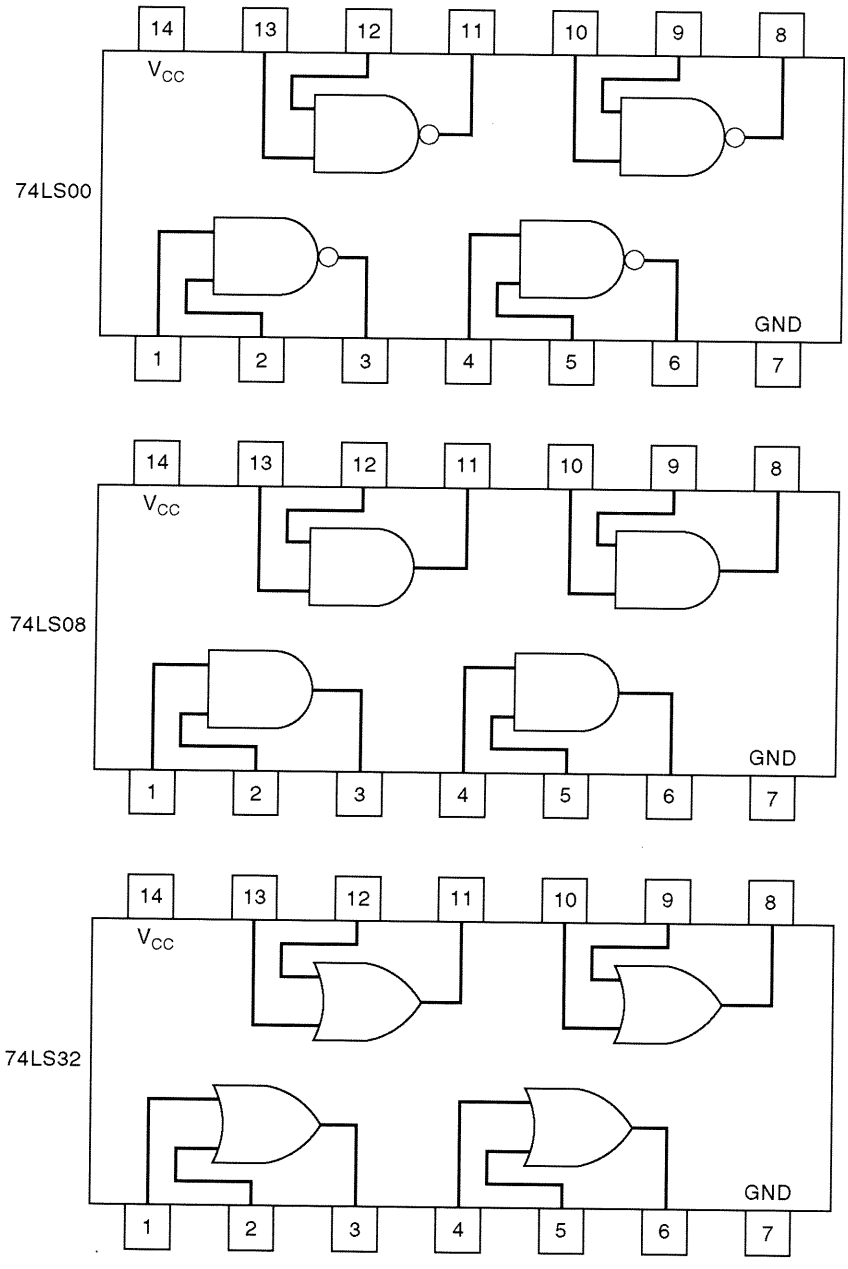



**Fig. 3-31** CIs disponíveis para o Exemplo 3-18.

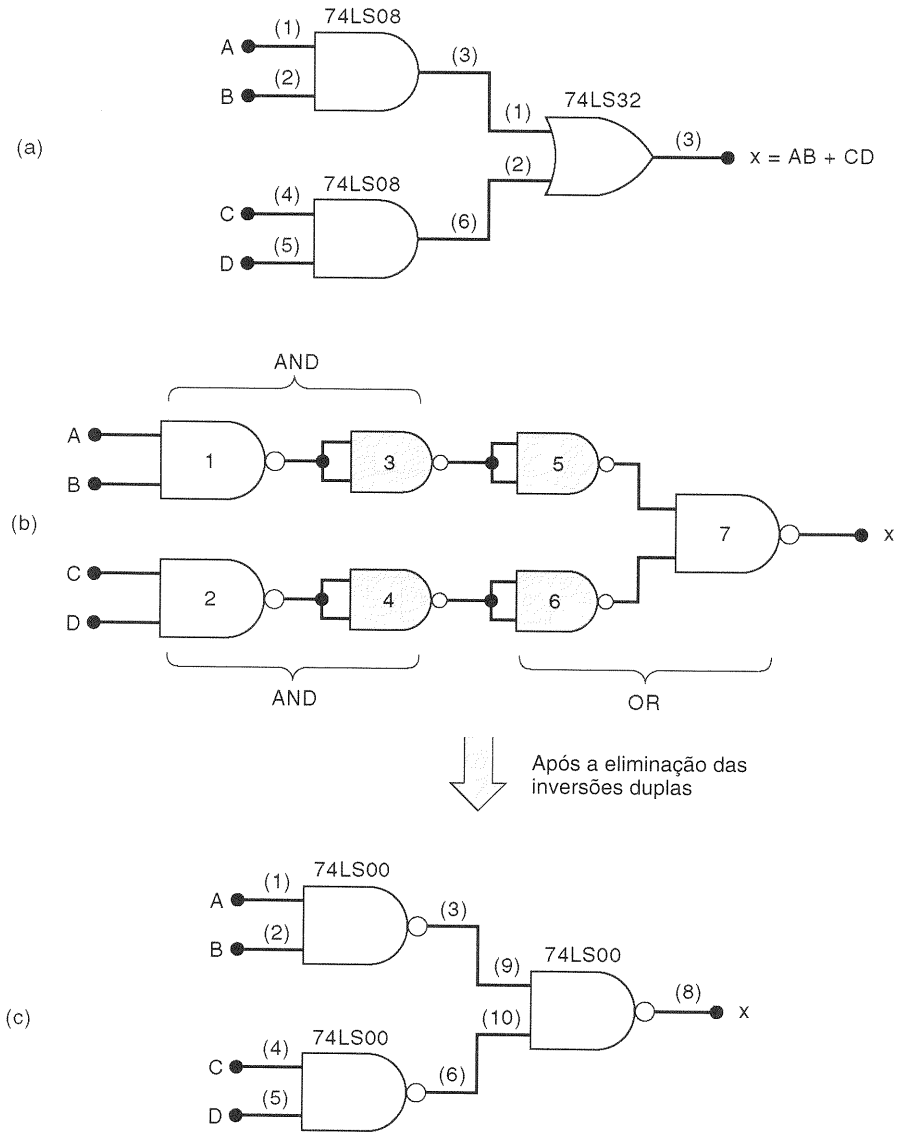



**Fig. 3-32** Implementações possíveis para o Exemplo 3-18.

## 3-13 REPRESENTAÇÕES ALTERNATIVAS DAS PORTAS LÓGICAS

Introduzimos as cinco portas lógicas básicas (AND, OR, NOT, NAND e NOR) e os símbolos padronizados usados para representá-las em diagramas de circuitos lógicos. Embora você possa encontrar alguns diagramas de circuitos que ainda utilizam exclusivamente estes símbolos, é cada vez mais comum encontrarmos diagramas de circuitos que utilizam **símbolos lógicos alternativos em conjunto** com os símbolos padronizados.

Antes de discutirmos as razões para utilizar um símbolo alternativo para uma porta lógica, apresentaremos os símbolos alternativos para cada porta lógica e mostraremos que eles são equivalentes aos símbolos padronizados. Observe a Fig. 3-33. O lado esquerdo da figura mostra o símbolo padronizado para cada porta lógica, e o lado direito mostra os símbolos alternativos. O símbolo alternativo para cada porta é obtido a partir do símbolo original fazendo-se o seguinte:

1. Inverta cada entrada e saída do símbolo padronizado. Isto é feito adicionando bolhas (pequenos círculos) em entradas e saídas que não possuem bolhas e removendo-as de onde elas já existem.
2. Troque o símbolo da operação AND pelo símbolo da operação OR ou troque de OR para AND (no caso especial do INVERSOR, o símbolo da operação não é trocado).

Por exemplo, o símbolo padronizado NAND é o símbolo AND com uma bolha em sua saída. Seguindo os passos descritos anteriormente, temos que remover a bolha da saída e adicionar uma bolha para cada entrada. Feito isto, podemos trocar o símbolo AND pelo símbolo OR. O resultado será um símbolo OR com bolhas em suas entradas.

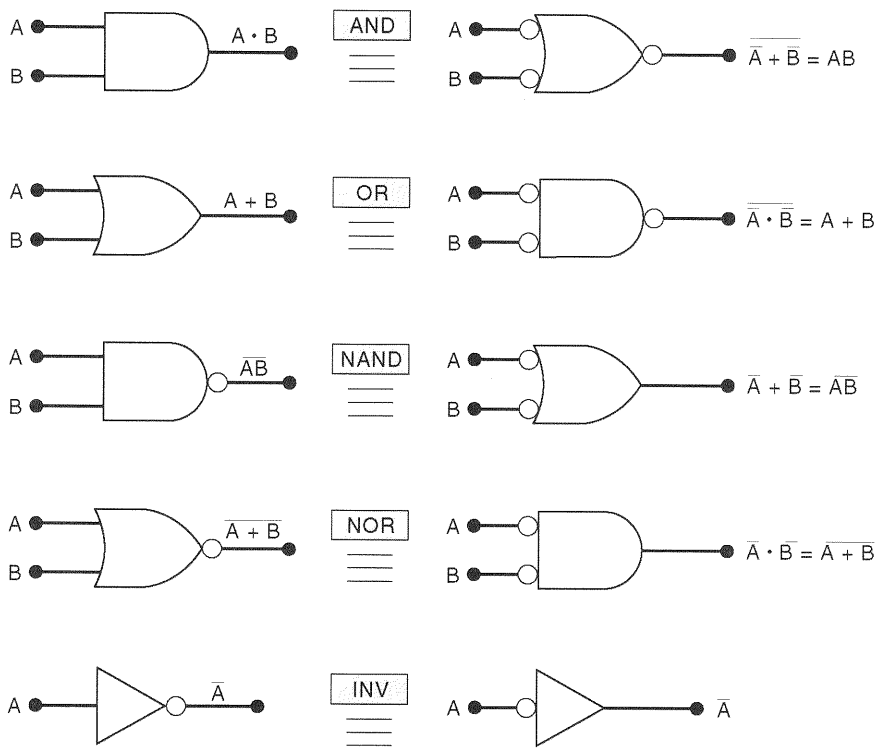



**Fig. 3-33** Símbolos padronizados e alternativos para várias portas lógicas e para o inversor.

Podemos facilmente provar que esse símbolo alternativo é equivalente ao símbolo padronizado utilizando o teorema de DeMorgan e lembrando que a bolha representa uma operação de inversão. A expressão para a saída de um símbolo padronizado NAND é $\overline{AB} = \overline{A} + \overline{B}$, que é a mesma expressão para a saída do símbolo alternativo. Este mesmo procedimento pode ser seguido para cada um dos pares de símbolos da Fig. 3-33.

Vários pontos devem ser enfatizados no que se refere às equivalências de símbolos lógicos:

1. As equivalências podem ser estendidas a portas com *qualquer* número de entradas.
2. Nenhum dos símbolos padronizados tem bolhas em suas entradas, ao passo que todos os alternativos têm.
3. Os símbolos padronizados e alternativos para cada porta representam o mesmo circuito físico. *Não há diferenças nos circuitos representados pelos dois símbolos.*
4. NAND e NOR são portas inversoras, e portanto os símbolos padronizado e alternativo para cada uma terão uma bolha *ou* na entrada *ou* na saída. AND e OR são portas *não-inversoras*, e portanto os símbolos alternativos terão bolhas *tanto nas* entradas *quanto* na saída.

## Interpretação dos Símbolos Lógicos

Cada um dos símbolos lógicos da Fig. 3-33 fornece uma interpretação única do funcionamento da porta. Antes de demonstrar estas interpretações, devemos primeiro estabelecer o conceito de **níveis lógicos ativos.**

Quando uma linha de entrada ou saída de um símbolo de circuito lógico *não possui a bolha de inversão*, diz-se que esta linha é ativa em nível lógico ALTO (**ativa-ALTO**). Quando a linha de entrada ou saída *possui a bolha de inversão*, diz-se que esta linha é ativa em nível lógico BAIXO (**ativa-BAIXO**). A presença ou ausência da bolha de inversão, então, determina a condição ativa-ALTO/ativa-BAIXO das entradas ou saídas e é usada para interpretar a operação do circuito.

Para ilustrar, a Fig. 3-34(a) mostra o símbolo padronizado para uma porta NAND. O símbolo tem uma bolha de inversão em sua saída e não possui bolhas nas entradas. Então ela possui uma saída ativa-BAIXO e entradas do tipo ativa-ALTO. A operação lógica representada por esse símbolo pode ser interpretada da seguinte maneira:

**A saída vai para BAIXO somente quando *todas* as entradas estão em ALTO.**

Observe que esta frase diz que a entrada irá para o seu estado ativo somente quando *todas* as entradas estiverem em seus estados ativos. A palavra "todas" é usada por causa do símbolo AND.

O símbolo alternativo para a porta NAND mostrado na Fig. 3-34(b) possui uma saída ativa-ALTO e entradas do tipo ativa-BAIXO. Assim, sua operação pode ser descrita como se segue:

**A saída vai para ALTO quando *qualquer das entradas* estiver em BAIXO.**

Esta afirmação nos diz que a saída estará no seu estado ativo sempre que *qualquer* uma das entradas estiver no seu estado ativo. A palavra "qualquer" é usada por causa do símbolo OR.

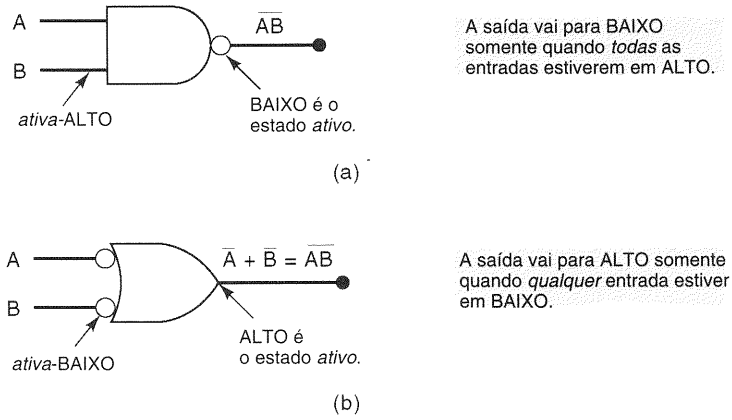



**Fig. 3-34** Interpretação dos dois símbolos das portas NAND.

Com um pouco de raciocínio, podemos notar que essas duas interpretações para os símbolos da porta NAND na Fig. 3-34 são maneiras diferentes de dizer a mesma coisa.

## Resumo

Neste ponto, você deve estar imaginando por que existe a necessidade de termos dois símbolos e interpretações diferentes para cada porta lógica. Felizmente, as razões disso ficarão claras após estudarmos a próxima seção. Por enquanto, vamos resumir pontos importantes referentes à representação de portas lógicas.

1. Para obter o símbolo alternativo para cada porta lógica, tome o símbolo padronizado e troque seu símbolo de operação (OR por AND, ou AND por OR) e troque as bolhas de inversão nas entradas e na saída (isto é, retire-as se estiverem presentes e coloque-as se não estiverem).
2. Para interpretar a operação da porta lógica, primeiro observe qual o estado lógico, 0 ou 1, é o estado ativo para as entradas e qual é o estado ativo para a saída. Feito isso, descubra se o estado ativo da saída é produzido quando *todas* as entradas estiverem no seu estado ativo (se o símbolo AND for usado) ou quando *qualquer* uma das entradas estiver no seu estado ativo (se o símbolo OR for usado).

### EXEMPLO 3-19

Interprete os dois símbolos para a porta lógica OR.

### Solução

As respostas estão mostradas na Fig. 3-35. Observe que a palavra "qualquer" é usada quando o símbolo de operação é o símbolo OR e a palavra "todas" é usada quando utilizamos o símbolo AND.

### Questões de Revisão

1. Escreva a interpretação para a operação realizada pelo símbolo padronizado NOR na Fig. 3-33.
2. Repita a questão 1 para o símbolo alternativo da porta NOR.
3. Repita a questão 1 para o símbolo alternativo da porta AND.
4. Repita a questão 1 para o símbolo padronizado da porta AND.

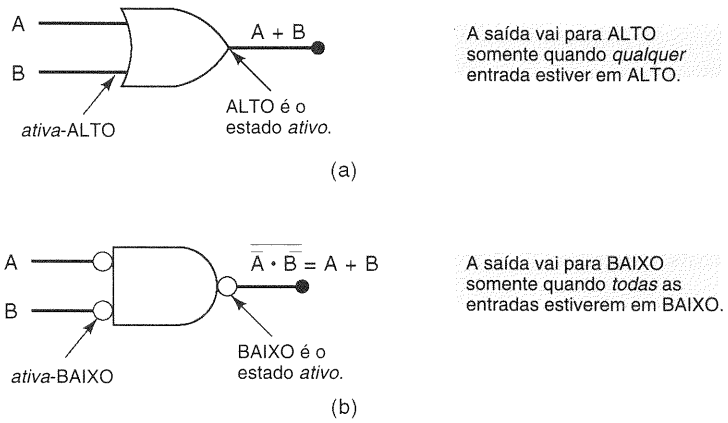



**Fig. 3-35** Interpretação dos dois símbolos das portas OR.

## 3-14 QUE REPRESENTAÇÃO DE PORTA LÓGICA USAR

Alguns projetistas de circuitos lógicos e alguns livros utilizam apenas os símbolos padronizados para as portas lógicas nos esquemáticos de seus circuitos. Apesar de esta prática não estar incorreta, ela não torna a operação do circuito mais fácil de acompanhar. A utilização adequada dos símbolos alternativos para as portas nos diagramas de circuitos pode tornar a operação do circuito bem mais clara. Isto pode ser ilustrado considerando-se o exemplo a seguir da Fig. 3-36.

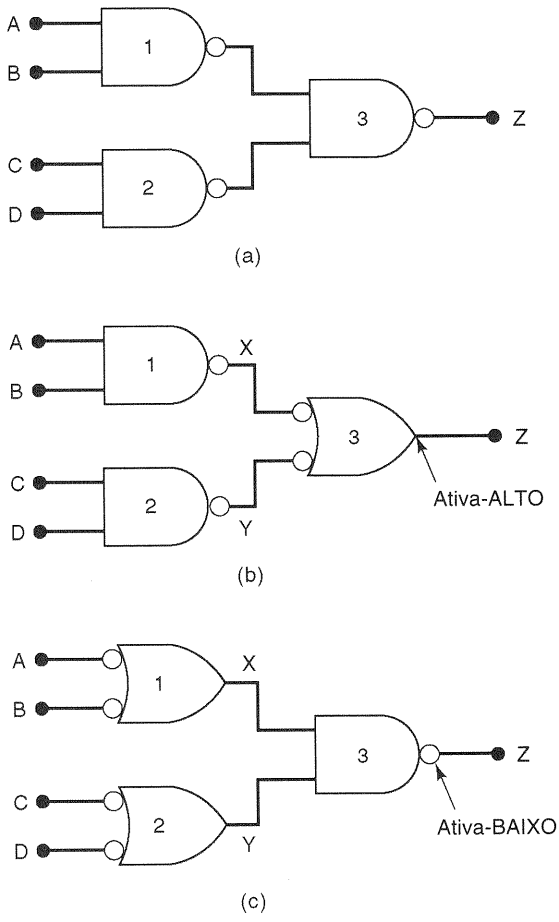
O circuito da Fig. 3-36(a) contém três portas NAND conectadas para produzir uma saída $Z$ que depende das entradas $A$, $B$, $C$ e $D$. O diagrama do circuito utiliza o símbolo padrão para cada porta NAND. Mesmo esse diagrama estando logicamente correto, ele não facilita no entendimento de como o circuito funciona. As representações do circuito apresentadas nas Fig. 3-36(b) e (c), no entanto, podem ser analisadas mais facilmente para determinar a operação do circuito.

A representação da Fig. 3-36(b) é obtida do diagrama do circuito original substituindo-se a porta NAND 3 pelo seu símbolo alternativo. Nesse diagrama, a saída $Z$ é gerada de um símbolo de porta NAND que tem uma saída ativa em ALTO. Desse modo, podemos dizer que $Z$ vai para ALTO quando ou $X$ ou $Y$ for BAIXO. Agora, já que $X$ e $Y$ aparecem na saída de símbolos NAND, que possuem saídas ativas em BAIXO, podemos dizer que $X$ vai para BAIXO somente se $A = B = 1$ e $Y$ vai para BAIXO somente se $C = D = 1$. Em resumo, podemos descrever a operação do circuito do seguinte modo:

**A saída $Z$ vai para ALTO sempre que ou $A = B = 1$ ou $C = D = 1$ (ou ambos).**

Esta descrição pode ser colocada na forma de tabela-verdade fazendo $Z = 1$ para os casos em que $A = B = 1$ e para os casos em que $C = D = 1$. Para todos os outros casos, $Z$ deve ser 0. A tabela-verdade resultante é mostrada na Fig. 3-36(d).

A representação da Fig. 3-36(c) é obtida do diagrama do circuito original substituindo-se as portas NAND 1 e 2 pelos seus símbolos alternativos. Nesta representação equivalente, a saída $Z$ é gerada de uma porta NAND que tem uma saída ativa-BAIXO. Portanto, podemos dizer que $Z$ vai para BAIXO somente quando $X = Y = 1$. Como $X$ e $Y$ são saídas ativas em ALTO, podemos dizer que $X$ será ALTO quando ou $A$ ou $B$ for BAIXO e $Y$ será ALTO quando ou $C$ ou $D$ for




| A | B | C | D | Z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 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 |

(d)

**Fig. 3-36** (a) Circuito original utilizando símbolos padrões NAND; (b) representação equivalente onde a saída $Z$ está ativa-ALTO; (c) representação equivalente onde a saída $Z$ está ativa-BAIXO; (d) tabela-verdade.

BAIXO. Resumindo, podemos descrever a operação do circuito do seguinte modo:

**A saída *Z* vai para BAIXO somente quando *A* ou *B* for BAIXO e *C* ou *D* for BAIXO.**

Esta descrição pode ser colocada na forma de tabela-verdade tornando $Z = 0$ para todos os casos em que pelo menos uma das entradas *A* ou *B* está BAIXA, ao mesmo tempo em que pelo menos uma das entradas *C* ou *D* está BAIXA. Para todos os outros casos, *Z* deve ser 1. A tabela-verdade resultante é a mesma que foi obtida do diagrama do circuito da Fig. 3-36(b).

## Que Diagrama de Circuito Deve Ser Usado?

A resposta para esta pergunta depende da função específica sendo realizada pela saída do circuito. Se o circuito está sendo usado para causar alguma ação (por exemplo: ligar um LED — *Light Emitting Diode* — ou ativar um outro circuito lógico) quando a saída *Z* vai para o estado 1, então dizemos que a saída *Z* deve ser ativa-ALTO, e o diagrama de circuito da Fig. 3-36(b) deveria ser usado. Por outro lado, se o circuito está sendo usado para causar alguma ação quando *Z* vai para o estado 0, então Z deve ser ativa-BAIXO, e o diagrama de circuito da Fig. 3-36(c) deveria ser usado.

Naturalmente existem situações em que *ambos* os estados de saída são usados para produzir diferentes ações, e qualquer um pode ser considerado o estado ativo. Para esses casos, qualquer representação de circuito pode ser usada.

## Colocação da Bolha

Veja a representação do circuito da Fig. 3-36(b) e note que os símbolos para as portas NAND 1 e 2 foram escolhidos para terem saídas ativas em BAIXO, para combinar com as entradas ativas em BAIXO da porta NAND 3. Veja a representação do circuito da Fig. 3-36(c) e note que os símbolos para as portas NAND 1 e 2 foram escolhidos para terem saídas ativas em ALTO, para combinar com as entradas ativas em ALTO da porta NAND 3. Isto leva para a seguinte regra geral na preparação de esquemáticos de circuitos lógicos:

**Sempre que possível, escolha símbolos para as portas tais que saídas com bolha sejam conectadas em entradas com bolha, e saídas sem bolha em entradas sem bolha.**

Os exemplos a seguir mostram como essa regra pode ser aplicada.

### EXEMPLO 3-20

O circuito lógico na Fig. 3-37(a) está sendo usado para ativar um alarme quando sua saída *Z* vai para ALTO. Modifique o diagrama do circuito de modo que ele represente mais eficientemente a operação do circuito.

#### Solução

Já que $Z = 1$ ativará o alarme, *Z* deve ser ativa-ALTO. Logo, o símbolo da porta AND 2 não deve ser mudado. O símbolo da porta NOR deveria ser substituído pelo símbolo alternativo sem bolha na saída (ativa em ALTO), para combinar com a entrada sem bolha da porta AND 2, conforme mostrado na Fig. 3-37(b). Note que o circuito agora tem saídas sem bolha conectadas nas entradas sem bolha da porta 2.

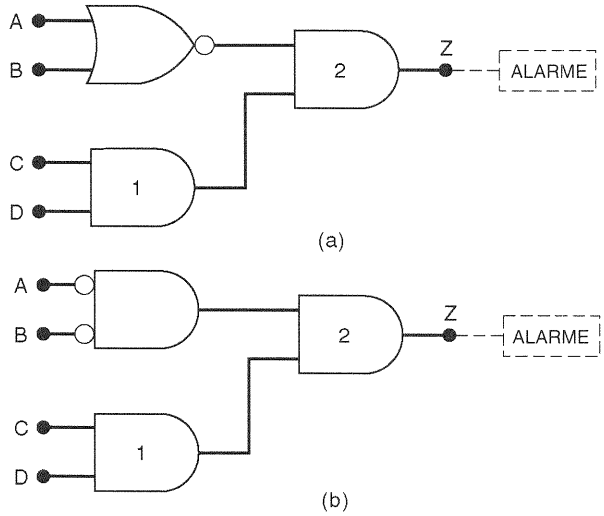



**Fig. 3-37** Exemplo 3-20.

### EXEMPLO 3-21

Quando a saída do circuito lógico na Fig. 3-38(a) vai para BAIXO, ela aciona um outro circuito lógico. Modifique o diagrama do circuito para representar mais eficientemente a operação do circuito.

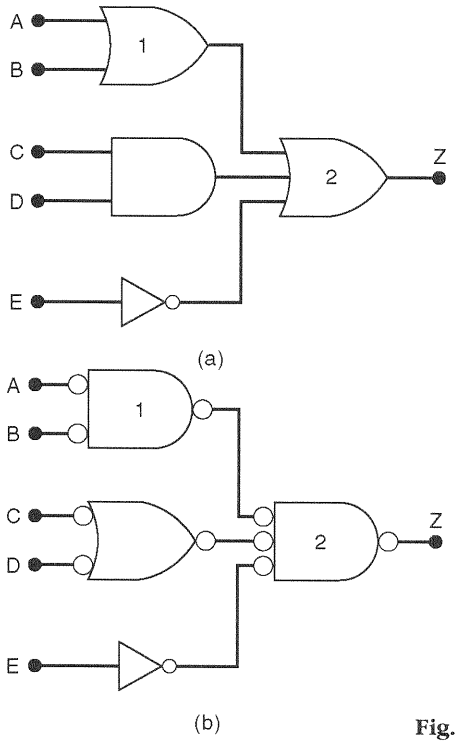



**Fig. 3-38** Exemplo 3-21.

**Solução**

Já que *Z* deve ser ativa-BAIXO, o símbolo para a porta OR 2 deve ser mudado para o símbolo alternativo, como mostra a Fig. 3-38(b). O novo símbolo da porta OR 2 tem entradas com bolha, e portanto os símbolos da porta AND e da porta OR 1 devem ser trocados para terem saídas com bolhas, conforme mostra a Fig. 3-38(b). O INVERSOR já possui saída com bolha. Agora o circuito tem todas as saídas com bolha conectadas nas entradas com bolha da porta 2.

## Analisando Circuitos

Quando um esquemático de circuito lógico é desenhado usando as regras que utilizamos nesses exemplos, é bem mais fácil para um engenheiro ou técnico (ou estudante) acompanhar o fluxo do sinal através do circuito e determinar as condições de entrada que são necessárias para ativar a saída. Isto será ilustrado nos próximos exemplos, que "por acaso" usam diagramas de circuitos obtidos de esquemáticos de um microcomputador real.

### EXEMPLO 3-22

O circuito lógico na Fig. 3-39 gera uma saída *MEM*, que é usada para ativar os CIs de memória de um microcomputador. Determine as condições de entrada necessárias para ativar *MEM*.

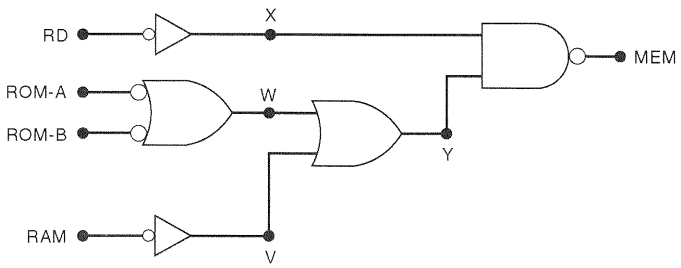



**Fig. 3-39** Exemplo 3-22.

**Solução**

Uma maneira de fazer isso é escrever a expressão para *MEM* em termos das entradas *RD*, *ROM-A*, *ROM-B* e *RAM* e avaliá-la para as 16 combinações possíveis destas entradas. Apesar de esse método funcionar, ele demanda muito mais trabalho do que seria necessário.

Um método mais eficiente é interpretar o diagrama do circuito usando as idéias que desenvolvemos nas duas últimas seções. Os passos são os seguintes:

1. *MEM* é ativa-BAIXO, e fica BAIXO somente quando *X* e *Y* estão em ALTO.
2. *X* fica ALTO somente quando *RD* = 0.
3. *Y* fica ALTO quando ou *W* ou *V* está em ALTO.
4. *V* fica ALTO quando *RAM* = 0.
5. *W* fica ALTO quando ou *ROM-A* ou *ROM-B* = 0.
6. Em resumo, *MEM* vai para BAIXO somente quando *RD* = 0 e pelo menos uma das três entradas *ROM-A*, *ROM-B* ou *RAM* estiver em BAIXO.

### EXEMPLO 3-23

O circuito lógico na Fig. 3-40 é usado para controlar o motor de uma unidade de disco quando o microcomputador está enviando ou recebendo dados do disco. O circuito ligará o motor quando *DRIVE* = 1. Determine as condições de entrada necessárias para ligar o motor.

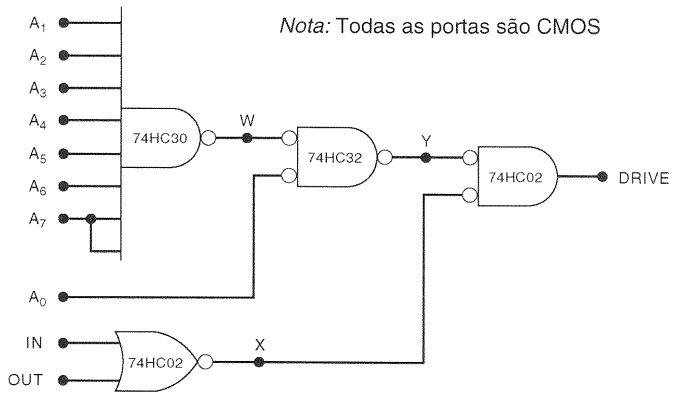



**Fig. 3-40** Exemplo 3-23.

**Solução**

Mais uma vez, interpretaremos o diagrama passo a passo:

1. *DRIVE* é ativa-ALTO, e vai para ALTO somente quando $X = Y = 0$.
2. *X* fica BAIXO quando ou *IN* ou *OUT* está em ALTO.
3. *Y* fica BAIXO somente quando $W = 0$ e $A_0 = 0$.
4. *W* fica BAIXO somente quando $A_1$ até $A_7$ estiverem em ALTO.
5. Em resumo, *DRIVE* fica ALTO quando $A_1 = A_2 = A_3 = A_4 = A_5 = A_6 = A_7 = 1$ e $A_0 = 0$, e ou *IN* ou *OUT* ou ambos forem 1.

Note o símbolo diferente para a porta NAND CMOS de 8 entradas (74HC30); repare também que o sinal $A_7$ está conectado em duas entradas da NAND.

## Níveis de Acionamento

Descrevemos sinais lógicos como estando ativos em BAIXO ou ativos em ALTO. Por exemplo, a saída *MEM* na Fig. 3-39 é ativa-BAIXO, e a saída *DRIVE* na Fig. 3-40 é ativa-ALTO, tendo em vista que esses estados de saída fazem algo acontecer. Do mesmo modo, a Fig. 3-40 tem as entradas de $A_1$ até $A_7$ ativas em ALTO, e a entrada $A_0$ ativa em BAIXO.

Quando um sinal lógico está em seu estado ativo, pode-se dizer que ele está **acionado**. Por exemplo, quando dizemos que a entrada $A_0$ está acionada, estamos di-

zendo que ela está no seu estado ativo-BAIXO. Quando um sinal lógico não está no seu estado ativo, diz-se estar **não-acionado**. Portanto, quando dizemos que *DRIVE* está não-acionado, significa que ele está no seu estado inativo (BAIXO).

É claro que os termos "acionado" e "não-acionado" são sinônimos de "ativo" e "inativo", respectivamente:

**acionado = ativo**
**não-acionado = inativo**

Ambos os termos são de uso comum na área digital, portanto você deve reconhecer os dois modos de descrever o estado ativo de um sinal lógico.

### Identificando Sinais Lógicos Ativos em BAIXO

Tornou-se prática comum usar uma barra sobreposta para identificar sinais ativos em BAIXO. A barra serve como outra indicação de que o sinal é ativo em BAIXO, e é claro que a ausência da barra significa que o sinal é ativo em ALTO.

Para ilustrar, todos os sinais na Fig. 3-39 são ativos em BAIXO e portanto podem ser identificados como segue:

$\overline{RD}$, $\overline{ROM\text{-}A}$, $\overline{ROM\text{-}B}$, $\overline{RAM}$, $\overline{MEM}$

Lembre-se, a barra é simplesmente um modo de enfatizar que esses sinais são ativos em BAIXO. Empregaremos essa convenção para identificação de sinais lógicos sempre que for apropriado.

### Identificando Sinais de Dois Estados

Freqüentemente, um sinal de saída tem dois estados ativos, isto é, ele tem uma função importante no estado ALTO e uma outra no estado BAIXO. É usual identificar tais sinais de modo que ambos os estados ativos sejam aparentes. Um exemplo comum é o sinal de leitura/escrita $RD/\overline{WR}$, [do inglês, *read/write*], que é interpretado como segue: quando este sinal está em ALTO, a operação de leitura (*RD*) é realizada; quando está em BAIXO, a operação de escrita ($\overline{WR}$) é realizada.

**Questões de Revisão**

1. Use o método dos Exemplos 3-22 e 3-23 para determinar as condições de entrada necessárias para ativar a saída do circuito na Fig. 3-37(b).
2. Repita a questão 1 para o circuito da Fig. 3-38(b).
3. Quantas portas NAND existem na Fig. 3-39?
4. Quantas portas NOR existem na Fig. 3-40?
5. Qual será o nível de saída na Fig. 3-38(b) quando todas as entradas estiverem acionadas?
6. Que entradas são necessárias para acionar a saída de alarme na Fig. 3-37(b)?
7. Quais dos seguintes sinais são ativos em BAIXO: *RD*, $\overline{W}$ e $R/\overline{W}$?

## 3-15 SÍMBOLOS LÓGICOS DO PADRÃO IEEE/ANSI

Os símbolos lógicos que usamos em todo este capítulo são símbolos padronizados e bem conhecidos amplamente utilizados na indústria digital há muitos anos. Estes símbolos representam bem as portas lógicas básicas porque cada símbolo de porta tem uma forma característica, e cada entrada tem a mesma função. Eles não fornecem informação útil suficiente, no entanto, para dispositivos lógicos mais complexos tais como: flip-flops, contadores, decodificadores, multiplexadores, memórias e CIs de interface de microprocessadores. Estes CIs complexos freqüentemente têm várias entradas e saídas com diferentes funções e modos de operação.

Para fornecer informação mais útil sobre esses dispositivos lógicos complexos, um novo conjunto de símbolos padronizados foi introduzido em 1984 pelo Padrão IEEE/ANSI 91-1984. Esses novos **símbolos IEEE/ANSI** estão sendo, paulatinamente, aceitos por mais e mais companhias de eletrônica e fabricantes de CIs e já começam a aparecer nas suas literaturas técnicas. Além disso, os contratos militares nos Estados Unidos agora exigem a utilização desses novos símbolos. Por conseguinte, é importante ficar familiarizado com esses novos símbolos padronizados.

A principal diferença no padrão IEEE/ANSI é que ele utiliza símbolos retangulares para todos os dispositivos, em vez de formas diferentes de símbolos para cada dispositivo. Um sistema especial de *notação de dependência* é usado para indicar como as saídas *dependem* das entradas. A Fig. 3-41 mostra os novos símbolos retangulares lado a lado com

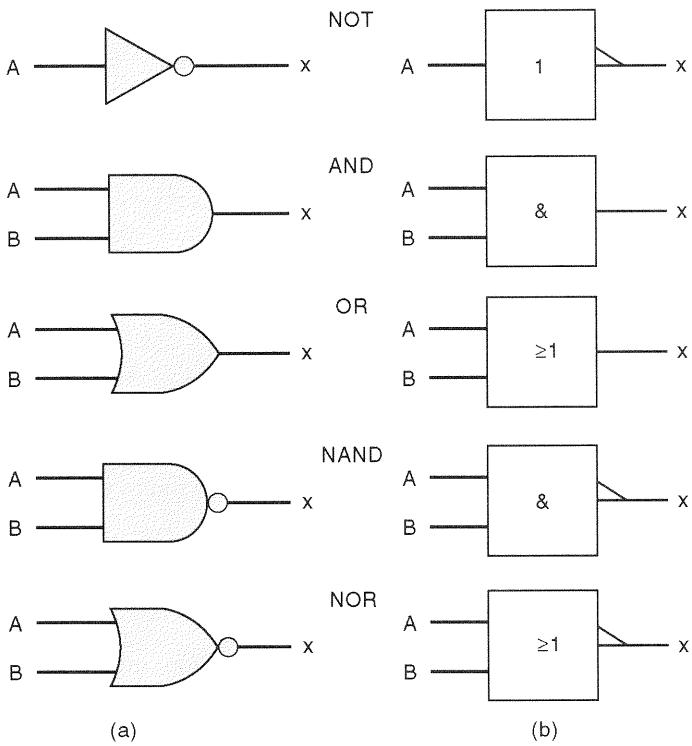



**Fig. 3-41** Símbolos lógicos padronizados: (a) tradicionais; (b) retangulares.

os símbolos tradicionais para as portas básicas. Estude-os cuidadosamente e note os seguintes pontos:

1. Os símbolos retangulares usam um pequeno triângulo no lugar da pequena bolha dos símbolos tradicionais para indicar inversão do nível lógico. A presença ou ausência do triângulo também indica se uma entrada ou saída é ativa-BAIXO ou ativa-ALTO.
2. Uma notação especial dentro de cada símbolo retangular descreve a relação lógica entre entradas e saída. O "1" dentro do símbolo INVERSOR indica um dispositivo com somente *uma* entrada; o triângulo na saída significa que a saída vai para o seu estado ativo-BAIXO quando aquela única entrada estiver no seu estado ativo-ALTO. O "&" dentro do símbolo AND significa que a saída vai para o seu estado ativo-ALTO quando todas as entradas estiverem em seus estados ativos em ALTO. O "≥" dentro da porta OR significa que a saída vai para o seu estado ativo (ALTO) sempre que *uma ou mais* entradas estiverem no seu estado ativo (ALTO).
3. Os símbolos retangulares para NAND e NOR são os mesmos de AND e OR, respectivamente, apenas adicionando-se o pequeno triângulo de inversão na saída.

## Símbolos IEEE/ANSI para CIs de Portas Lógicas

Os símbolos retangulares também podem ser usados para representar a lógica completa de um CI que contenha um certo número de portas independentes. Isto está ilustrado na Fig. 3-42 para o CI TTL INVERSOR sêxtuplo 74LS04, e na Fig. 3-43 para o CI CMOS NAND duplo de quatro entradas 74HC20. Cada porta lógica é representada como um bloco retangular separado. Repare que os símbolos retangulares indicam a notação da operação lógica apenas no bloco superior; entende-se que deve ser aplicada para todos os blocos representativos das outras portas no chip.

É importante compreender a diferença entre os dois modos alternativos para representar uma porta lógica num

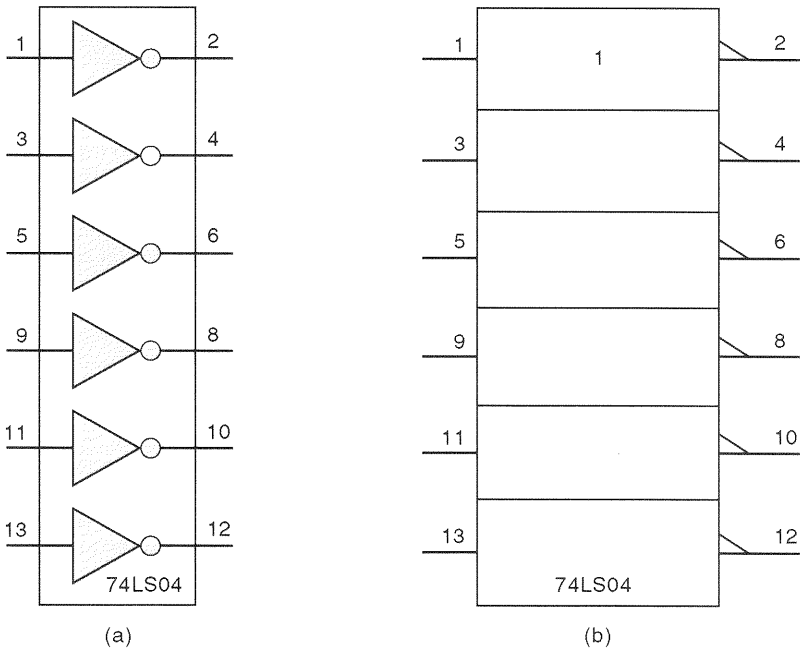



**Fig. 3-42** CI INVERSOR sêxtuplo 74LS04: (a) símbolo lógico tradicional; (b) símbolo lógico retangular. A notação "1" aparece apenas no retângulo superior mas aplica-se a todos os outros blocos.

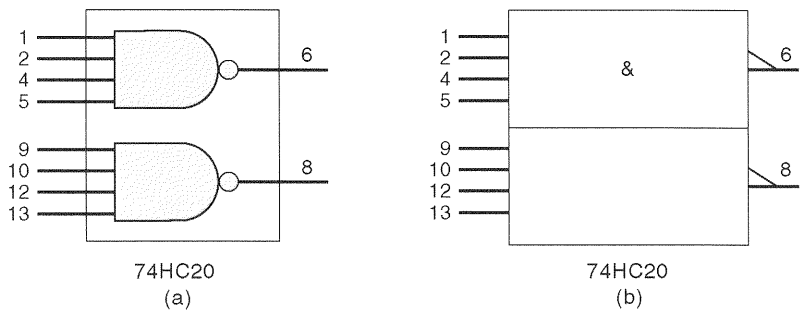



**Fig. 3-43** CI NAND duplo de quatro entradas 74HC20: (a) símbolo tradicional; (b) símbolo retangular.

circuito e a diferença entre os dois padrões diferentes de símbolos para portas lógicas. Escolhe-se qual conjunto de símbolos padronizados usar — ou os símbolos tradicionais (formas características para cada tipo de porta) ou o novo padrão de símbolos retangulares. Independentemente do símbolo escolhido para utilização, existem duas maneiras possíveis de representar uma porta num diagrama de circuito dependendo do seu estado de saída ativo. Isto está ilustrado no Exemplo 3-24.

**EXEMPLO 3-24**

A Fig. 3-44(a) mostra as duas representações para uma porta NOR usando os símbolos lógicos tradicionais. Lembre-se, a escolha de que representação usar num diagrama de circuito é determinada pelo estado ativo desejado para a saída. Desenhe novamente as duas representações utilizando os novos símbolos IEEE/ANSI.

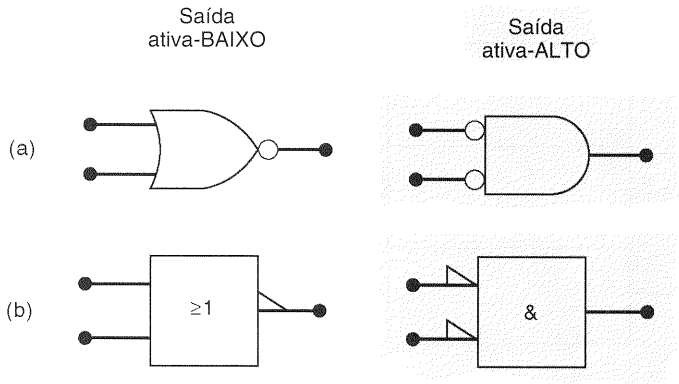



**Fig. 3-44** Ambas as representações de uma porta NOR usando os dois tipos de símbolos: (a) tradicional; (b) retangular.

**Solução**

A Fig. 3-44(b) apresenta os resultados.

## Símbolos IEEE/ANSI para CIs Complexos

Não haveria qualquer vantagem real para os novos símbolos se tivéssemos apenas que lidar com as portas lógicas básicas. Para os dispositivos lógicos mais complexos, entretanto, os novos símbolos padronizados com sua *notação de dependência* especificam a operação lógica completa do dispositivo. Isto torna praticamente desnecessário consultar o manual do fabricante para descobrir como um determinado CI lógico está funcionando em um circuito. Veremos exemplos disso quando encontrarmos os circuitos lógicos mais complexos em capítulos posteriores.

Os símbolos lógicos tradicionais são empregados na maioria dos diagramas de circuito em todo este livro, e os símbolos IEEE/ANSI são usados apenas ocasionalmente. Alguns problemas do final do capítulo necessitam de análise e construção de circuitos utilizando a notação mais nova. Além disso, sempre que um novo tipo de dispositivo lógico ou circuito for introduzido, ambos os tipos de símbolos serão apresentados. Desse modo, você ficará familiarizado com a notação de dependência que é a principal vantagem do novo padrão.

**Questões de Revisão**

1. Qual é a maior vantagem dos novos símbolos IEEE/ANSI?
2. Desenhe todas as portas lógicas básicas usando tanto os símbolos tradicionais quanto os símbolos padronizados IEEE/ANSI.
3. Repita a questão 2 para a representação alternativa de cada porta.

# RESUMO

1. A álgebra booleana é uma ferramenta matemática usada em análise e projeto de circuitos digitais.
2. As operações booleanas básicas são as operações OR, AND e NOT.
3. Uma porta OR produz uma saída em ALTO quando qualquer entrada está em ALTO. Uma porta AND produz uma saída em ALTO somente quando todas as entradas estão em ALTO. Um circuito NOT (INVERSOR) produz uma saída que é o nível lógico oposto ao da entrada.
4. Uma porta NOR é o mesmo que uma porta OR com a saída conectada a um INVERSOR. Uma porta NAND é o mesmo que uma porta AND com a saída conectada a um INVERSOR.
5. Teoremas e regras booleanas podem ser usados para simplificar a expressão de um circuito lógico e podem levar a um modo mais simples de implementar o circuito.
6. Portas NAND podem ser usadas para implementar qualquer das operações booleanas básicas. Portas NOR podem ser usadas com o mesmo objetivo.
7. Tanto os símbolos padronizados quanto os alternativos podem ser usados para cada porta lógica, dependendo se a saída deve estar ativa-ALTO ou ativa-BAIXO.
8. O padrão IEEE/ANSI para símbolos lógicos utiliza símbolos retangulares para cada dispositivo lógico, com notações especiais dentro dos retângulos para mostrar como as saídas dependem das entradas.

# TERMOS IMPORTANTES*

álgebra booleana
nível lógico
tabela-verdade
operação OR
porta OR
operação AND
porta AND
operação NOT
inversão-complemento
circuito NOT (INVERSOR)

*Esses termos podem ser encontrados em negrito no capítulo e estão definidos no Glossário ao final do livro.

porta NOR
porta NAND
teoremas booleanos
teoremas de DeMorgan
símbolos lógicos alternativos
níveis lógicos ativos
ativa(o)-ALTO
ativa(o)-BAIXO
acionado
não-acionado
símbolos IEEE/ANSI

# PROBLEMAS

As letras em negrito que precedem alguns problemas são usadas para indicar a natureza ou tipo de problema como segue:

**C** (do inglês, *challenging*) problema desafiador
**T** (do inglês, *troubleshooting*) problema de depuração
**D** (do inglês, *design*) problema de projeto ou modificação de circuito
**N** (do inglês, *new concept*) novo conceito ou técnica não abordada no texto

## SEÇÃO 3-3

**3-1.** Desenhe a forma de onda de saída para o circuito da Fig. 3-45.

**3-2.** Suponha que a entrada *A* na Fig. 3-45 seja involuntariamente colocada em curto com a terra (isto é, $A = 0$). Desenhe a forma de onda resultante na saída.

**3-3.** Suponha que a entrada *A* na Fig. 3-45 seja involuntariamente colocada em curto com a fonte de +5 V (isto é, $A = 1$). Desenhe a forma de onda resultante na saída.

**3-4.** Leia as afirmações a seguir relativas a uma porta OR. Inicialmente elas podem parecer válidas, mas após alguma análise você deve perceber que nenhuma é *sempre* verdadeira. Prove isto mostrando um exemplo específico para refutar cada afirmação.

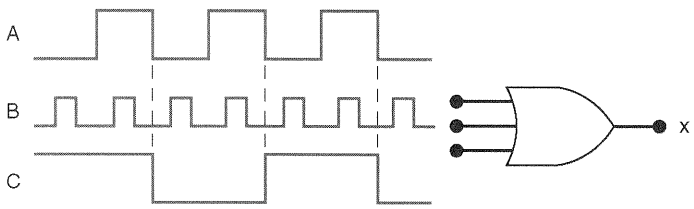



**Fig. 3-45**

**(a)** Se a forma de onda de saída de uma porta OR é a mesma forma de onda de uma das entradas, a outra entrada está sendo mantida permanentemente em BAIXO.

**(b)** Se a forma de onda de saída de uma porta OR está sempre em ALTO, uma das entradas está sendo permanentemente mantida em ALTO.

**3-5.** Quantos conjuntos diferentes de condições de entrada produzem uma saída em ALTO para uma porta OR de cinco entradas?

## SEÇÃO 3-4

**3-6.** Troque a porta OR na Fig. 3-45 por uma porta AND.

**(a)** Desenhe a forma de onda de saída.

**(b)** Desenhe a forma de onda de saída se a entrada *A* está permanentemente colocada em curto com a terra.

**(c)** Desenhe a forma de onda de saída se a entrada *A* está permanentemente colocada em curto com +5 V.

**D**

**3-7.** Consulte a Fig. 3-4. Modifique o circuito de modo que o alarme seja ativado somente quando a pressão e a temperatura excederem os seus limites máximos ao mesmo tempo.

**3-8.** Troque a porta OR na Fig. 3-6 para uma porta AND e desenhe a forma de onda de saída.

**3-9.** Suponha que você tenha uma porta desconhecida de duas entradas que é ou uma porta OR ou uma porta AND. Que combinação de níveis de entrada você deve aplicar nas entradas da porta para determinar qual é o tipo da porta?

**3-10.** *Verdadeiro ou falso:* Não importa quantas entradas tenha, uma porta AND produz uma saída em ALTO para somente uma combinação dos níveis de entrada.

## SEÇÕES 3-5 A 3-7

**3-11.** Acrescente um INVERSOR na saída da porta OR da Fig. 3-45. Desenhe a forma de onda na saída do INVERSOR.

**3-12.** **(a)** Escreva a expressão booleana para a saída *x* na Fig. 3-46(a). Determine o valor de *x* para todas as condições de entrada possíveis e relacione os valores em uma tabela-verdade.

**(b)** Repita para o circuito na Fig. 3-46(b).

**3-13.** Monte a tabela-verdade completa para o circuito da Fig. 3-15(b) determinando os níveis lógicos presentes em cada saída de porta para cada uma das 32 combinações possíveis de entrada.

**3-14.** Troque cada OR por um AND e cada AND por um OR na Fig. 3-15(b). Escreva a expressão para a saída.

**3-15.** Monte a tabela-verdade completa para o circuito da Fig. 3-16 determinando os níveis lógicos presentes em cada saída de porta para cada uma das 16 combinações possíveis de níveis de entrada.

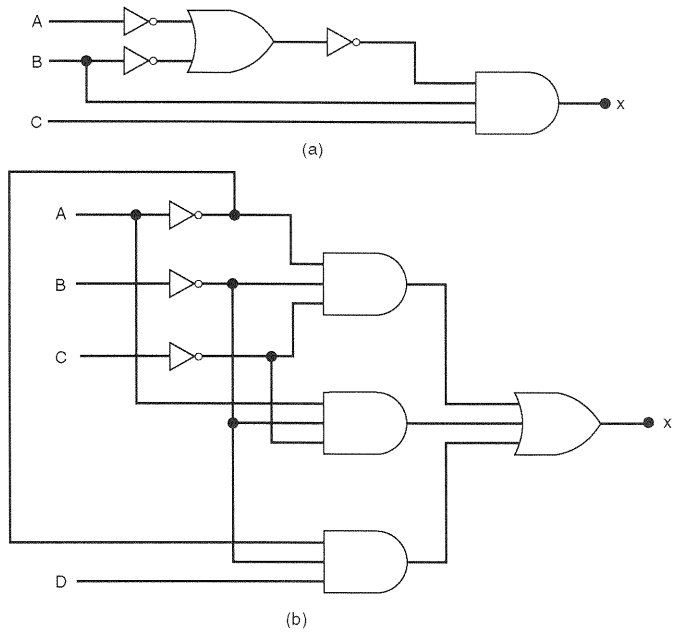



**Fig. 3-46**

## SEÇÃO 3-8

**3-16.** Para cada uma das seguintes expressões, construa o circuito lógico correspondente, usando portas AND, OR e INVERSORes.

**(a)** $x = \overline{AB(C + D)}$
**(b)** $z = \overline{(A + B + \overline{C}D\overline{E})} + \overline{B}C\overline{D}$
**(c)** $y = \overline{(M + N + \overline{P}Q)}$
**(d)** $x = \overline{W + P\overline{Q}}$
**(e)** $z = MN(P + \overline{N})$

## SEÇÃO 3-9

**3-17.** **(a)** Aplique as formas de onda de entrada da Fig. 3-47 numa porta NOR e desenhe a forma de onda de saída.
**(b)** Repita com $C$ mantido permanentemente em BAIXO.
**(c)** Repita com $C$ mantido ALTO.

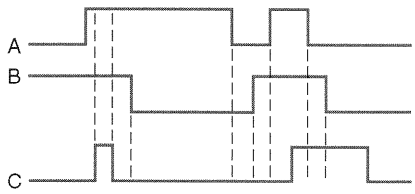



**Fig. 3-47**

**3-18.** Repita o Problema 3-17 para uma porta NAND.
**3-19.** Escreva a expressão de saída para o circuito da Fig. 3-48. Monte uma tabela-verdade completa.

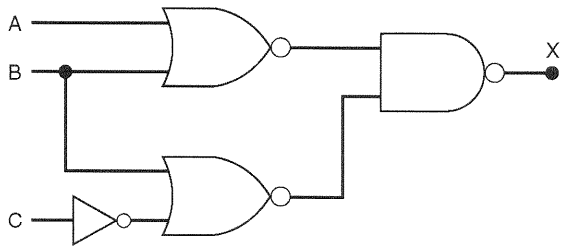



**Fig. 3-48**

**3-20.** Determine a tabela-verdade para o circuito da Fig. 3-24.
**3-21.** Modifique os circuitos que foram construídos no Problema 3-16 de modo que portas NAND e portas NOR sejam usadas sempre que for apropriado.

## SEÇÃO 3-10

**3-22.** QUESTÃO DE FIXAÇÃO
Complete cada expressão.

**(a)** $A + 1 =$ _____
**(b)** $A \cdot A =$ _____
**(c)** $B \cdot \overline{B} =$ _____
**(d)** $C + C =$ _____
**(e)** $x \cdot 0 =$ _____
**(f)** $D \cdot 1 =$ _____
**(g)** $D + 0 =$ _____
**(h)** $C + \overline{C} =$ _____
**(i)** $G + GF =$ _____
**(j)** $y + \overline{w}y =$ _____

**3-23.** **(a)** Prove o teorema (15) experimentando todos os casos possíveis.
**(b)** Prove-o usando o teorema (14) para substituir $x$.

**C**

**3-24.(a)** Simplifique a expressão seguinte usando os teoremas (13b), (3) e (4):

$$x = (M + N)(\overline{M} + P)(\overline{N} + \overline{P})$$

**(b)** Simplifique a expressão seguinte usando os teoremas (13a), (8) e (6):

$$z = \overline{A}B\overline{C} + AB\overline{C} + B\overline{C}D$$

## SEÇÕES 3-11 E 3-12

**3-25.** Prove os teoremas de DeMorgan experimentando todos os casos possíveis.
**3-26.** Simplifique cada uma das expressões seguintes utilizando os teoremas de DeMorgan.

(a) $\overline{AB\overline{C}}$
(b) $\overline{A + \overline{B}C}$
(c) $\overline{AB\overline{C}D}$
(d) $\overline{A(\overline{B + \overline{C}})D}$
(e) $\overline{(M + \overline{N})(\overline{M} + N)}$
(f) $\overline{\overline{AB}CD}$

**3-27.** Use os teoremas de DeMorgan para simplificar a expressão para a saída da Fig. 3-48.

**C**

**3-28.** Converta o circuito da Fig. 3-46(b) para outro que use apenas portas NAND. Depois escreva a expressão de saída para o novo circuito, simplifique-a usando os teoremas de DeMorgan e compare-a com a expressão para o circuito original.
**3-29.** Converta o circuito da Fig. 3-46(a) para outro que use apenas portas NOR. Depois escreva a expressão para o novo circuito, simplifique-a usando os teoremas de DeMorgan e compare-a com a expressão para o circuito original.
**3-30.** Mostre como uma porta NAND de duas entradas pode ser construída com portas NOR de duas entradas.
**3-31.** Mostre como uma porta NOR de duas entradas pode ser construída com portas NAND de duas entradas.
**3-32.** Um avião a jato emprega um sistema para monitoração dos valores de rpm, pressão e temperatura dos motores utilizando sensores que operam como segue:

saída do sensor $RPM$ = 0 somente quando a velocidade < 4800 rpm
saída do sensor $P$ = 0 somente quando a pressão < $1{,}5 \times 10^6$ N/m²
saída do sensor $T$ = 0 somente quando a temperatura < 95ºC

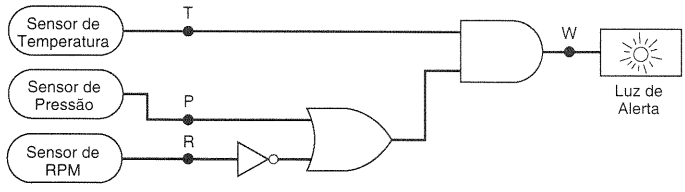



**Fig. 3-49**

A Fig. 3-49 mostra o circuito lógico que controla a luz de alerta da cabine do piloto para certas combinações das condições do motor. Suponha que um nível ALTO na saída *W* ativa a luz de alerta.

**(a)** Determine que condições do motor darão um alerta para o piloto.

**(b)** Altere o circuito para um outro que use apenas portas NAND.

## SEÇÕES 3-13 E 3-14

**3-33.** Desenhe as representações padronizadas para cada porta lógica básica. Depois desenhe as representações alternativas.

**3-34.** Para cada sentença a seguir, desenhe a representação de porta lógica apropriada e indique o tipo de porta.

**(a)** Uma saída em ALTO ocorre apenas quando todas as três entradas estão em BAIXO.

**(b)** Uma saída em BAIXO ocorre quando qualquer uma das quatro entradas está em BAIXO.

**(c)** Uma saída em BAIXO ocorre apenas quando todas as oito entradas estão em ALTO.

**3-35.** O circuito da Fig. 3-48 é uma simples tranca de combinação digital cuja saída gera um sinal ativo-BAIXO $\overline{UNLOCK}$ para apenas uma combinação das entradas.

**(a)** Modifique o diagrama do circuito de modo que ele represente mais efetivamente a operação do circuito.

**(b)** Use o novo diagrama do circuito para determinar a combinação de entrada que ativa a saída. Faça isto analisando desde a saída usando as informações dadas pelos símbolos das portas como foi feito nos Exemplos 3-22 e 3-23. Compare os resultados com a tabela-verdade obtida no Problema 3-19.

**3-36. (a)** Determine as condições de entrada necessárias para ativar a saída *Z* na Fig. 3-37(b). Faça isto analisando desde a saída como foi feito nos Exemplos 3-22 e 3-23.

**(b)** Admita que é o estado BAIXO de *Z* que ativa o alarme. Altere o diagrama do circuito para refletir isto, e depois use o diagrama revisado para determinar as condições de entrada necessárias para ativar o alarme.

**D**

**3-37.** Modifique o circuito da Fig. 3-40 de modo que $A_1 = 0$ seja necessário para produzir *DRIVE* = 1 em vez de $A_1 = 1$.

**3-38.** Determine as condições de entrada necessárias para que a saída na Fig. 3-50 vá para o seu estado ativo.

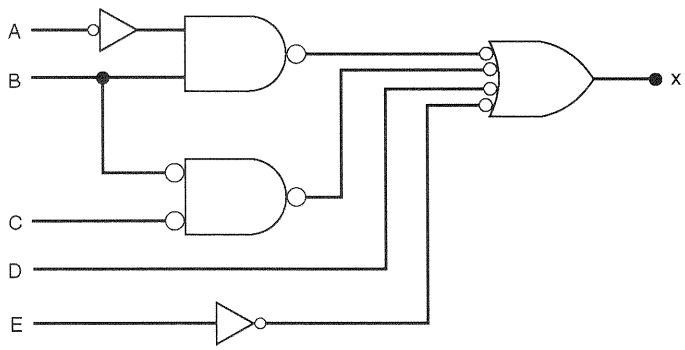



**Fig. 3-50**

**3-39.** Use os resultados do Problema 3-38 para obter a tabela-verdade completa para o circuito da Fig. 3-50.

**3-40.** Qual é o estado ativo para a saída da Fig. 3-50? E para a saída da Fig. 3-36(c)?

**3-41.** A Fig. 3-51 mostra uma aplicação de portas lógicas que simula os interruptores que usamos em nossas casas para acender e apagar uma luz de dois lugares diferentes. Aqui a luz é um LED que será LIGADO (conduzindo) quando a saída da porta NOR estiver em BAIXO. Note que esta saída é denominada $\overline{LIGHT}$ para indicar que é ativa-BAIXO. Determine as condições de entrada necessárias para ligar o LED. Depois verifique que o circuito opera como os interruptores descritos usando as chaves *A* e *B*. No Cap. 4 você aprenderá como projetar circuitos como este para produzir uma determinada relação entre entradas e saídas.

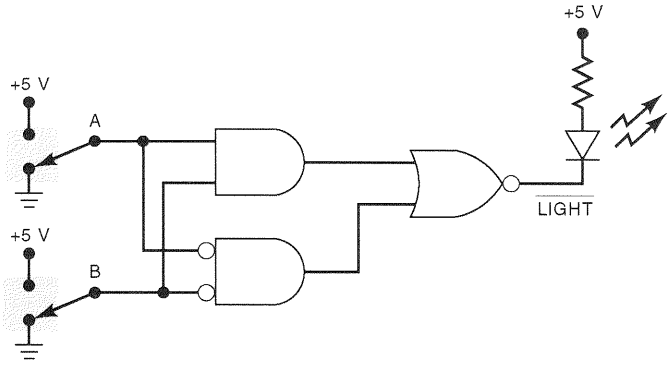



**Fig. 3-51**

## SEÇÃO 3-15

**3-42.** Desenhe os circuitos da **(a)** Fig. 3-50 e **(b)** Fig. 3-51 usando os símbolos IEEE/ANSI.

**3-43.** Determine a expressão booleana para a saída *Z* na Fig. 3-52.

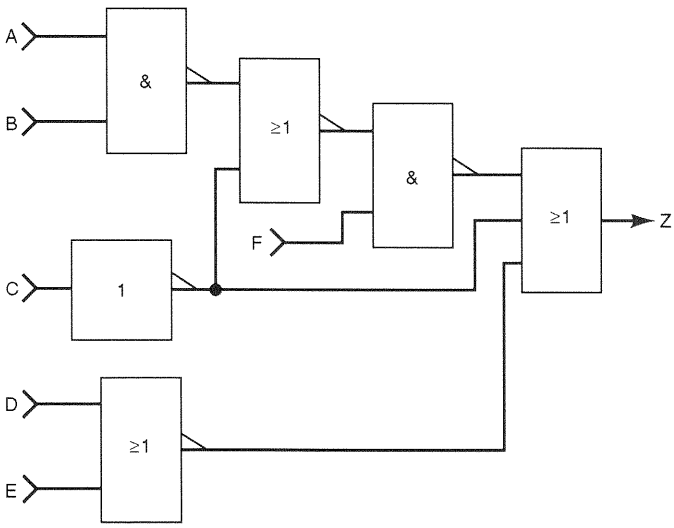



**Fig. 3-52**

**C**

**3-44.** Supõe-se que a saída do circuito da Fig. 3-52 é ativa-BAIXO. Desenhe-o para representar mais efetivamente a operação do circuito.

**C**

**3-45.** Use a versão redesenhada do circuito da Fig. 3-52 e faça o seguinte:

**(a)** Determine as várias condições de entrada que produzem um estado de saída ativa-BAIXO. Faça isto usando apenas o diagrama do circuito sem escrever a expressão para *Z* e sem gerar uma tabela-verdade completa. Os resultados deveriam ser:

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |

**(b)** Verifique que a expressão simplificada para a saída $Z$ é dada por

$$Z = \overline{ABCF(D + E)}$$

**(c)** Teste cada conjunto de condições de (a) na expressão obtida em (b) e verifique que cada uma produz $Z = 0$.

### APLICAÇÃO EM MICROCOMPUTADOR

**C**

**3-46.** Consulte a Fig. 3-40 no Exemplo 3-23. As entradas $A_7$ até $A_0$ são entradas de *endereço* que são fornecidas para esse circuito por saídas do chip do microprocessador dentro do microcomputador. O código de endereço de oito bits de $A_7$ até $A_0$ seleciona qual dispositivo o microprocessador deseja ativar. No Exemplo 3-23, o código de endereço necessário para ativar a unidade de disco é $A_7$ até $A_0 = 11111110_2 = FE_{16}$.

Modifique o circuito de modo que o microprocessador deva fornecer um código de endereço de $4A_{16}$ para ativar a unidade de disco.

### EXERCÍCIOS DESAFIADORES

**C**

**3-47.** Mostre como $x = AB\overline{C}$ pode ser implementado com uma porta NOR de duas entradas e uma porta NAND de duas entradas.

**C**

**3-48.** Implemente $y = ABCD$ usando portas NAND de duas entradas.

## RESPOSTAS PARA AS QUESTÕES DE REVISÃO DAS SEÇÕES

### SEÇÃO 3-2

**1.** $x = 1$ **2.** $x = 0$ **3.** 32

### SEÇÃO 3-3

**1.** Todas as entradas em BAIXO **2.** $x = A + B + C + D + E + F$

**3.** Constante ALTO

### SEÇÃO 3-4

**1.** Todas as cinco entradas = 1. **2.** Uma entrada em BAIXO manterá a saída em BAIXO.

**3.** Falso; veja a tabela-verdade de cada porta.

### SEÇÃO 3-5

**1.** A saída do segundo INVERSOR será a mesma que a entrada $A$.

**2.** $y$ será BAIXO somente para $A = B = 1$.

### SEÇÃO 3-6

**1.** $x = \overline{A} + B + C + \overline{AD}$

### SEÇÃO 3-7

**1.** $x = 1$ **2.** $x = 1$

### SEÇÃO 3-8

**1.** Veja a Fig. 3-15(a). **2.** Veja a Fig. 3-17(b).

**3.** Veja a Fig. 3-15(b).

### SEÇÃO 3-9

**1.** Todas as entradas em BAIXO **2.** $x = 0$

**3.** $x = \overline{A + B + \overline{CD}}$

### SEÇÃO 3-10

**1.** $y = A\overline{C}$

**2.** $y = \overline{A}\,\overline{B}\,\overline{D}$

### SEÇÃO 3-11

**1.** $z = \overline{A}\,\overline{B} + C$ **2.** $y = (\overline{R} + S + \overline{T})Q$ **3.** O mesmo que a Fig. 3-28 exceto que o NAND é trocado por NOR. **4.** $y = A\overline{B}(C + \overline{D})$

### SEÇÃO 3-12

**1.** Três **2.** O circuito NOR é mais eficiente porque pode ser implementado com um CI 74LS02. **3.** $x = (\overline{AB})(\overline{C}\,\overline{D}) = (\overline{\overline{AB}}) + (\overline{\overline{CD}}) = AB + CD$

### SEÇÃO 3-13

**1.** Saída fica BAIXO quando qualquer entrada está em ALTO.

**2.** Saída fica ALTO somente quando todas as entradas estão em BAIXO.

**3.** Saída fica BAIXO quando qualquer entrada está em BAIXO.

**4.** Saída fica ALTO somente quando todas as entradas estão em ALTO.

### SEÇÃO 3-14

**1.** $Z$ fica ALTO quando $A = B = 0$ e $C = D = 1$. **2.** $Z$ fica BAIXO quando $A = B = 0$, $E = 1$, e ou $C$ ou $D$ ou ambos são 0.

**3.** Duas **4.** Duas **5.** BAIXO **6.** $A = B = 0$, $C = D = 1$ **7.** $\overline{W}$

### SEÇÃO 3-15

**1.** Os símbolos IEEE/ANSI com sua notação de dependência especificam a operação completa do dispositivo lógico. **2.** Veja a Fig. 3-41. **3.** Veja a Fig. 3-44.