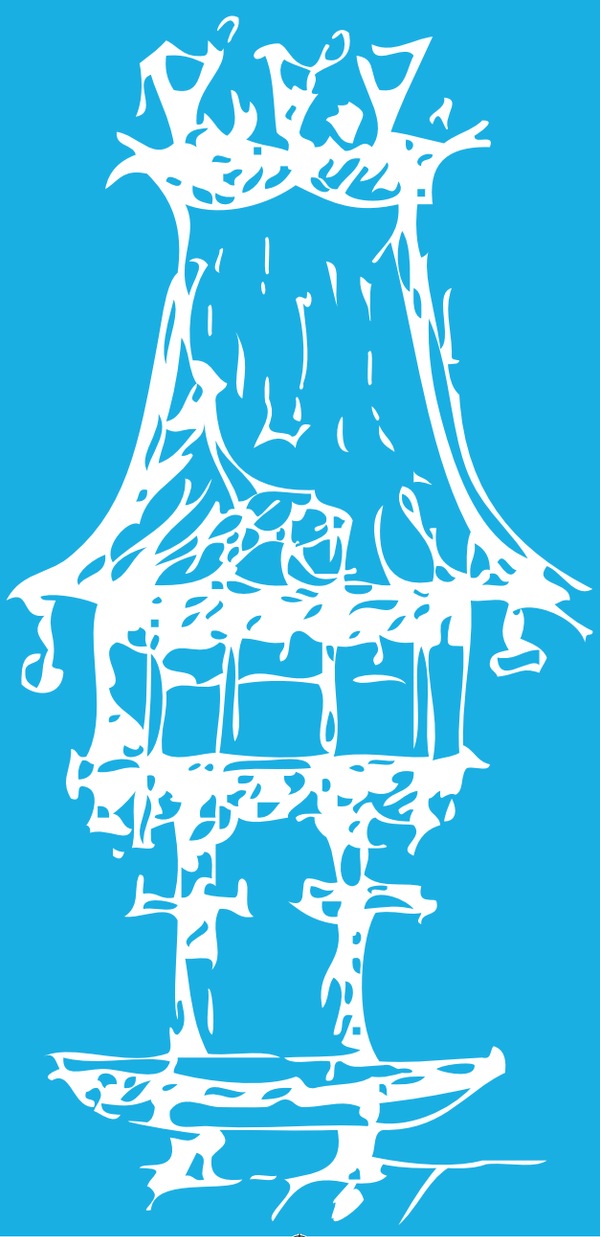
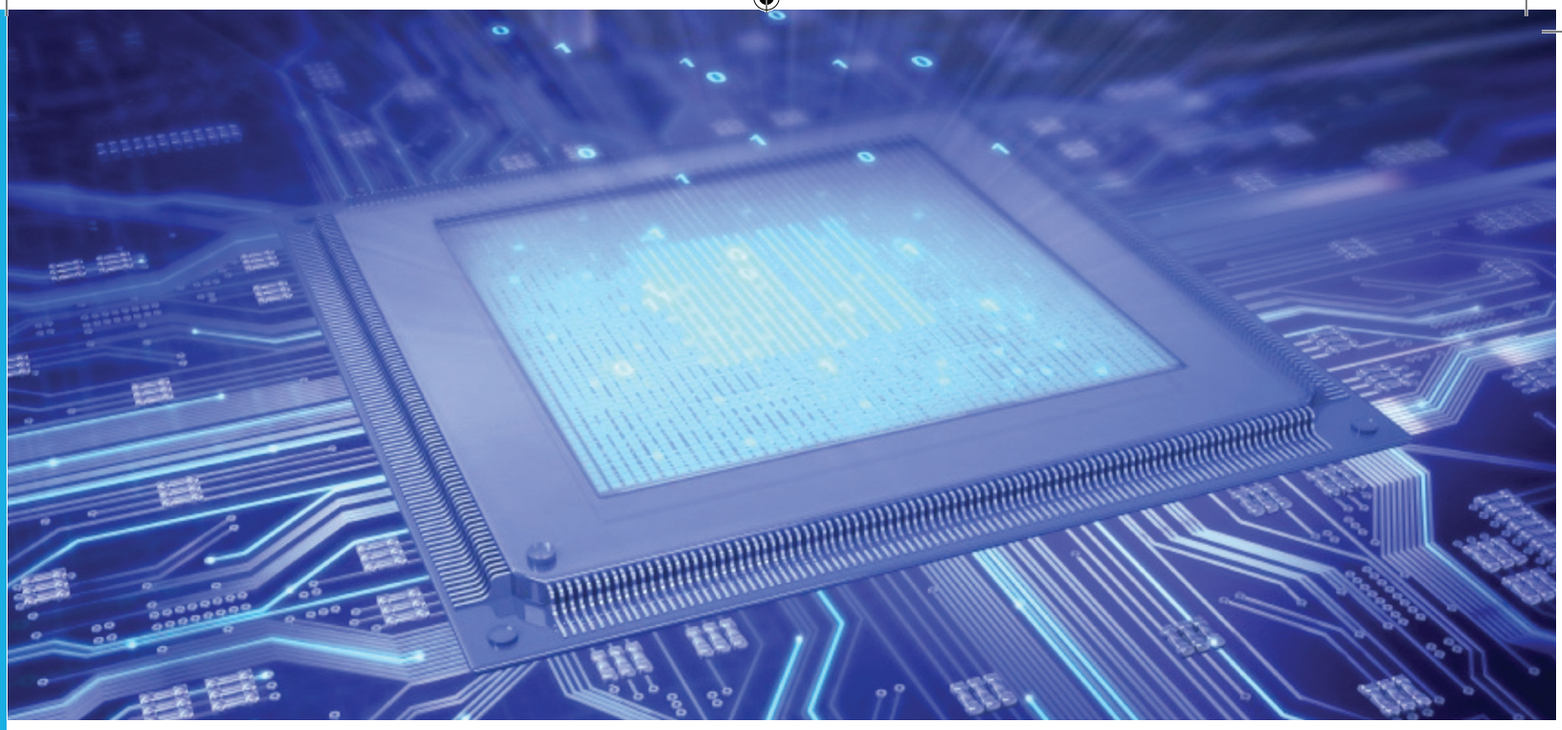




Álgebra e Lógica

Módulo 2

Caraterização do Módulo

Apresentação

Este módulo pretende desenvolver uma lógica onde se procura tratar o objeto de estudo tradicional, assente nos argumentos válidos ou corretos, através de uma forma semelhante à matemática, fazendo, para isso, uso de três caraterísticas essenciais dessa lógica:

- O uso de variáveis para representar determinadas grandezas dando, assim, uma grande generalidade e agilidade ao pensamento matemático;
- A ideia de que é possível dispor uma parte do conhecimento, na forma axiomática, onde as verdades podem ser deduzidas a partir de uma pequena lista de verdades fundamentais;
- A conceção de cálculo de um procedimento mecânico e automático para a realização de operações.

Objetivos de aprendizagem

Conhecer as principais operações lógicas, as respetivas propriedades e os teoremas da Álgebra de Boole.

Representar através de uma tabela de verdade um problema enunciado em linguagem natural.

Efetuar a simplificação de funções booleanas, usando métodos algébricos e/ou mapas de Karnaugh.

Âmbito de conteúdos

Os operadores lógicos not, and, or, xor, nand, nor, respetivas propriedades e símbolos lógicos.

Conceito de variável e função booleana.

A tabela de verdade: forma de expressar um problema em lógica. Expressões e funções booleanas.

Teoremas da Álgebra de Boole. Leis de DeMorgan.

Desenho de circuitos lógicos a partir de funções booleanas.

Simplificação algébrica de funções booleanas usando as propriedades dos operadores lógicos e os teoremas da álgebra de boole.

O mapa de Karnaugh como uma organização de espaços equivalente à tabela de verdade.

Passagem de funções booleanas na forma and-or e or-and para o mapa de Karnaugh.

Simplificação de funções, a partir do mapa de Karnaugh. Justificação do método.

Realização e experimentação prática de circuitos lógicos, usando “portas” lógicas em circuitos integrados TTL.

Portas Lógicas e Álgebra Booleana

Como mencionado, os 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 caraterística dos circuitos digitais permite-nos 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 módulo vamos estudar 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 reconstruido utilizando um menor número de portas lógicas e/ou de ligações entre estas.

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 ideia básica neste módulo e, depois, faremos uma cobertura mais completa deste tópico quando estudarmos o projeto de circuitos lógicos.

Como a álgebra booleana expressa a operação de circuitos lógicos de forma algébrica, apresenta-se como a forma ideal de descrever a operação de um circuito lógico para um programa de computador que precise 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.

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.

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, num 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 a representação de números mas, ao contrário, representam o estado do nível de tensão de uma variável, ou, como é chamado, o seu nível lógico. Diz-se que o nível de tensão num circuito digital está no nível lógico 0 ou no nível lógico 1, dependendo do seu valor numérico. Em lógica digital, vários outros termos são usados como sinónimos de 0 e 1. Alguns dos mais comuns são mostrados na Tabela 1. Vamos usar as designações 0/1 e BAIXO/ALTO na maioria das vezes.

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

Tabela 1 – Sinónimos para Níveis Lógicos

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 cubicas, 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 díodos, transístores e resistências ligadas de forma a que a saída do circuito seja o resultado da operação lógica básica (OR, AND, NOT) realizada sobre as suas entradas. Utilizaremos a álgebra, primeiramente para descrever e analisar essas portas lógicas básicas, e posteriormente para analisar e projetar combinações dessas portas lógicas ligadas como circuitos lógicos.

Tabelas de verdade

A tabela de verdade é uma maneira de descrever como a saída de um circuito lógico depende dos níveis lógicos presentes nas entradas do circuito. A Fig. 1(a), mostra a tabela de 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. 1(b) e (c) mostram exemplos de tabelas de 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 uma tabela de verdade de duas entradas, 8 linhas para uma tabela de 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 de verdade de N entradas. Note também que a lista de todas as combinações possíveis de entrada acompanha a sequência de contagem binaria, e, assim, torna-se bastante simples escrever todas as combinações possíveis sem esquecer nenhuma

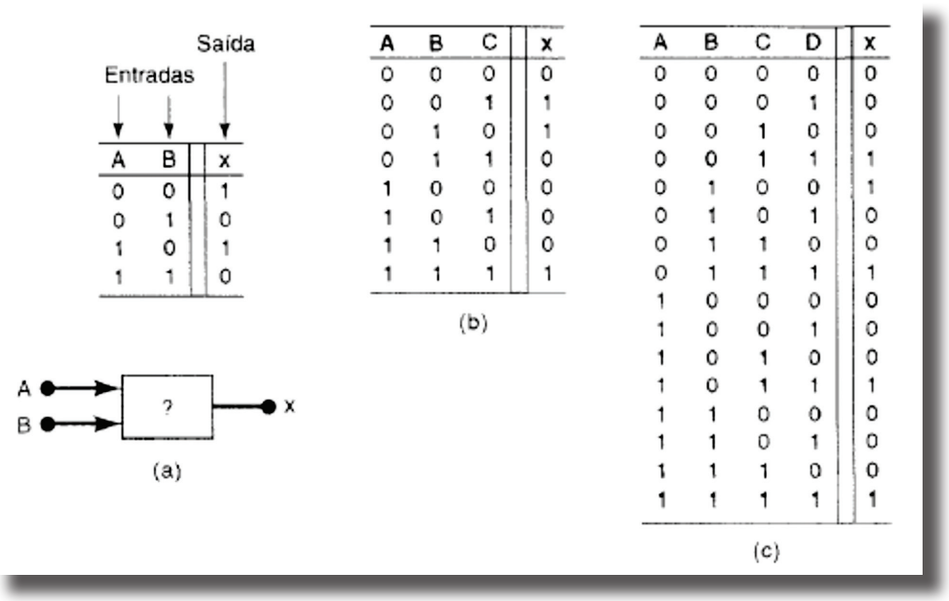


Fig. 1: (a) Exemplo de tabelas de verdade para circuitos de duas entradas, (b) de três entradas e (c) de quatro entradas.

Operação OR com portas OR

A operação OR é a primeira das três operações booleanas básicas a ser estudada. A tabela de verdade na Fig.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 e dada por:

$x = A + B$

Nesta expressão, o sinal de + não representa a operação de adição normal, mas representa a operação OR. A operação OR é semelhante à adição normal, 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 a ser 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 de verdade na Fig. 2(a), e não a operação de adição normal.

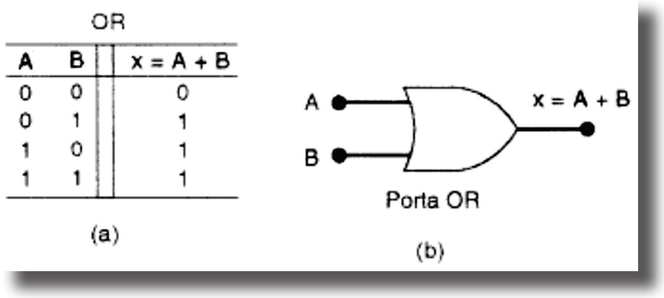


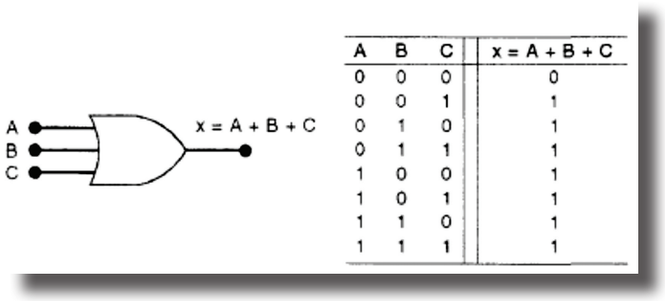
Fig. 2: (a) Tabela de verdade que define a operação OR;
(b) símbolo para uma porta OR de duas entradas.

Porta OR

Em circuitos digitais, uma porta OR é um circuito que possui duas ou mais entradas e cuja saída e igual à combinação das entradas através da operação OR. A Fig. 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, $x = A + B$. Por outras palavras, a porta OR funciona de tal modo que a 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 ideia pode ser estendida para um maior número de entradas. A Fig.3 mostra uma porta OR de 3 entradas e a sua tabela de verdade. O exame desta tabela de 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 deve-mos enfatizar que o sinal de + representa a operação OR. A saída de qualquer porta OR pode ser expressa pela combinação das entradas através da operação OR. Utilizamos esta circunstância quando estivermos a analisar circuitos lógicos.



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: Símbolo e a tabela de verdade para uma porta OR de três entradas

Resumo da Operação OR

Os pontos mais importantes a serem retidos 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 1:

Em muitos sistemas de controlo industriais é necessário ativar uma função de saída sempre que uma das várias entradas for ativada. Por exemplo, num processo químico, pode ser desejável que um alarme seja activado de cada vez que a temperatura do processo exceder um valor máximo ou sempre que a pressão estiver acima de um certo limite. A Fig.4 mostra um diagrama de blocos desta situação. O circuito transdutor de temperatura produz uma tensão proporcional à temperatura do processo. Esta tensão, VT, é comparada com uma tensão de referência de temperatura, VTR, através de um circuito comparador.

A saída do comparador está normalmente 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 respetivo comparador passa do nível baixo para nível alto quando a pressão for excessiva.

Uma vez que desejamos que o alarme seja ativado quando a temperatura (ou a pressão) estiver muito alta, podemos ligar 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 ideia pode ser obviamente estendida para situações com mais do que duas variáveis de processo.

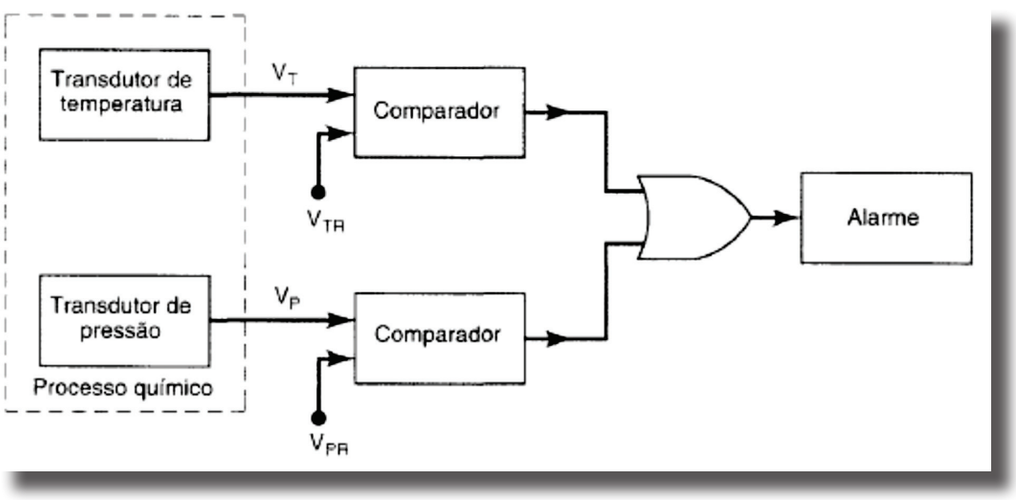


Fig. 4: Exemplo de utilização da porta OR em um sistema de alarme

Exercício 1:

Determine a saída da porta OR mostrada na Fig.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 sucessivamente.

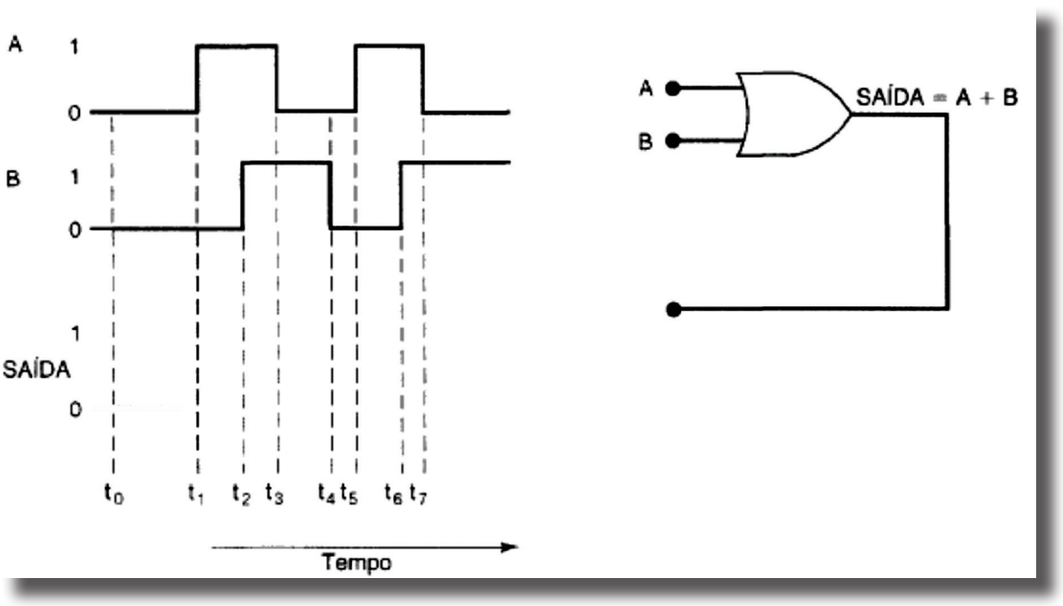


Fig. 5: Exemplo 2

Exemplo 2A:

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

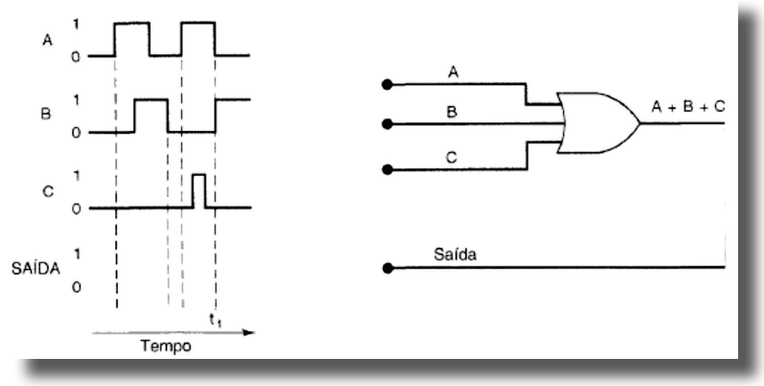


Fig. 6: Exemplos 2A e 2B

Exemplo 2B:

O que aconteceria ao glitch (pico de subida ou descida do nível lógico) mostrado na Fig.6 caso a entrada C permanecesse em nível ALTO enquanto A e B estivessem a mudar de estado em t_1?

Operação AND com portas AND

A operação AND é a segunda operação booleana básica. A tabela de verdade que aparece na Fig.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 apenas 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 normal. No entanto, a operação AND sobre variáveis booleanas opera da mesma maneira que a multiplicação normal, como pode ser visto através de um exame da tabela de 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 . B é lida como = A AND B". O sinal • é geralmente omitido de modo que a expressão se torna apenas x = AB. O mais importante a ser lembrado é que a operação AND produzirá 1 como resultado apenas quando todas as entradas (variáveis) forem iguais a 1, exatamente como na multiplicação. Este facto 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.B.C = ABC. O único momento em que x pode ser igual a 1 é quando A = B = C = 1.

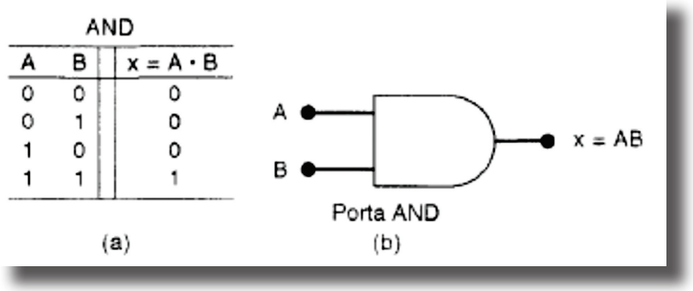


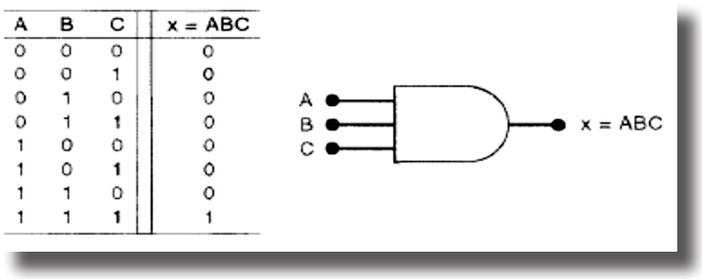
Fig. 7: (a) Tabela de verdade para a operação AND; (b) símbolo da porta AND.

Porta AND

O símbolo lógico para uma porta AND de duas entradas pode ser visto na Fig.7 (b). A saída da porta AND é igual ao produto das entradas lógicas, isto é, x = AB. Por outras palavras, a porta AND é um circuito que opera de tal maneira que a 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 de verdade correspondente podem ser vistas na Fig. 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 aparecer o símbolo de uma porta AND num diagrama de circuitos lógicos, isto diz-nos que a saída estará em ALTO apenas quando todas as entradas estiverem em ALTO. Sempre que o símbolo de uma porta for OR, isto significa que a saída estará em ALTO quando qualquer uma das entradas estiver em ALTO.



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. 8: Tabela de verdade e o símbolo para uma porta AND de três entradas.

Resumo da Operação AND

1. A operação AND é realizada exatamente do mesmo modo que a multiplicação normal de 0s e 1s.
2. A saída é igual a 1 quando todas as entradas forem iguais a 1.
3. A saída 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.

Exercício 1:

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

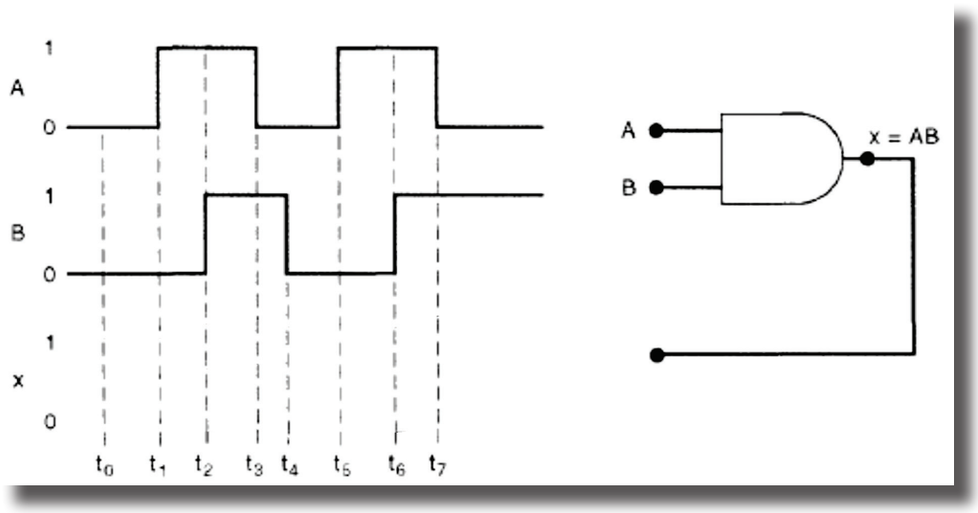


Fig. 9: Exemplo 1

Exercício 2A:

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

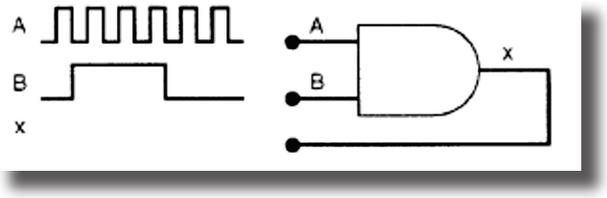


Fig. 10: Exercício 2A e 2B

Exercício 2B:

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

Operação NOT

A operação NOT é realizada, ao contrário das operações AND e OR, sobre uma única entrada. Por exemplo, se a 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 de verdade apresentada na Fig.11 (a) esclarece esta afirmação para os dois casos possíveis, A = 0 e A = 1, isto é:

$\overline{1}$ =0 ; Porque NOT 1 é 0

$\overline{0}$ =1 ; Porque NOT 0 é 1

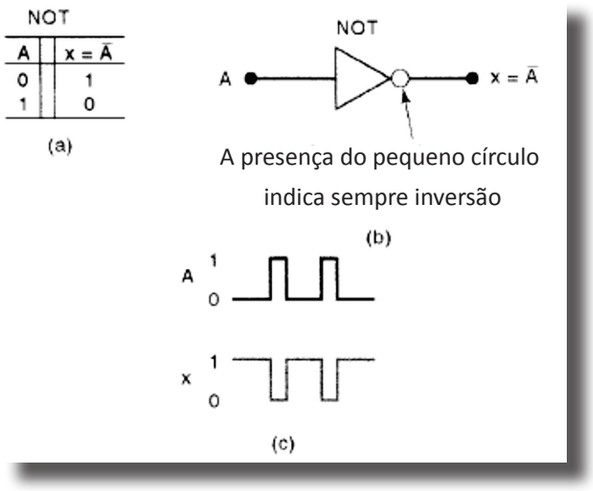


Fig. 11: (a) Tabela de verdade; (b) símbolo para o INVERSOR (NOT); (c) formas de onda.

A operação NOT é também chamada de inversão ou complemento; estes termos serão sempre usados no resto da disciplina. Apesar de utilizarmos sempre a barra sobreposta para representar inversão, é importante mencionar um outro símbolo para representar a inversão que é o apóstrofo (’), isto é:

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

Ambos os símbolos são reconhecidos como indicadores da operação de inversão

Circuito NOT (Inversor)

A Fig.11 (b) mostra o símbolo para a representação do circuito NOT, que é mais conhecido como INVERSOR.

Este circuito tem sempre uma única entrada, e o nível lógico de sua saída é sempre o oposto ao nível lógico da entrada. A Fig.11 (c) mostra como o INVERSOR atua sobre o sinal de entrada. Ele inverte (complementa) o sinal de entrada em todos os pontos da forma de onda da entrada.

Resumo das Operações Booleanas

As regras para as operações AND, OR e NOT podem ser resumidas como se mostra a seguir:

OR	AND	NOT
0 + 0 = 0	0 . 0 = 0	
0 + 1 = 1	0 . 1 = 0	$\overline{1} = 0$
1 + 0 = 1	1 . 0 = 0	$\overline{0} = 1$
1 + 1 = 1	1 . 1 = 0	

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. 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 .B. Esta saída é ligada a uma porta OR, juntamente com C que é a outra entrada do circuito. A porta OR funciona 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. B + C (esta última expressão também poderia ter sido escrita como x = C + A . B, uma vez que a ordem dos termos não importa na operação OR).

Ocasionalmente pode haver dúvida em relação à operação que deve ser realizada primeiro. A expressão A.B+C pode ser interpretada de duas maneiras:

(1) É feita a operação A . B OR C,

(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.

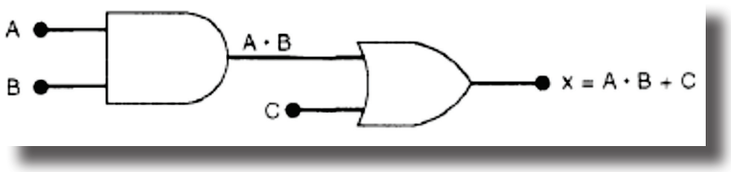


Fig. 12: Circuito lógico com a sua expressão booleana.

Segue mais um exemplo, considere o circuito da Fig.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).C. Observe o uso de parenteses para indicar que A OR B é realizada primeiro, isto é, antes que se faça um AND desta soma OR com C. Sem os parenteses, poderíamos interpretar a expressão de forma incorreta, uma vez que A + B. C significa A OR com o produto B.C.

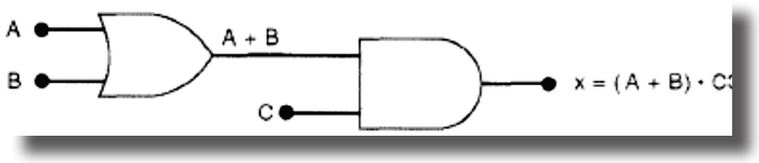


Fig. 13: Circuito lógico cuja expressão requer parenteses.

Circuitos Contendo Inversores

Sempre que um INVERSOR é apresentado num diagrama de circuitos lógicos, a expressão para a sua saída é simplesmente igual à expressão da entrada com uma barra sobre ela. A Fig.14 mostra dois exemplos usando inversores. Na Fig.14 (a), a entrada é conectada a um inversor, e a saída do mesmo é igual a $\overline{A}$. A saída do inversor é ligada 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 e primeiramente invertido e depois é feita uma operação OR com B.

Na Fig.14 (b), a saída da porta OR é igual a A + B e esta é ligada 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.

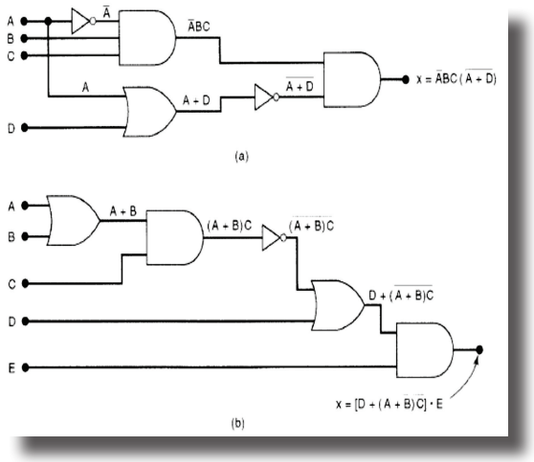


Fig. 14: Circuitos que usam inversores

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

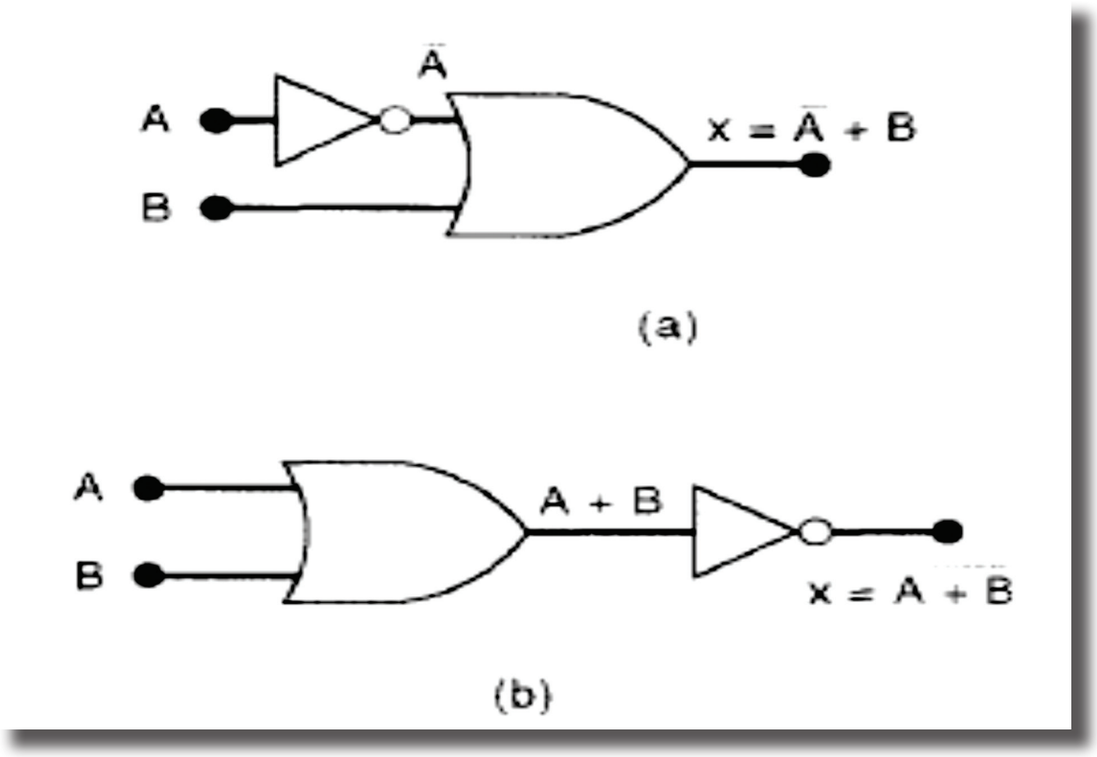


Fig. 15: Mais exemplos

Determinação do 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.15(a), para o caso em que A = 0, B = 1, C = 1 e D = 1. Como na álgebra normal, o valor de x pode ser encontrado substituindo-se os valores das variáveis na expressão e fazendo as operações como se seguem:

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

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

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

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, 0 = 1 ou 1 = 0.

2. De seguida, 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.15 para o caso em que todas as entradas são iguais a 1. As respostas são $x = 0$ e $x = 1$, respectivamente.

Determinação do 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 frequentemente usada por técnicos durante testes ou reparações de circuitos digitais, uma vez que ela mostra qual deveria ser a saída de cada porta, bem como qual deverá ser a saída final do sistema. Por exemplo, o circuito da Fig.15 (a) foi redesenhado na Fig.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 para cada inversor, ou porta, o valor de sua saída até que o valor da saída final do sistema seja encontrado.

Na Fig.16, a porta AND número 1 tem todas as suas entradas no 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 . 1 . 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 . 1 = 0.

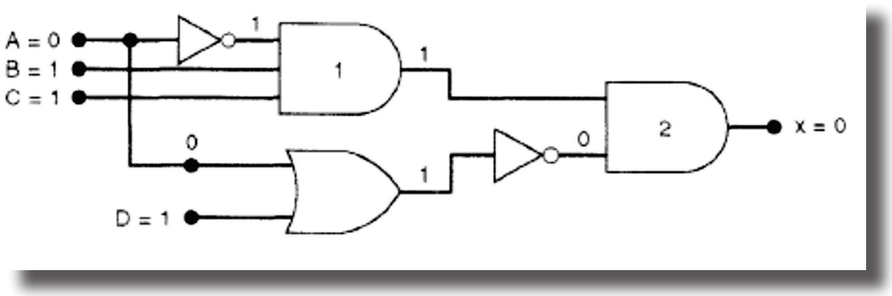


Fig. 16: Determinação o nível lógico de saída a partir do diagrama do circuito

Exercício 1:

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

Implementação de 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 definido pela expressão x = A . B . C, percebemos imediatamente que tudo de que precisamos é uma porta AND de três entradas. Se precisássemos de um circuito definido pela expressão $x = A + \overline{B}$, poderíamos usar uma porta OR de duas entradas com um inversor numa das 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 = AC + B$\overline{C}$ + $\overline{A}$. BC. Esta expressão booleana possui três termos (AC, B, B$\overline{C}$, $\overline{A}$. BC) sobre os quais é feita uma operação OR. Isto diz-nos que precisamos 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.17 (a), onde uma porta OR de três entradas está desenhada com as 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 criar cada um desses termos. Isto é mostrado na Fig.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 existam técnicas melhores e mais eficientes que podem ser empregues, como veremos mais tarde.

Por enquanto, esse método direto de implementar circuitos lógicos deve ser utilizado para diminuir o número de coisas novas que devem ser aprendidas.

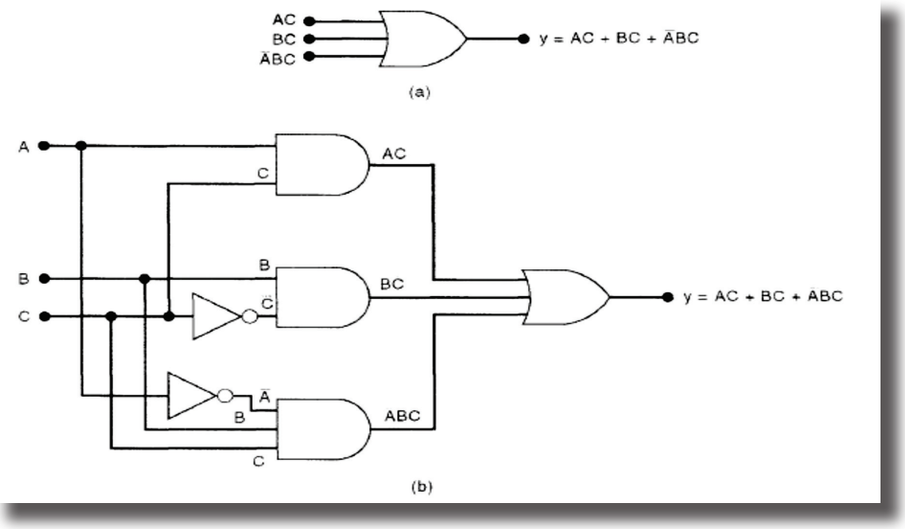


Fig. 17: Construindo um circuito lógico a partir de uma expressão booleana.

Exercício 1:

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

Portas NOR e portas NAND

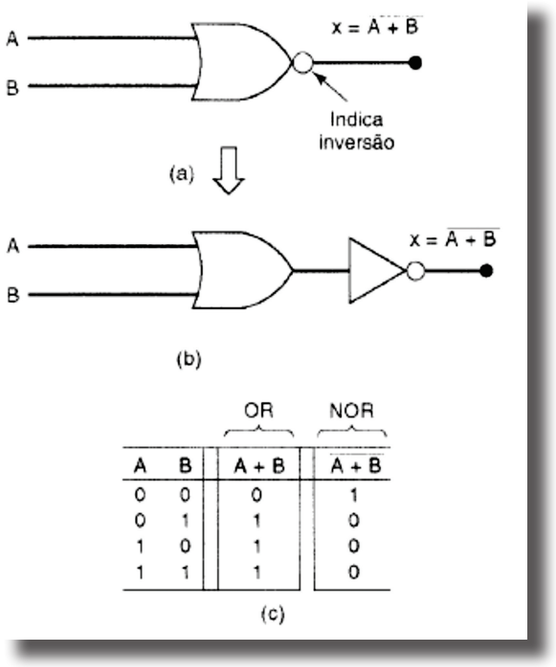
Existem dois outros tipos de portas lógicas, portas NOR e portas NAND, que são amplamente utilizadas em circuitos digitais. Estas portas, na verdade, combinam as operações básicas AND, OR e NOT. Por isso é relativamente simples descrever o seu funcionamento utilizando as operações booleanas aprendidas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

Porta NOR

O símbolo para uma porta NOR de duas entradas pode ser visto na Fig.18 (a). Este símbolo é igual ao símbolo de uma porta OR, exceto pelo pequeno círculo que possui na 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.18 (a) e (b) são equivalentes e a expressão booleana para a saída de uma porta NOR e dada por $x = \overline{A+B}$.

A tabela de verdade, que pode ser vista na Fig.18 (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. O 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. 18: (a) Símbolo para porta NOR;

(b) circuito equivalente;

Exercício 1:

Determine a forma de onda da saída de uma porta NOR para as formas de onda mostradas na Fig.19.

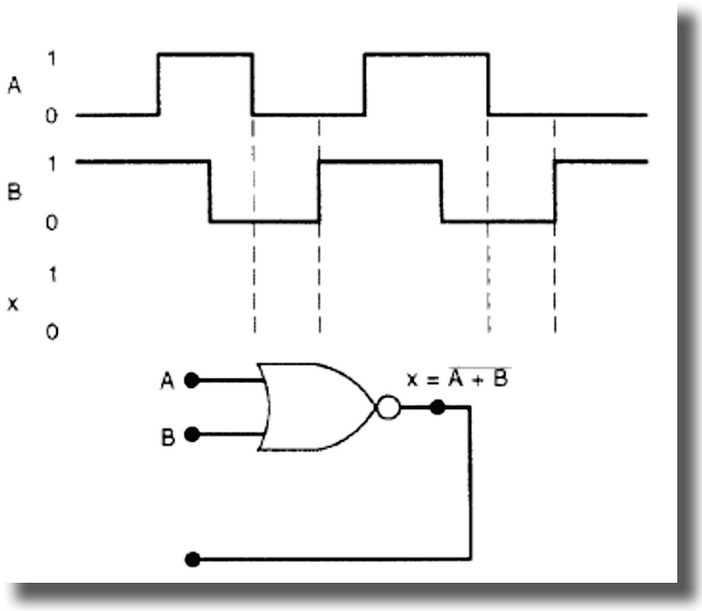


Fig. 19: Exercício 1

Exercício 2:

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

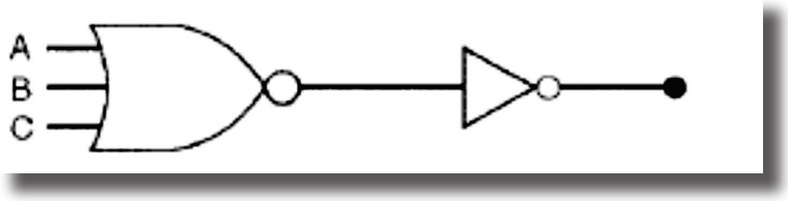


Fig. 20: Exercício 2

Porta NAND

O símbolo para uma porta NAND de duas entradas pode ser visto na Fig. 21(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.21 (a) e (b) são equivalentes e que a expressão booleana para a saída de uma porta NAND é $x = \overline{AB}$.

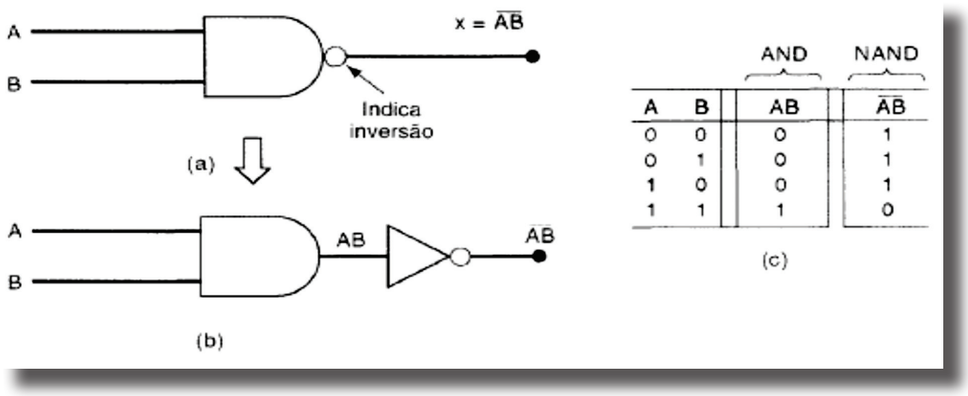


Fig. 21: (a) Símbolo para porta NAND; (b) Tabela de Verdade

A tabela de verdade vista na Fig.21 (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.

Exercício 1:

Determine a forma de onda da saída de uma porta NAND cujas formas de onda das entradas estão mostradas na Fig.22.

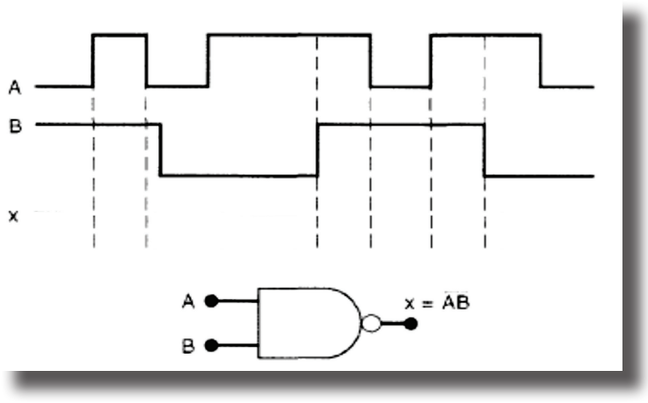


Fig.22: Exercício 1

Exercício 2:

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

Exercício 3:

Determine o nível lógico da saída do exercicio anterior quando A = B = C = 1 e D = 0.

Teoremas da Álgebra Booleana

Vimos como a álgebra booleana pode ser usada para nos ajudar a analisar um circuito lógico e expressar a sua operação matematicamente. Vamos continuar o estudo da álgebra booleana analisando os vários teoremas (regras), chamados teoremas booleanos, pois podem ajudar-nos a simplificar expressões e circuitos lógicos. O primeiro grupo de teoremas é mostrado na Fig.23. Em cada um, x é uma variável lógica que pode ser igual a 0 ou 1. Cada teorema está acompanhado por um circuito lógico que demonstra a 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 normal, 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 normal com a operação AND.

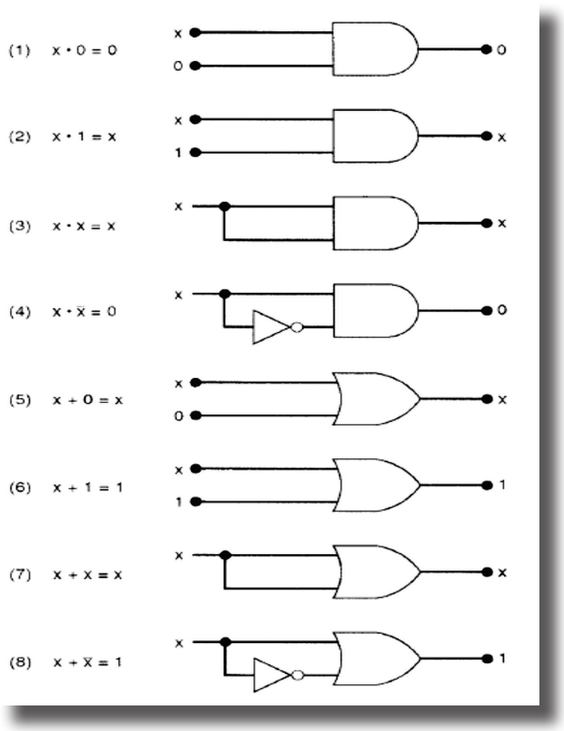


Fig. 23: Teoremas de uma variável

O teorema (3) pode ser provado ao verificar o resultado para cada valor possível de entrada. Se x = 0, entao 0.0 =0; se x = 1, então 1 . 1 = 1. Portanto, x . x = x.

O teorema (4) pode ser provado do mesmo modo. Em qualquer instante ou x ou o seu inverso 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 normal ou na operação OR.

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, em que qualquer instante x ou seu inverso estará em nível lógico 1, pelo que teremos sempre a operação OR de 0 e 1, cujo resultado será sempre 1.

Antes de introduzirmos mais teoremas, devemos ter em conta 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 raciocínio pode ser aplicado para o uso de qualquer um dos teoremas.

Teoremas com mais de uma Variável

Os teoremas apresentados de seguida envolvem o uso de mais de uma variável:

$(9)\ x + y = y + x$

$(10)\ x \,.\, y = y \,.\, x$

$(11)\ x + (y + z) = (y + x) + z = x + y + z$

$(12)\ x(yz) = (xy)z = xyz$

$(13a)\ x(y + z) = xy + xz$

$(13b)\ (w + x)(y + z) = wy + xy + wz + xz$

$(14)\ x + xy = x$

$(15)\ x + \bar{x}y = x + y$

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, que 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 evidenciar uma expressão. Caso tenhamos a soma de dois (ou mais) termos, cada um contendo uma variável comum, podemos colocar em evidência essa variável como fazemos na álgebra comum. Por exemplo, na expressão A $\overline{B}$ C + $\overline{A}$ $\overline{B}$ $\overline{C}$, podemos colocar em evidência 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 e B podem ser evidenciados, como vemos de seguida:

ABC + ABD = AB(C +D)

Os teoremas (9) a (13) são fáceis de memorizar 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 o demonstrado para o teorema (14):

Caso 1: Para $x = 0, y = 1$

$x + xy = x$

$x + xy = x$

$0 = 0$

Caso 2: Para $x = 0, y = 1$

$x + xy = x$

$0 + 0.1 = 0$

$0 + 0 = 0$

$0 = 0$

Caso 3: Para $x = 1, y = 0$

$x + xy = x$

$1 + 1.0 = 1$

$1 + 0 = 1$

$1 = 1$

Caso 4: Para $x = 1, y = 1$

$x + xy = x$

$1 + 1.1 = 1$

$1 + 1 = 1\ 1 = 1$

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

$x + xy = x(1 + y)\ x + xy = x\ .1[teorema\ (6)]\ \ x + xy = x[teorema(2)]$

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.

Exercício 1:

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

Exercício 2:

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

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:

$$(16)\ \left(\overline{x+y}\right)=\overline{x}\ .\overline{y}$$

$$(17)\ \left(\overline{x\ .y}\right)=\overline{x}+\overline{y}$$

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.

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:

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

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 torna-se

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

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

$$\left(\overline{A}+\overline{B}\right).\ \overline{C}=\overline{A}\ .\overline{C}+B\overline{C}$$

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

Exemplo 1:

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

O Exemplo 1 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 repetido até que a expressão seja reduzida a uma outra na qual apenas variáveis simples se encontram invertidas.

Outros dois exemplos podem ser vistos a seguir:

Exemplo 1	Exemplo 2
$z = \overline{\overline{A} + \overline{B}.\ C}$ $z = \overline{A}\ .\ (\overline{\overline{B}\ .\ C})$ $z = \overline{A}\ .\ (\overline{\overline{B}} + \overline{C})$	$w = \overline{\left(A + BC\right).\left(D + EF\right)}$ $w = \left(\overline{A + BC}\right) + \left(\overline{D + EF}\right)$ $w = \left(\overline{A}\ .\overline{BC}\right) + \left(\overline{D}.\ \overline{EF}\right)$ $w = \left[\overline{A}\ .\left(\overline{B} + \overline{C}\right)\right] + \left[\overline{D}.\left(\overline{E} + \overline{F}\right)\right]$ $w = \overline{A}\overline{B} + \overline{A}\overline{C} + \overline{D}\overline{E} + \overline{D}\overline{F}$

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

$$\overline{x + y + z} = \overline{x}.\overline{y}.\overline{z}$$

$$\overline{x.y.z} = \overline{x} + \overline{y} + \overline{z}$$

Aqui podemos ver que o grande sinal de inversão foi partido em dois pontos e, nesses pontos, o sinal do operador foi trocado pelo 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:

$$x = \overline{\overline{AB}.\overline{CD}.\overline{EF}}$$

$$x = \overline{\overline{AB}} + \overline{\overline{CD}} + \overline{\overline{EF}}$$

$$x = AB + CD + EF$$

Implicações dos Teoremas de DeMorgan

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

$$\left(\overline{x+y}\right)=\overline{x}\ .\overline{y}$$

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.24 (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.24 (b), onde os pequenos círculos nas entradas representam a operação de inversão.

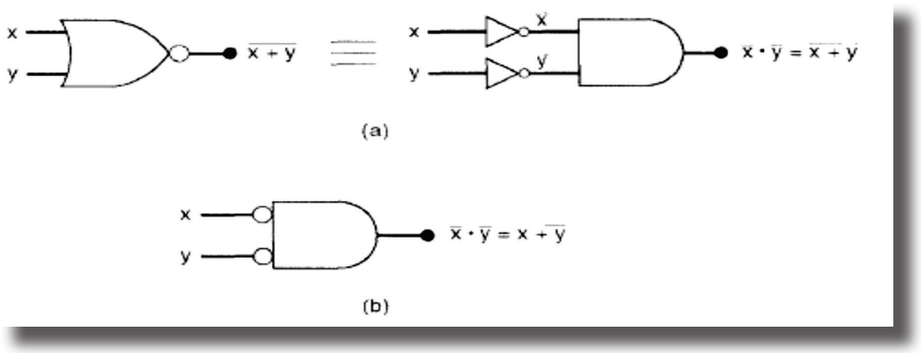


Fig. 24: (a) Circuitos equivalentes obtidos pela aplicação do teorema (16); (b) símbolo alternativo para a função NOR

Agora considere o teorema (17):

$$\left(\overline{x.y}\right) = \overline{x} + \overline{y}$$

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.25(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 frequentemente desenhada como mostra a Fig.25 (b), onde os círculos mais uma vez representam inversão.

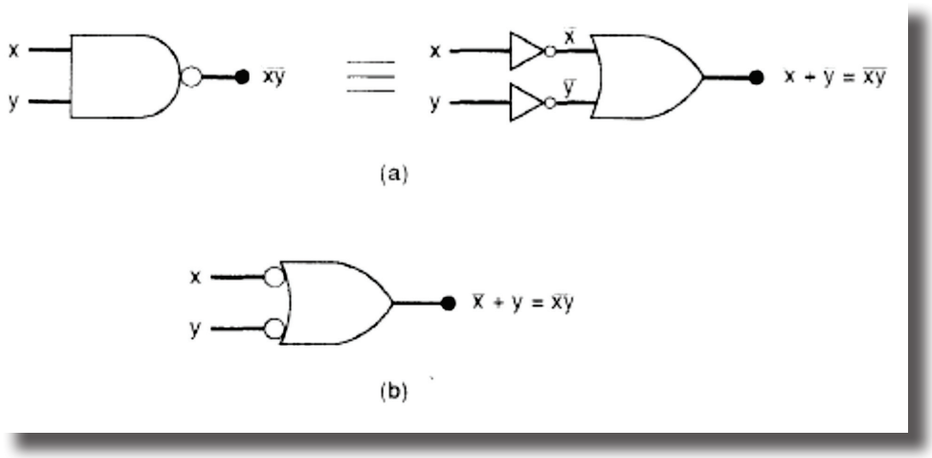


Fig. 25: (a) Circuitos equivalentes obtidos pela aplicação do teorema (17); (b) símbolo alternativo para a função NAND.

Exercício 1:

Determine a expressão lógica para a saída do circuito da Fig.26 e simplifique-a usando os teoremas de DeMorgan.

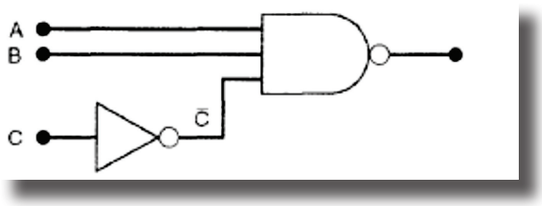


Fig.26: Exercício 1

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 NOT. 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.27.

Em primeiro lugar, na Fig.27 (a), temos uma porta NAND de duas entradas onde estas se encontram propositadamente ligadas a uma mesma variável A. Nesta configuração, a porta NAND simplesmente age como um simples INVERSOR, uma vez que a sua saída $x = \overline{A.A} = \overline{A}$.

Na Fig.27(b) temos duas portas NAND ligadas 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 $\overline{AB}$ = *AB*, que é a função AND desejada.

A operação OR pode ser implementada usando portas NAND como está apresentado na Fig.27 (c). Neste caso, as portas NAND número 1 e 2 são usadas como inversoras para inverter as entradas, de modo a que o resultado final seja $x = \overline{\overline{A}.\overline{B}}$, que pode ser simplificado para $x = a + b$ através do teorema de DeMorgan.

De modo similar, pode ser provado que as portas NOR podem ser combinadas para implementar qualquer uma das operações booleanas. Isto é ilustrado na Fig.28. O item (a) mostra que uma porta NOR com as entradas ligadas juntas comporta-se como um INVERSOR, uma vez que a sua saída e x = A + A = A.

Na Fig.28 (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.28(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* . *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. Esta característica das portas NAND e NOR pode ser bastante útil no projeto de circuitos lógicos, como mostra o exemplo a seguir.

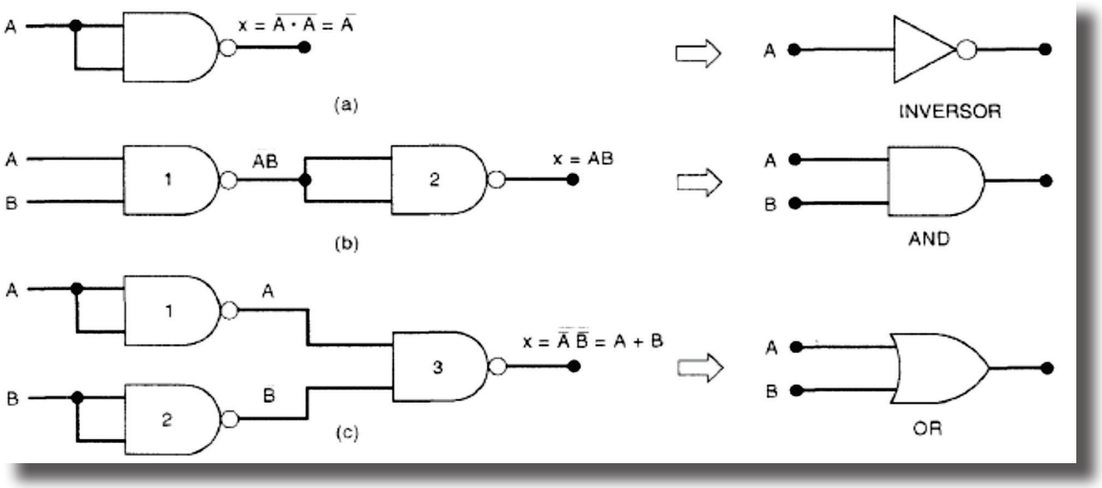


Fig. 27: Portas NAND podem ser usadas para implementar qualquer função booleana

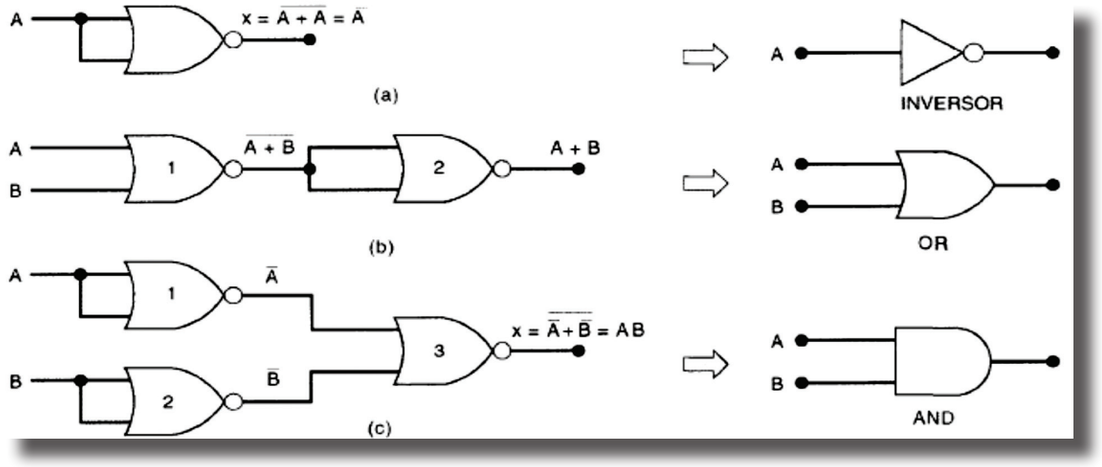


Fig. 28: Portas NOR podem ser usadas para implementar qualquer função booleana.

Exemplo 1:

Num determinado processo de fabricação, uma esteira de transporte deve ser desligada sempre que ocorram determinadas condições. Estas condições são monitorizadas 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.

É necessário um circuito lógico para produzir 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.29 estão disponíveis.

Cada CI é quádruplo, o que significa que ele contém quatro portas idênticas num 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.30 (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 respetivos 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.30 (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.30 (b).

À primeira vista, esse novo circuito parece precisar de sete portas NAND. No entanto, as portas NAND de número 3 e 5 estão ligadas 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.30 (c).

Esse circuito é mais eficiente do que o da Fig.30 (a) porque utiliza três portas NAND de duas entradas que podem ser implementadas por um CI, o 74LS00.

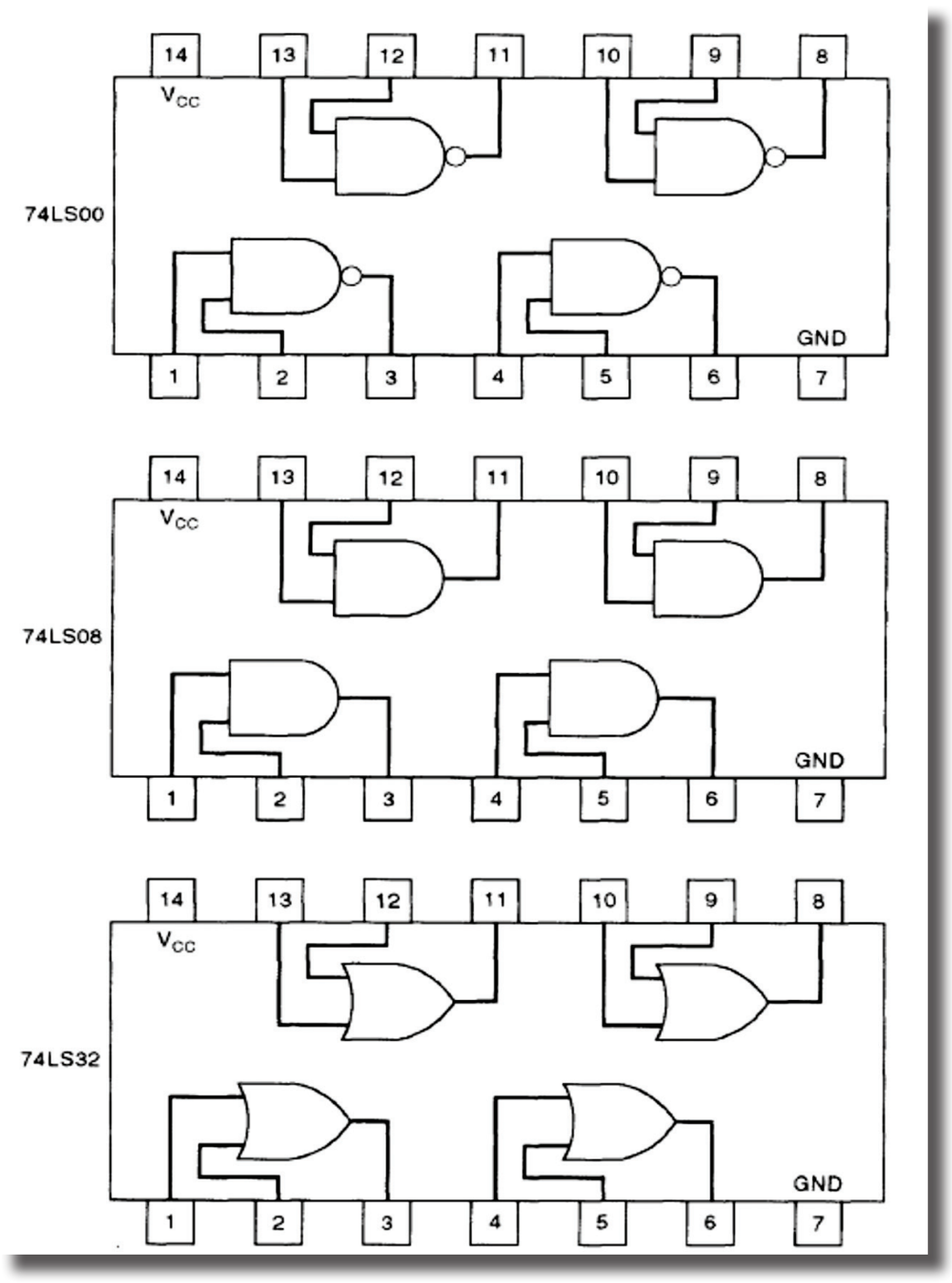


Fig. 29: Exemplo 1

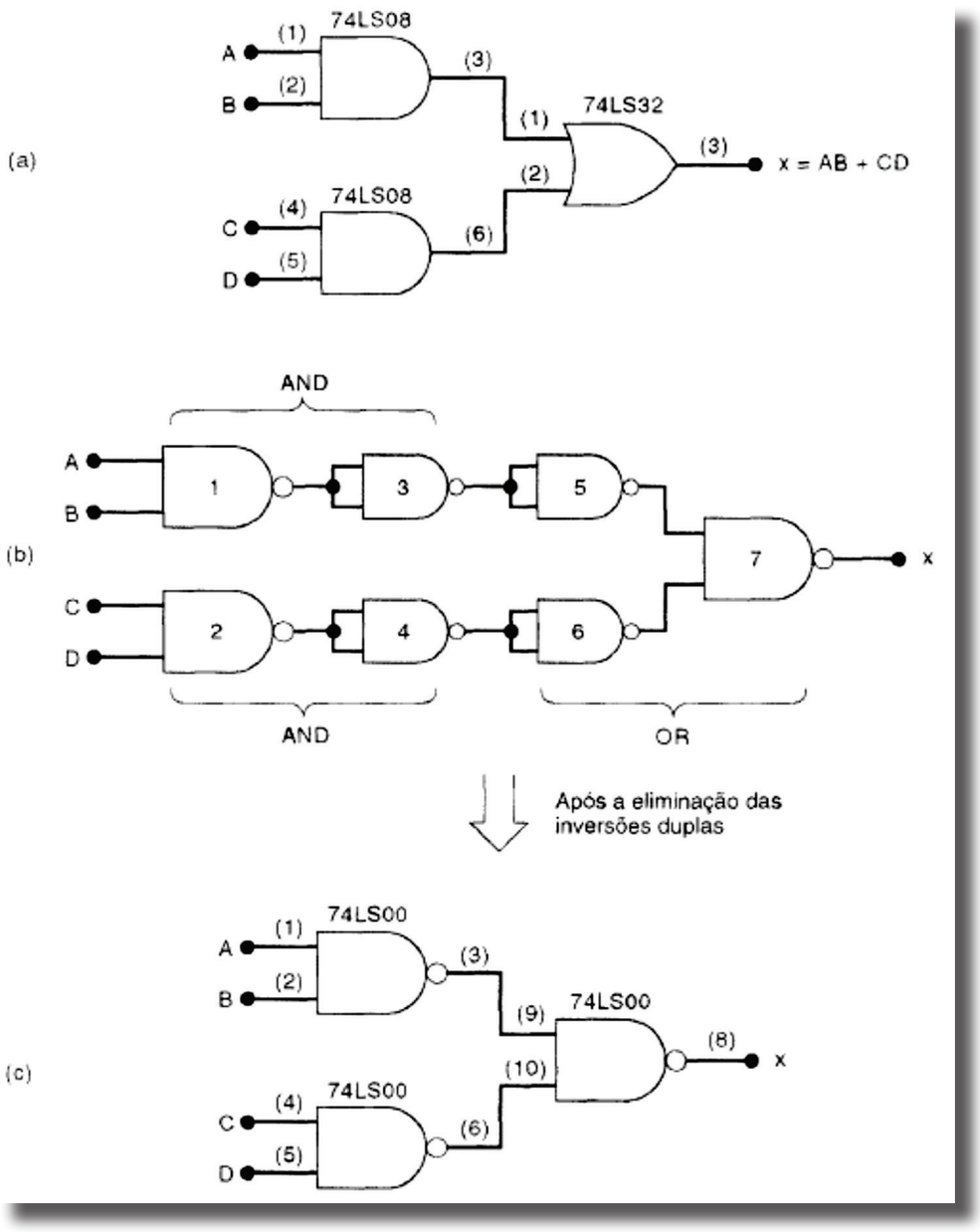


Fig. 30: Implementações possíveis para o Exemplo 1

Circuitos XOR e XNOR

Dois circuitos lógicos especiais que frequentemente aparecem em sistemas digitais são os circuitos exclusive-OR (XOR) e o exclusive-NOR (XNOR).

XOR

Considere o circuito lógico da Fig. 31 (a). A expressão de saída deste circuito é:

$$x = \bar{A}B + A\bar{B}$$

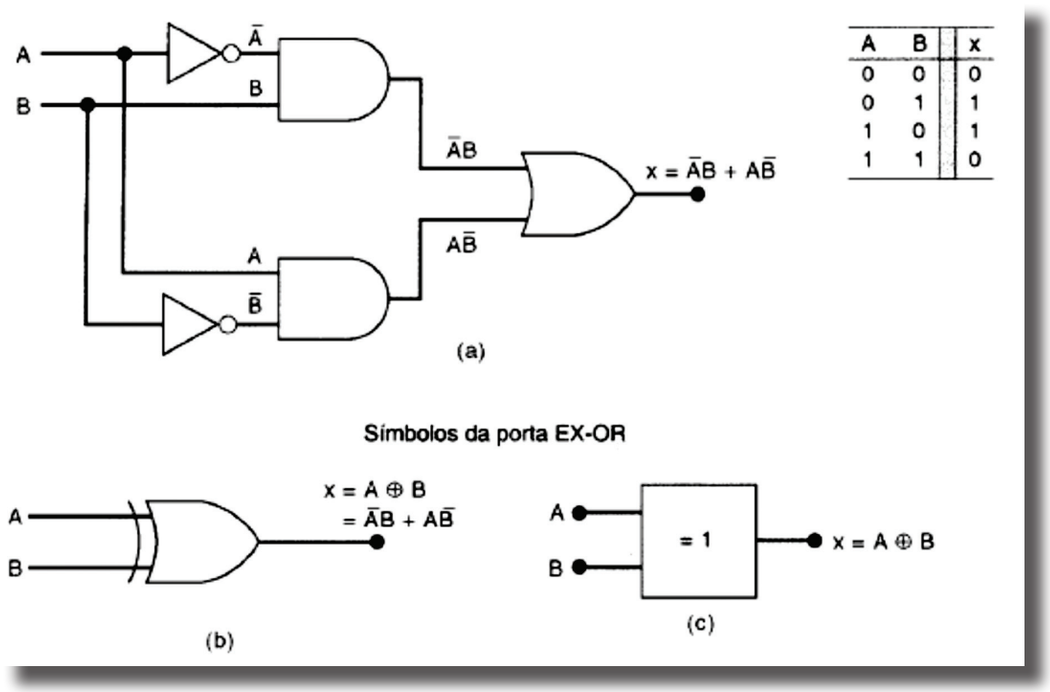


Fig. 31: (a) Tabela DE verdade e circuito exclusive-OR; (b) símbolo tradicional da porta EX-OR; (c) símbolo IEEE/ANSI para a porta EX-OR.

A tabela de verdade apresentada mostra que x = 1 para dois casos: A = 0, B = 1 (o termo $\bar{A}$ B) e A = 1, B = 0 (o termo A $\bar{B}$). Noutras palavras:

Este circuito produz uma saída em ALTO sempre que as duas entradas estão em níveis opostos.

Este é o circuito exclusive-OR, que daqui para a frente será abreviado como XOR.

Esta combinação especial de portas lógicas ocorre frequentemente e é muito útil em certas aplicações. Na verdade, o circuito XOR tem um símbolo próprio, que é apresentado na Fig.31 (b). Supõe-se que este símbolo contém todas as portas lógicas de um circuito XOR e portanto tem a mesma expressão lógica e a mesma tabela-verdade.

Este circuito XOR é normalmente mencionado como uma porta XOR, que é considerado um outro tipo de porta lógica. O símbolo IEEE/ANSI para uma porta EX-OR é mostrado na Fig.31 (c). A notação de dependência (= 1) dentro do bloco indica que a saída está ativa-ALTO somente quando uma única entrada está em ALTO.

Uma porta XOR tem apenas duas entradas. Não existem portas XOR de três ou quatro entradas. As duas entradas são combinadas de modo que $x = \overline{A}\,B + A\,\overline{B}$. Um modo abreviado que algumas vezes é usado para indicar uma expressão de saída XOR é

$$x = A \oplus B$$

Onde o símbolo $\oplus$ representa a operação da porta XOR.

As características de uma porta XOR podem ser resumidas da seguinte maneira:

1. Tem apenas duas entradas e sua saída é: $x = \overline{A}B + A\overline{B} = A \oplus B$

2. A sua saída está em ALTO somente quando as duas entradas estão em níveis diferentes.

Diversos CIs que contêm portas XOR estão disponíveis. Os chips relacionados a seguir são XOR quádruplos, que contêm quatro portas XOR.

- 74LS86 XOR quadrupla (família TTL)
- 74C86 XOR quadrupla (família CMOS)
- 74HC86 OR quadrupla (família HCMOS — Highspeed CMOS — CMOS de alta velocidade)

X NOR

O circuito exclusive - NOR (abreviado como XNOR) opera ao contrário do circuito XOR. A Fig.32(a) mostra um circuito XNOR e sua respetiva tabela de verdade. A expressão de saída é

$$x = AB + \bar{A}\bar{B}$$

que indica juntamente com a tabela de verdade que x é 1 para dois casos: A = B = 1 (o termo AB) e A = B = 0 (o termo $\bar{A}\bar{B}$). Em outras palavras:

Este circuito produz uma saída em ALTO sempre que as duas entradas estão no mesmo nível.

Deve estar claro que a saída de um circuito XNOR e exatamente o inverso da saída de um circuito XOR. O símbolo tradicional de uma porta XNOR é obtido simplesmente adicionando-se um pequeno círculo à saída do símbolo do XOR Fig.32 (b). O símbolo IEEE/ANSI adiciona um pequeno triângulo a saída do símbolo XOR. Ambos os símbolos indicam uma saída que vai para o estado ativo em BAIXO quando somente uma entrada está em ALTO.

A porta XNOR também tem apenas duas entradas e combina-as de modo que a sua saída seja:

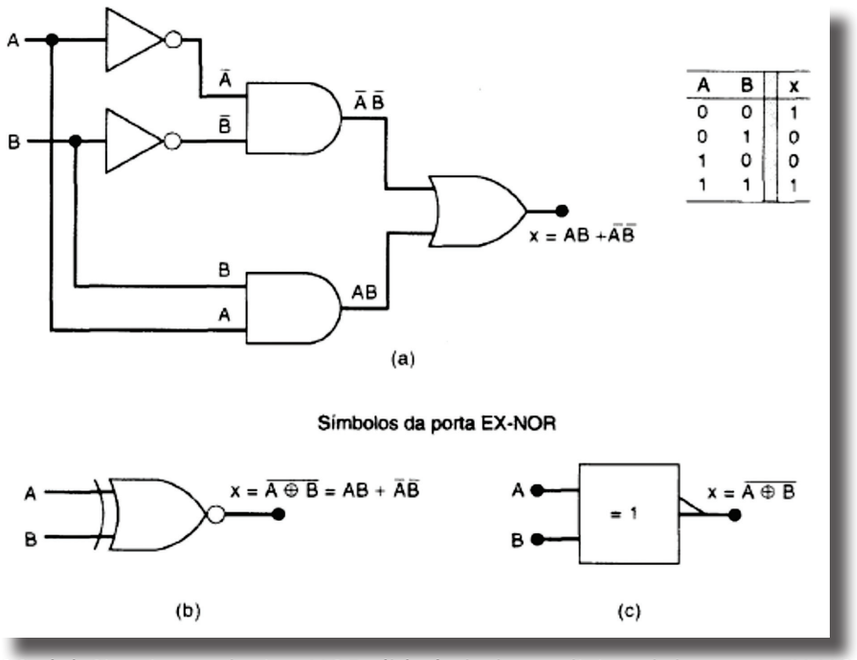


Fig. 32: (a) Circuito exclusive-NOR; (b) símbolo tradicional da porta EX-NOR; (c) símbolo IEEE/ANSI.

Um modo abreviado de indicar uma expressão de saída de um XNOR é:

$x = \overline{A \oplus B}$

que é simplesmente o inverso da operação XOR. A porta XNOR é resumida como se segue:

1. Tem apenas duas entradas e sua saída é

$$x = AB + \overline{A}\,\overline{B} = \overline{A \oplus B}$$

2. A sua saída está em ALTO somente quando as duas entradas estão no mesmo nível.

Diversos CIs que contêm portas EX-NOR estão disponíveis. Os chips apresentados a seguir são XNOR quádruplos, que contêm quatro portas XNOR.

- 74LS266 EX-NOR quadrupla (família TTL)
- 74C266 EX-NOR quadrupla (família CMOS)
- 74HC266 EX-NOR quadrupla (família HCMOS)

Exercício 1:

Determine a forma de onda de saída para as formas de onda de entrada na Fig. 32.

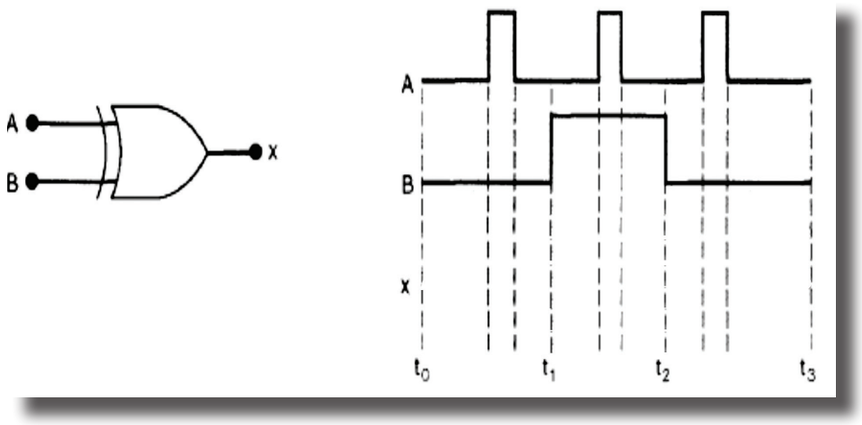


Fig. 32: Exemplo 1

Exercício 2:

x_1x_0 , representa um número binário de dois bits que pode ter qualquer valor (00, 01, 10 ou 11); por exemplo, quando x_1 = 1 e x_0 = 0, o número binário é 10, e assim por diante. Analogamente, y_1y_0 representa um outro número binário de dois bits.

Projete um circuito lógico, usando as entradas $x_1x_0 \, e \, y_1y_0$, cuja saída vai para ALTO somente quando os dois números binários $x_1x_0 \, e \, y_1y_0$ são iguais.

Exercício 3:

Quando se simplifica a expressão para a saída de um circuito lógico convencional, podem encontrar-se operações XOR ou XNOR durante a evidenciação. Isto conduz, frequentemente, ao uso de portas XOR ou XNOR na implementação do circuito final.

Para ilustrar, simplifique o circuito da Fig.33 (a).

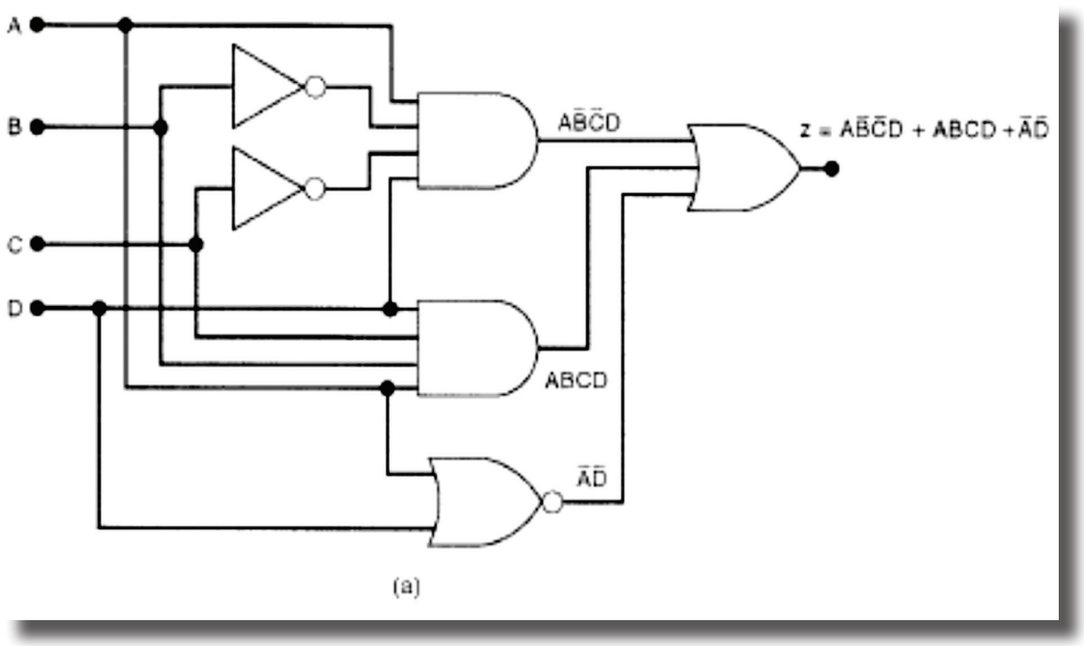


Fig. 33: O exercício 3

Método do mapa de Karnaugh

O mapa de Karnaugh é um método gráfico usado para simplificar uma equação lógica ou para converter uma tabela de verdade no seu circuito lógico correspondente de um modo simples e ordenado. Embora um mapa de Karnaugh (daqui para a frente abreviado como mapa K) possa ser usado em problemas que envolvem qualquer número de variáveis de entrada, a sua utilidade prática está limitada a seis variáveis. A apresentação que se segue está restrita a problemas com até quatro entradas, pois mesmo os problemas com cinco ou seis entradas são demasiado complicados, sendo melhor resolvidos por um programa de computador.

Formato do Mapa de Karnaugh

O mapa K, como uma tabela de verdade, **é** um meio de mostrar a relação entre as entradas lógicas e a saída desejada. A Fig.34 apresenta três exemplos de mapas K, para duas, para três e para quatro variáveis, em conjunto com as tabelas de verdade correspondentes. Estes exemplos ilustram os seguintes pontos importantes:

1. A tabela de verdade fornece o valor da saída X para cada combinação de valores da entrada. O mapa K fornece a mesma informação num formato diferente. Cada linha na tabela de verdade corresponde a um quadrado no mapa K. Por exemplo, na Fig. 34 (a), a condição A = 0, B = 0, na tabela de verdade, corresponde ao quadrado $\overline{AB}$ no mapa K. Como a tabela de verdade mostra X = 1 para este caso, 1 é colocado no quadrado $\overline{A}\ \overline{B}$ no mapa K. Do mesmo modo, a condição A = 1, B = 1 na tabela de verdade corresponde ao quadrado AB no mapa K. Como X = 1 para este caso, 1 é colocado no quadrado AB. Todos os outros quadrados são preenchidos com 0s. Esta mesma ideia é usada nos mapas de três e quatro variáveis mostrados na figura.

2. Os quadrados do mapa K são identificados de modo que quadrados adjacentes horizontalmente diferem apenas numa variável. Por exemplo, o quadrado do canto superior esquerdo no mapa de quatro variáveis é $\overline{A}\overline{B}\overline{C}\overline{D}$, enquanto o quadrado imediatamente à sua direita é $\overline{A}\overline{B}\overline{C}\,D$ (apenas a variável D é diferente). Do mesmo

modo, quadrados adjacentes verticalmente diferem apenas numa variável. Por exemplo, o quadrado do canto superior esquerdo no mapa de quatro variáveis é $\overline{A}\overline{B}\overline{C}\overline{D}$, enquanto o quadrado diretamente abaixo dele é $\overline{A}\,B\,\overline{C}\overline{D}$ (apenas a variável B é diferente). Note que cada quadrado na linha superior é considerado adjacente ao quadrado correspondente na linha inferior. Por exemplo, o quadrado $\overline{A}\overline{B}\,CD$ na linha superior é adjacente ao quadrado $\overline{A}\overline{B}\,CD$ na linha inferior, pois diferem apenas na variável *A*. Podemos imaginar que a parte superior do mapa foi dobrada para tocar na parte inferior. Analogamente, os quadrados da coluna mais à esquerda são adjacentes aos quadrados correspondentes da coluna mais a direita.

3. Para que os quadrados adjacentes, tanto na horizontal quanto na vertical, difiram em apenas uma variável, a identificação de cima para baixo deve ser feita na ordem apresentada: $\overline{A}\overline{B}$, $\overline{A}\,B$, AB, $A\,\overline{B}$. O mesmo se aplica a identificação da esquerda para a direita.

4. Uma vez que um mapa K foi preenchido com 0s e 1s, a expressão da soma de produtos para a saída A pode ser obtida juntando-se com OR os quadrados que contêm 1. No mapa de três variáveis da Fig.34 (b), os quadrados $\overline{A}\overline{B}\overline{C}$, $\overline{A}\overline{B}C$, $\overline{A}B\overline{C}$ e $AB\overline{C}$ contêm 1, portanto X = $\overline{A}\overline{B}\overline{C}$ + $\overline{A}\overline{B}C$ + $\overline{A}B\overline{C}$ + $AB\overline{C}$.

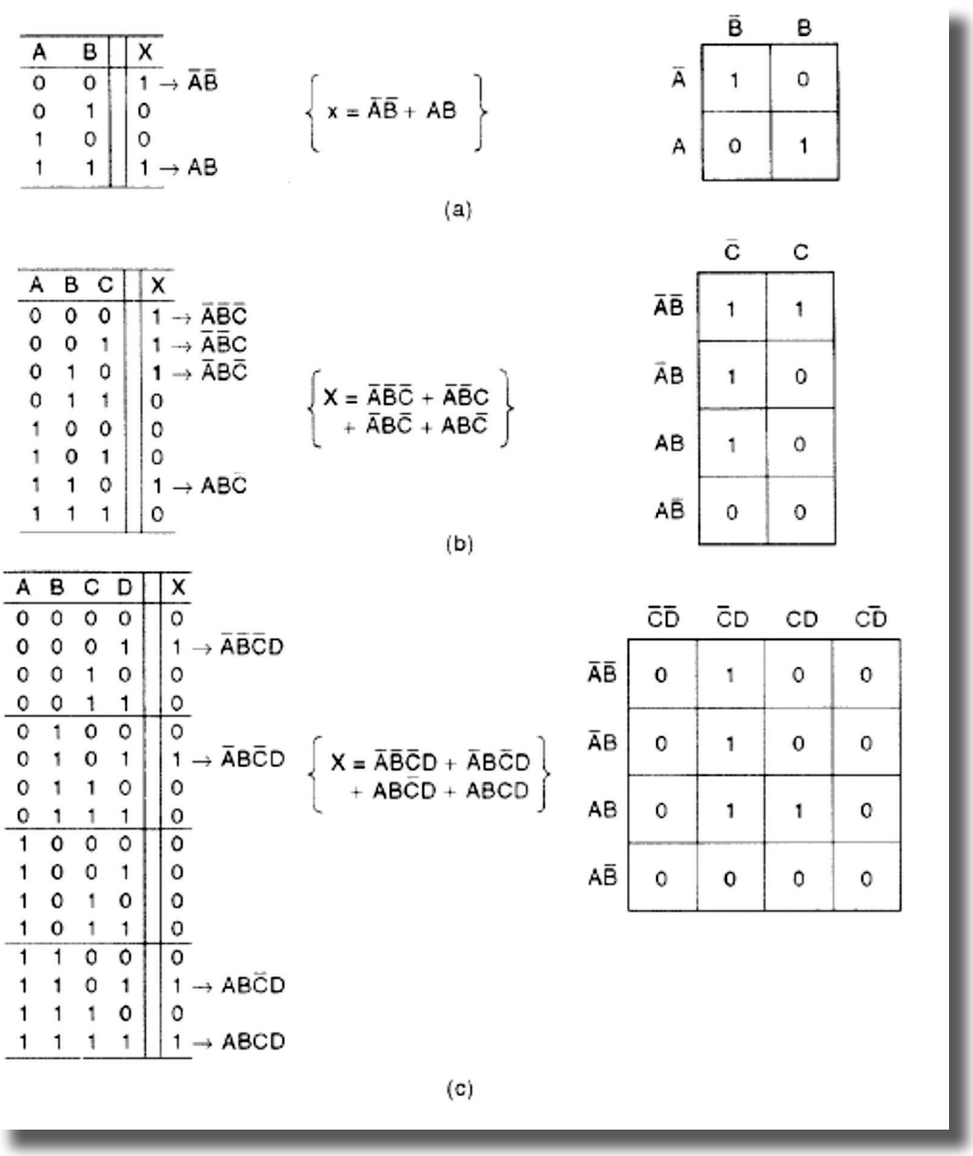


Fig. 34:Mapas de Karnaugh e tabelas de verdade para (a) duas, (b) três e (c) quatro variáveis.

Agrupamento de Termos no Mapa

A expressão para a saída *X* pode ser simplificada combinando-se adequadamente os quadrados no mapa K que contêm 1. O processo de combinar estes 1s é chamado de agrupamento.

Agrupando Dois Termos (Pares)

Na Fig. 40 (a) está o mapa K para uma determinada tabela de verdade de três variáveis. Este mapa contém um par de 1s que são adjacentes na vertical: o primeiro representa $\overline{A}B\overline{C}$ e o segundo representa $AB\overline{C}$. Repare que nestes dois termos apenas a variável *A* aparece tanto na forma normal quanto na complementar (*B* e $\overline{C}$ permanecem inalteradas).

Estes dois termos podem ser agrupados (combinados) para dar um resultado que elimina a variável *A*, visto que ela aparece em ambas as formas, normal e complementar. Isto é facilmente provado como se segue:

$$x = \overline{A}B\overline{C} + AB\overline{C}$$
$$x = B\overline{C}\left(\overline{A} + \overline{C}\right)$$
$$x = B\overline{C}\left(1\right) = B\overline{C}$$

Este mesmo princípio permanece válido para qualquer par de 1s adjacentes na vertical ou na horizontal. A Fig. 35 (b) mostra um exemplo de dois 1s horizontalmente adjacentes. Estes dois podem ser agrupados e a variável *C* eliminada, já que ela aparece nas formas não complementada e complementada para resultar em *X* = $\overline{A}B$.

Um outro exemplo está ilustrado na Fig. 40 (c). Num mapa K a linha superior e a linha inferior são consideradas adjacentes. Assim, os dois 1s neste mapa podem ser agrupados para produzir como resultado $\overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} = \overline{D}$.

A Fig. 35 (d) mostra um mapa K que tem dois pares de 1s que podem ser agrupados. Os dois 1s na linha superior são horizontalmente adjacentes. Os dois 1s na linha inferior também são adjacentes, já que, num mapa K, a coluna de quadrados mais a esquerda é considerada adjacente com a coluna mais à direita. Quando o par de 1s superior é agrupado, a variável D é eliminada (pois ela aparece tanto como *D* quanto como $\overline{D}$) para gerar o termo $\overline{A}\overline{B}C$. Agrupar o par inferior elimina a variável *C* para gerar o termo $A\overline{B}\overline{D}$. Estes dois termos são unidos por um *OR*, obtendo-se o resultado final para *X*

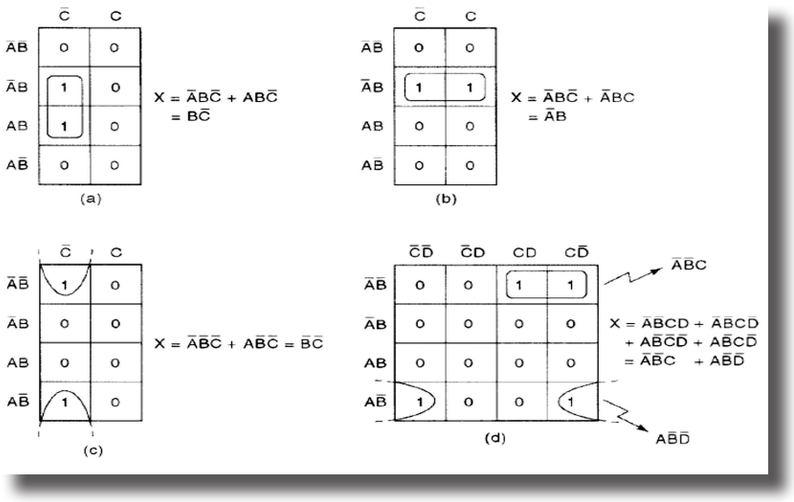


Fig. 35: Exemplos de agrupamentos de pares de 1s adjacentes.

Agrupar um par de 1s adjacentes num mapa K elimina a variável que aparece nas formas complementada e não complementada.

Agrupando Quatro Termos (Quartetos)

Um mapa K pode conter um grupo de quatro 1s adjacentes entre si. Este grupo é denominado quarteto. A Fig. 36 mostra vários exemplos de quartetos. Na parte (a) os quatro 1s são verticalmente adjacentes, e na parte (b) eles são adjacentes na horizontal. O mapa K na Fig. 36 (c) contém quatro 1s formando um quadrado, e eles são considerados adjacentes entre si. Os quatro 1s na Fig. 36 (d) também são adjacentes, assim como os da Fig. 36 (e) porque, conforme apresentado anteriormente, as linhas superior e inferior são consideradas adjacentes entre si, do mesmo modo que as colunas mais á esquerda e mais a direita.

Quando um quarteto é agrupado, o termo resultante contém apenas as variáveis que não mudam de forma para todos os quadrados do quarteto. Por exemplo, na Fig. 36 (a), os quatro quadrados que contêm 1s são $\overline{A}\overline{B}C$, $\overline{A}BC$, ABC e $A\overline{B}C$. Um exame destes termos revela que apenas a variável *C* permanece inalterada (tanto *A* como *B* aparecem nas formas não complementada e complementada). Assim, a expressão resultante para A e simplesmente X = *C*. Isto pode ser provado como se segue:

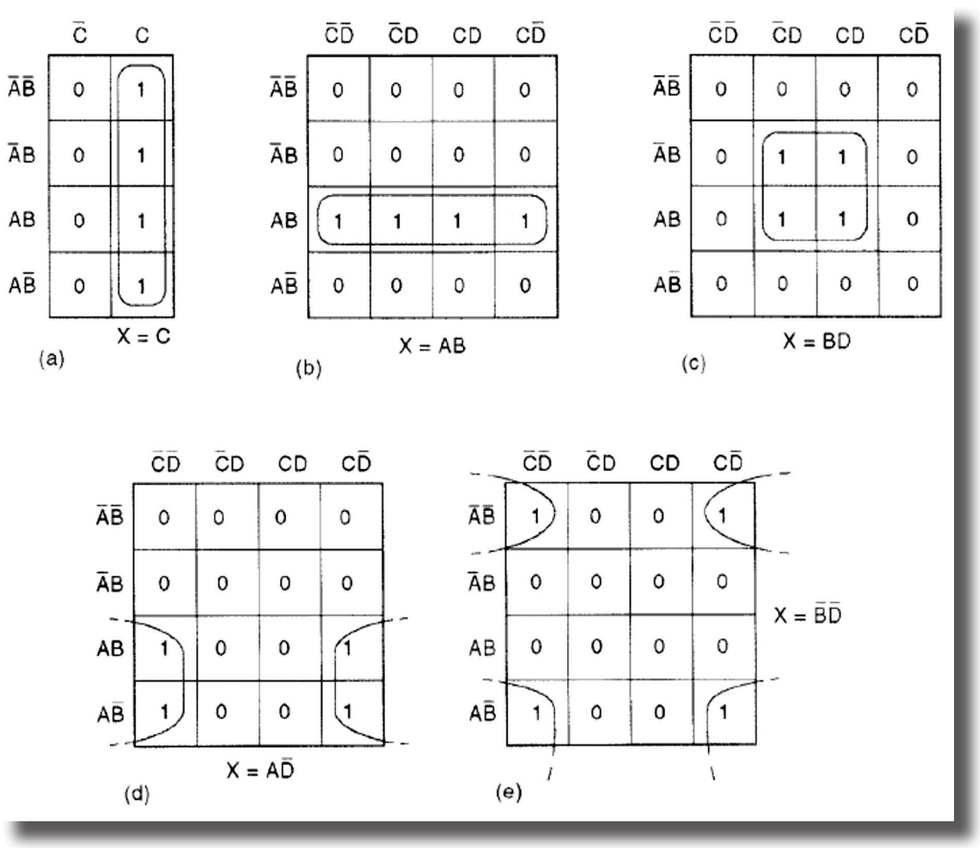


Fig. 36: Exemplos de agrupamentos de quatro 1s (quartetos).

$$X = \bar{A}\bar{B}C + \bar{A}BC + ABC + A\bar{B}C$$
$$X = \bar{A}C\left(\bar{B} + B\right) + AC\left(B + \bar{B}\right)$$
$$X = \bar{A}C + AC$$
$$X = C\left(\bar{A} + A\right) = C$$

Outro exemplo, considere a Fig.36 (d), onde os quatro quadrados que contêm 1s são: $AB\bar{C}\bar{D}$, $A\bar{B}\bar{C}\bar{D}$, $ABC\bar{D}$, e $A\bar{B}C\bar{D}$. Um exame destes termos indica que somente as variáveis *A* e *D* permanecem inalteradas, portanto a expressão simplificada para *X* é:

$$X = A\bar{D}$$

Isto pode ser provado da mesma maneira que foi feito anteriormente. Deve analisar-se cada um dos casos na Fig.36 para verificar as expressões indicadas para *X*.

Resumindo:

Agrupar um quarteto de 1s elimina as duas variáveis que aparecem nas formas complementada e não complementada.

Agrupando Oito Termos (Octetos)

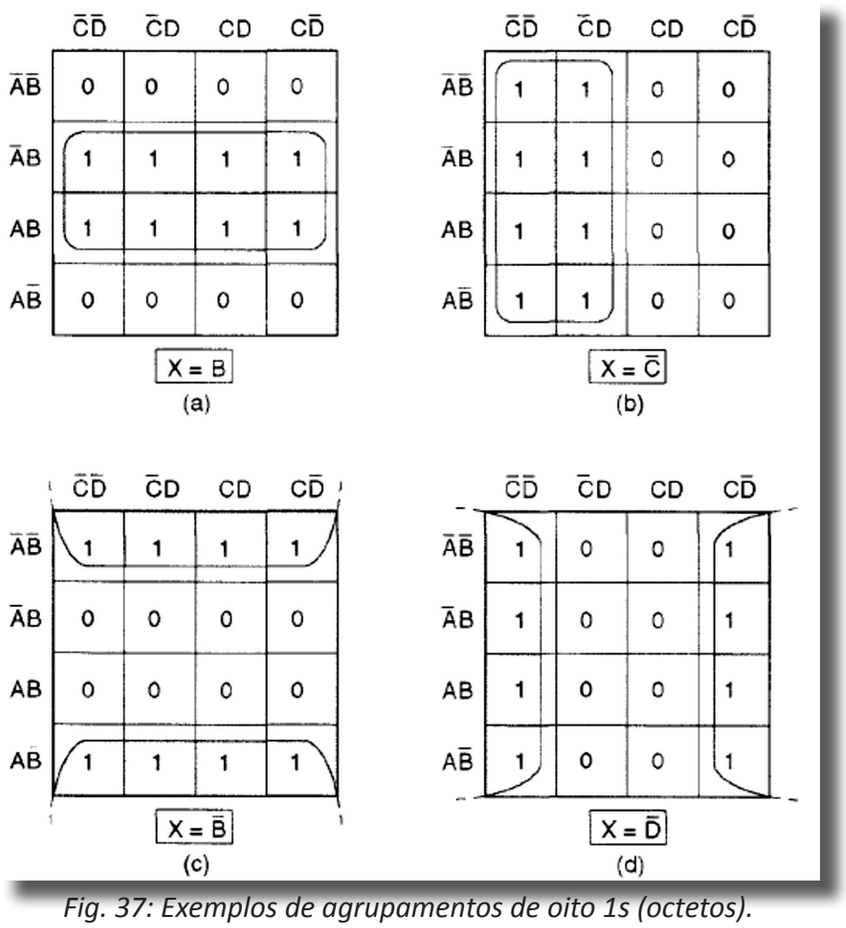


Fig. 37: Exemplos de agrupamentos de oito 1s (octetos).

Um grupo de oito 1s que são adjacentes entre si e chamado de octeto. Alguns exemplos de octetos são mostrados na Fig.37. Quando um octeto é agrupado num mapa de quatro variáveis, três das quatro variáveis são eliminadas, porque apenas uma variável permanece inalterada. Por exemplo, um exame dos oito quadrados agrupados na Fig. 37(a) mostra que somente a variável *B* está na mesma forma para todos os oito quadrados; as outras variáveis aparecem nas formas complementadas e não-complementada. Portanto, para este mapa, X=*B*. O leitor pode verificar os resultados para os outros exemplos na Fig. 37.

Resumindo:

Agrupar um octeto de 1s elimina as três variáveis que aparecem nas formas complementada e não -complementada.

Processo Completo de Simplificação

Vimos que o agrupamento de pares, quartetos e octetos num mapa K pode ser usado para obtermos uma expressão simplificada.

Podemos resumir a regra para grupos de qualquer tamanho:

Quando uma variável aparece nas formas complementada e não complementada dentro de um grupo, esta variável é eliminada da expressão. Variáveis que não mudam para todos os quadrados do grupo devem aparecer na expressão final.

Deve ficar claro que um grupo maior de 1s elimina mais variáveis. Um grupo de dois elimina uma variável, um grupo de quatro elimina duas e um grupo de oito elimina três. Este princípio será agora utilizado para obter uma expressão lógica simplificada a partir de um mapa K que contenha qualquer combinação de 1s e 0s.

O procedimento será primeiramente resumido e então aplicado em vários exemplos. Os passos a seguir são realizados para a utilização do método do mapa K para simplificação de uma expressão booleana:

Passo 1: Construa o mapa K e coloque 1s nos quadrados que correspondem aos 1s na tabela de verdade. Coloque 0s nos outros quadrados.

Passo 2: Examine o mapa para detetar 1s adjacentes e agrupe aqueles 1s que não são adjacentes a quaisquer outros 1s. Estes são denominados 1s isolados.

Passo 3: Em seguida, procure por aqueles 1s que são adjacentes a somente um outro 1. Agrupe todos os pares que contêm 1s.

Passo 4: Agrupe qualquer octeto, mesmo que ele contenha alguns 1s que já tenham sido combinados.

Passo 5: Agrupe qualquer quarteto que contenha um ou mais 1s que ainda não tenham sido combinados, certificando-se de usar o número mínimo de agrupamentos.

Passo 6: Agrupe quaisquer pares necessários para incluir quaisquer 1s que ainda não tenham sido combinados, certificando-se de usar o número mínimo de agrupamentos.

Passo 7: Forme a soma OR de todos os termos gerados por cada agrupamento.

Estes passos são seguidos e mencionados nos exemplos seguintes. Em cada caso, a expressão lógica resultante está na sua forma de soma de produtos mais simples.

Exemplo 1:

A Fig.38 (a) mostra o mapa K para um problema de quatro variáveis. Vamos supor que o mapa foi obtido a partir da tabela-verdade do problema (passo 1). Os quadrados estão numerados por conveniência para identificação de cada grupo.

Passo 2: O quadrado 4 é o único quadrado que contém um 1 que não é adjacente a qualquer outro 1. Está separado e indicado como grupo 4.

Passo 3: O quadrado 15 é adjacente apenas ao quadrado 11. Este par é agrupado e indicado como grupo 11, 15.

Passo 4: Não existem octetos.

Passo 5: Os quadrados 6, 7,10 e 11 formam um quarteto. Este quarteto é agrupado (grupo 6, 7, 10, 11). Repare que o quadrado 11 é usado novamente, embora já seja parte do grupo 11, 15.

Passo 6: Todos os 1s já estão agrupados.

Passo 7: Cada grupo gera um termo na expressão para X. O grupo 4 é simplesmente $A\bar{B}C\bar{D}$. O grupo 11, 15 e *ACD* (a variável *B* foi eliminada). O grupo 6, 7, 10, 11 e BD (A e C foram eliminadas).

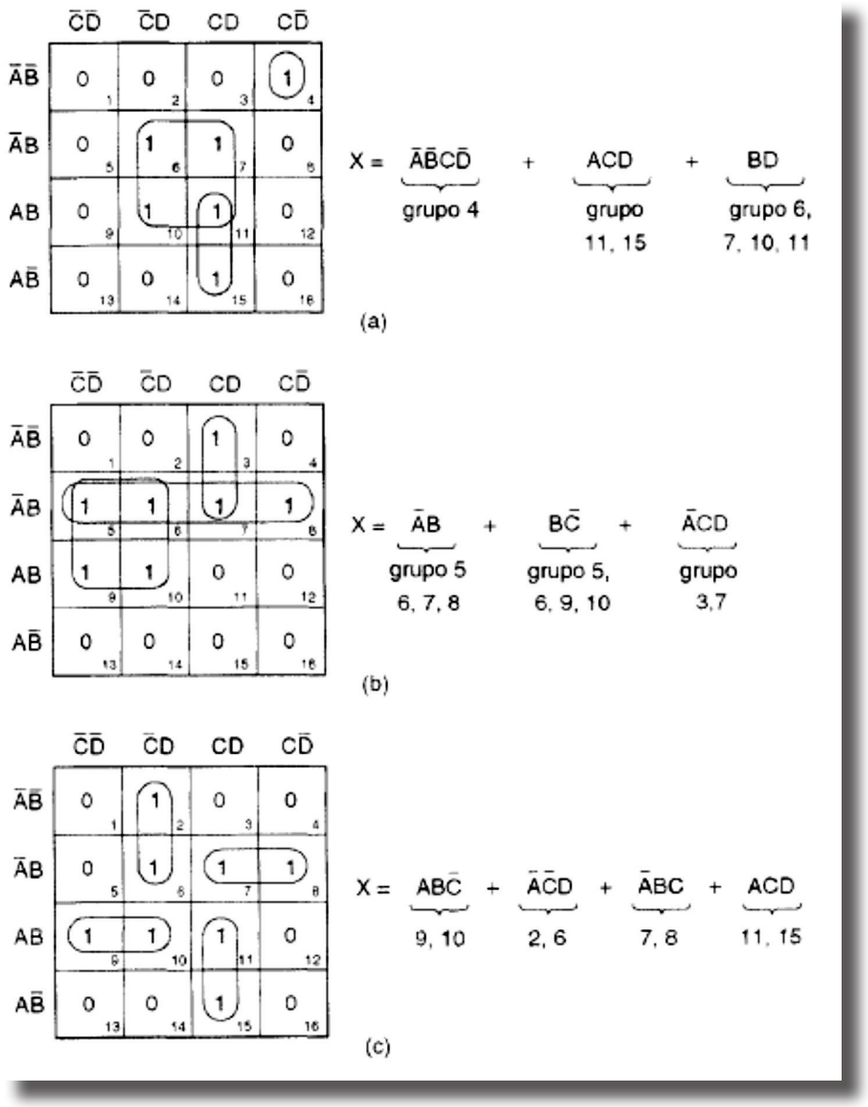


Fig. 38: Exemplo 1, 2 e 3

Exemplo 2:

Considere o mapa K na Fig.38 (b). Mais uma vez presumimos que o passo 1 já foi realizado.

Passo 2: Não existem 1s isolados.

Passo 3: O 1 no quadrado 3 é adjacente apenas ao 1 do quadrado 7. Agrupando este par (grupo 3, 7), produz-se o termo $\bar{A}CD$.

Passo 4: Não existem octetos.

Passo 5: Existem dois quartetos. Os quadrados 5, 6, 7 e 8 formam um quarteto. Reunindo-se este quarteto produz-se o termo $\bar{A}B$. O segundo quarteto é formado pelos quadrados 5, 6,9 e 10. Este quarteto é agrupado porque contem dois quadrados que não tinham sido combinados anteriormente. Este grupo produz $B\bar{C}$.

Passo 6: Todos os 1s já estão agrupados.

Passo 7: Os termos criados pelos três grupos são unidos por um OR para obtermos a expressão para X.

Exemplo 3:

Considere o mapa K na Fig.38 (c).

Passo 2: Não existem 1s isolados.

Passo 3: O 1 no quadrado 2 é adjacente apenas ao 1 no quadrado 6. Este par é agrupado para produzir $\overline{A}\overline{C}D$. Analogamente, o quadrado 9 é adjacente apenas ao quadrado 10. Combinando-se este par produz-se $AB\overline{C}$. Do mesmo modo, o grupo 7, 8 e o grupo 11,15 produzem os termos $\overline{A}BC$ e ACD, respectivamente.

Passo 4: Não existem octetos.

Passo 5: Existe um quarteto formado pelos quadrados 6, 7, 10 e 11. Este quadrado, no entanto, não é combinado, porque todos os 1s no quarteto já foram incluídos em outros grupos.

Passo 6: Todos os 1s já foram agrupados.

Passo 7: A expressão para X está apresentada na figura.

Exemplo 4:

Considere o mapa K na Fig. 39 (a).

Passo 2: Não existem 1s isolados.

Passo 3: Não existe nenhum 1 que seja adjacente a apenas um outro 1.

Passo 4: Não existem octetos.

Passo 5: Não existem quartetos.

Passos 6 e 7: Existem muitos pares possíveis. O processo de agrupar deve usar o mínimo número de grupos para envolver todos os 1s. Para este mapa existem duas possibilidades, que requerem apenas quatro pares envolvidos. A Fig. 39 (a) mostra uma solução e a expressão resultante. A Fig. 39 (b) mostra a outra. Note que, ambas as expressões têm a mesma complexidade, e portanto nenhuma é melhor do que a outra.

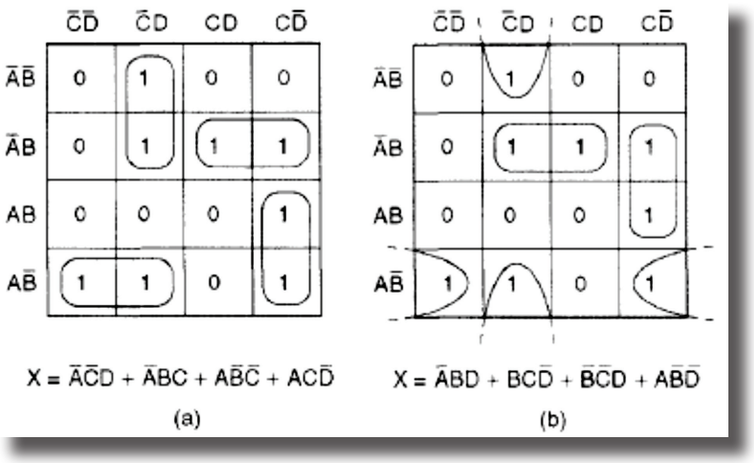


Fig. 39: O mesmo mapa K com duas soluções igualmente boas.

Exemplo 5:

Utilize o mapa K para simplificar a expressão $y = \bar{A}\bar{B}\bar{C} + BC + \bar{A}B$.

Solução:

Neste problema não é apresentada uma tabela de verdade para o preenchimento do mapa K. Em vez disso, devemos preencher o mapa K tomando cada um dos termos produto na expressão e colocando 1s nos quadrados correspondentes.

O primeiro termo, $\bar{A}\bar{B}\bar{C}$, indica que um 1 deve ser colocado no quadrado $\bar{A}\bar{B}\bar{C}$ do mapa (veja a Fig.40). O segundo termo, $\bar{B}C$, indica que um 1 deve ser colocado em cada quadrado que contem $\bar{B}C$ no seu rótulo. Na Fig. 40, isto acontece nos quadrados $A\bar{B}C$ e $\bar{A}\bar{B}C$. Do mesmo modo, o termo $\bar{A}B$ indica que 1 deve ser colocado nos quadrados $\bar{A}BC$ e $\bar{A}B\bar{C}$. Todos os outros quadrados devem ser preenchidos com 0s.

Agora o mapa K pode ser usado para simplificação. O resultado é $y = \bar{A} + \bar{B}C$, como apresentado na figura.

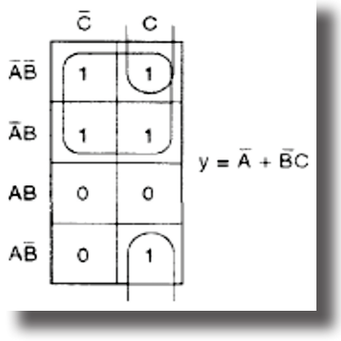


Fig. 45: Exemplo 5

Exercício 1:

Para as seguintes expressões desenhe as suas tabelas de verdade e simplifique-as usando mapas de Karnaugh e desenhe os seus circuitos lógicos.

a) $F = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,C$

b) $F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D + AD\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,D\,\overline{C} + A\,B\,D\,\overline{C} + A\,B\,C\,D + A\,C\,D\,\overline{B}$

Exercício 2:

Extraia as expressões simplificadas dos seguintes mapas de Karnaugh:

a)

F =

C \ AB	$\overline{A}\,\overline{B}$	$\overline{A}B$	$A\overline{B}$	AB
$\overline{C}\,\overline{D}$	1	0	0	1
$\overline{C}D$	0	1	1	0
$C\overline{D}$	0	0	0	0
CD	1	0	0	1

b)

F =

C \ AB	$\overline{A}\,\overline{B}$	$\overline{A}B$	$A\overline{B}$	AB
$\overline{C}$	1	0	1	1
C	1	1	0	1

Bibliografia

CUESTA, L.; PADILLA, A.; REMIRO, F., *Electrónica Digital*. Amadora: McGraw-Hill, 1994.

NUNES, Mário Serafim, *Sistemas Digitais*, 3ª ed.. Lisboa: Editorial Presença, 1989.

RODRIGUES, Pimenta; ARAÚJO, Mário, *Projecto de Sistemas Digitais*, 2ª ed.. Lisboa: Editorial Presença, sd.

TAUB, Herbert, *Circuitos Digitais e Microprocessadores*. S. Paulo: McGrawHill, 1984.