MANUAL DO ALUNO

DISCIPLINA
SISTEMAS
DIGITAIS

Módulos 1, 2 e 3

República Democrática de Timor-Leste
Ministério da Educação

FICHA TÉCNICA

TÍTULO
MANUAL DO ALUNO - DISCIPLINA DE SISTEMAS DIGITAIS
Módulos 1 a 3

AUTOR
JORGE FLÁVIO

COLABORAÇÃO DAS EQUIPAS TÉCNICAS TIMORENSES DA DISCIPLINA
XXXXXXX

COLABORAÇÃO TÉCNICA NA REVISÃO
XXXXXXXXX

DESIGN E PAGINAÇÃO
UNDESIGN - JOAO PAULO VILHENA
EVOLUA.PT

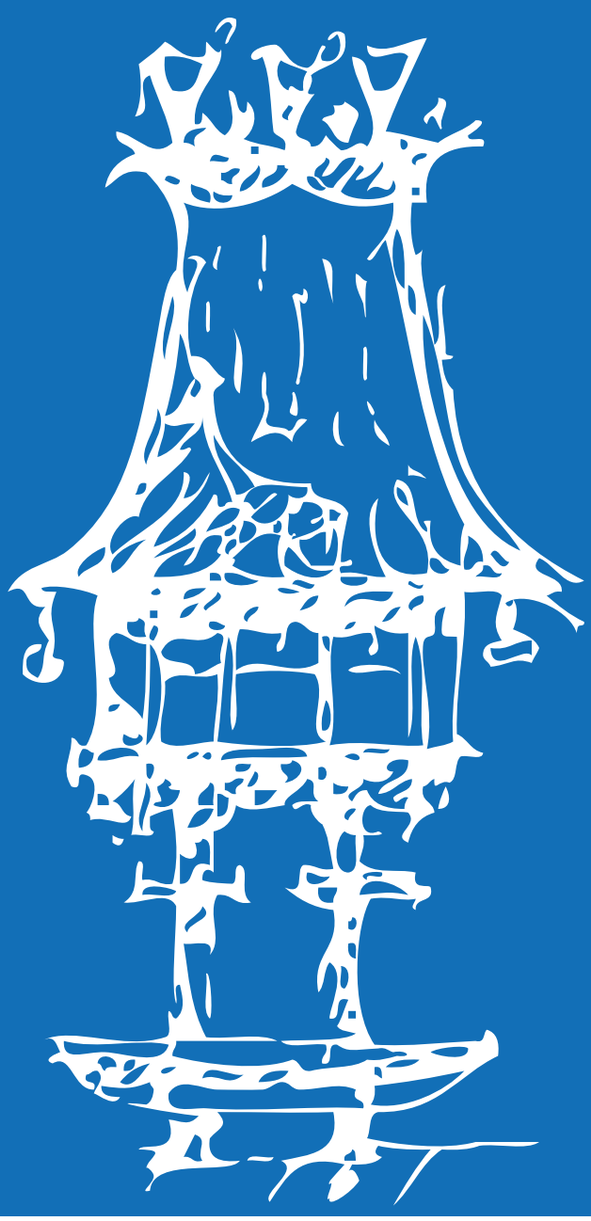
IMPRESSÃO E ACABAMENTO
XXXXXX

ISBN
XXX - XXX - X - XXXXX - X

TIRAGEM
XXXXXXX EXEMPLARES

COORDENAÇÃO GERAL DO PROJETO
MINISTÉRIO DA EDUCAÇÃO DE TIMOR-LESTE
2013

Índice



Sistemas de Numeração

Módulo 1

Apresentação

Este módulo tem carácter mais teórico devendo ser completado com a realização de exercícios de modo que o aluno consolide conhecimentos na área dos sistemas de numeração, da aritmética binárias e dos códigos binários.

Introdução

A abordagem deste módulo de Sistemas de Numeração leva-nos a uma melhor compreensão do funcionamento de vários tipos de aparelhos, que incorporam circuitos que utilizam estas características, existentes no mercado assim como a melhor escolha deste tipo de equipamentos para que se ajuste às crescentes evoluções disponíveis pelas diversas marcas.

Este módulo requer um conhecimento básico de matemática e análise de circuitos eletrónicos assim como a respetiva compreensão desses circuitos.

Objetivos de aprendizagem

- Caracterizar as diferentes bases de numeração.
- Representar números na base decimal, binário e hexadecimal.
- Efetuar a conversão entre decimal e as outras bases e vice-versa, de números inteiros e faccionários.
- Efetuar operações aritméticas em binário.
- Calcular o complemento a dois e a um de um número binário.
- Representar números binários com bit de sinal.
- Efetuar conversões entre o código BCD e o sistema decimal.
- Conhecer a utilização do código ASCII.
- Compreender o sistema de deteção de erros por bit de paridade.

Âmbito de conteúdos

- Sistemas de Numeração:
    - Sistema decimal.
    - Sistema binário.
    - Sistema hexadecimal.
    - Conversão entre sistemas.
- Aritmética Binária:
    - Adição e subtração binárias.
    - Complemento a dois e a um.
    - Representação de um número binário com bit de sinal.
- Códigos binários:
    - BCD
    - Paridade
    - Gray
    - ASCII
    - Deteção de erros através do bit de paridade.

Estrutura de um Sistema de Numeração

Um numeral é um símbolo ou grupo de símbolos que representa um número. Os numerais diferem dos números do mesmo modo que as palavras diferem das coisas a que se referem. Os símbolos «11», «onze» e «XI» são numerais diferentes, representando todos o mesmo número.

Um sistema de numeração, (ou sistema numeral) é um sistema em que um conjunto de números é representado por numerais de uma forma consistente. Pode ser visto como o contexto que permite ao numeral «11» ser interpretado como o numeral romano para dois, o numeral binário para três ou o numeral decimal para onze.

Em condições ideais, um sistema de numeração deve:

- Representar uma grande quantidade de números úteis (Ex: todos os números inteiros, ou todos os números reais);
- Dar a cada número representado uma única descrição (ou pelo menos uma representação padrão);
- Refletir as estruturas algébricas e aritméticas dos números.

Por exemplo, a representação decimal comum dos números inteiros fornece a cada número inteiro uma representação única como uma sequência finita de algarismos, com as operações aritméticas (adição, subtração, multiplicação e divisão) estando presentes como os algoritmos padrões da aritmética. Contudo, quando a representação decimal é usada para os números racionais ou para os números reais, a representação deixa de ser padronizada: muitos números racionais têm dois tipos de numerais, um padrão que tem fim (por exemplo 2,31), e outro que se repete periodicamente (como 2,30999999...).

Sistema Decimal

O sistema decimal é um sistema de numeração de posição que utiliza a base dez.

No sistema decimal existem dez símbolos básicos: 0 1 2 3 4 5 6 7 8 9

Através das combinações adequadas destes símbolos, constroem-se os números do Sistema Decimal. A regra de construção consiste na combinação sequencial dos símbolos, de modo que o valor do número dependa da posição dos símbolos básicos.

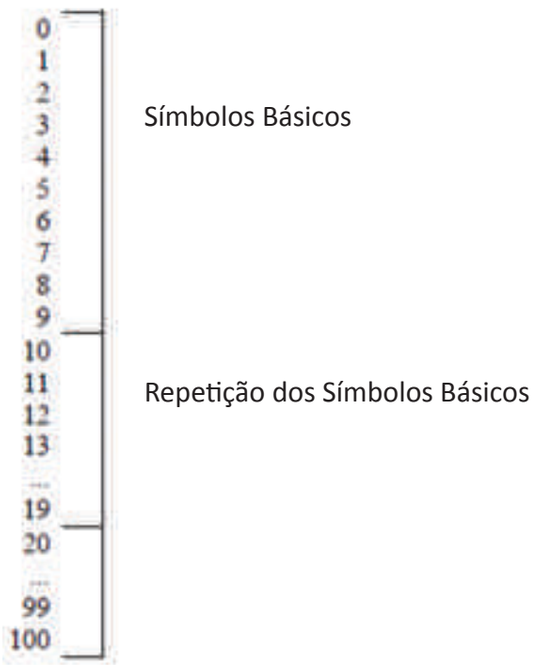


Fig. 1 – Sistema Decimal

No sistema decimal a posição dos símbolos recebe nomes:

Exemplo:

Milhar	Centena	Dezena	Unidade
10^3	10^2	10^1	10^0
3	2	5	9

O número decimal pode ser decomposto da seguinte forma:

Milhar x 10^3 + Centena x 10^2 + Dezena x 10^1 + Unidade x 10^0

No exemplo:

$$3259 = 3 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 9 \times 10^0$$

$$3259 = 3 \times 1000 + 2 \times 100 + 5 \times 10 + 9 \times 1$$

Significa que os números decimais são o somatório dos seus símbolos básicos, multiplicados (cada um) por uma base 10 com os seus expoentes sequenciais. Genericamente:

$$N^0 = S_y \times B^n + S_{y-1} \times B^{n-1} + \ldots + S_1 \times B^1 + S_0 \times B^0$$

Sendo:

Si = símbolo básico (0 < i < y)

n = expoente da base

B = base

O fator S_0 é a unidade, considerado o fator menos significativo e o fator S_y o fator mais significativo.

Os números do sistema decimal também são chamados números na base 10, devido à construção do sistema.

Para diferenciá-los dos números dos outros sistemas de numeração, utiliza-se a seguinte identificação:

$$(Nº.\ decimal)_{10} \rightarrow Cem = (100)_{10}$$

Sistema Binário

A eletrónica digital é baseada na lógica de dois níveis de tensão diferentes. Um sistema eletrónico com mais do que dois níveis de tensão diferentes seria um sistema de alta complexidade, complicando o projeto, as análises e até mesmo a confeção e manutenção deste sistema.

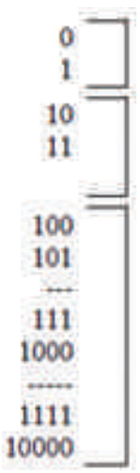
Portanto, perante o número elevado de símbolos, o sistema decimal não é adequado para representar os níveis de tensão na Eletrónica Digital.

O sistema Binário possui dois símbolos: 0 e 1

Através das combinações adequadas, obtém-se a mesma quantidade de números, com as suas respetivas grandezas, do Sistema Decimal. A diferença é apenas na representação dos números.

O sistema Binário é utilizado na lógica da Eletrónica Digital. Os Símbolos deste sistema de numeração representam os dois níveis de tensão utilizados na lógica dos circuitos digitais.

A construção do Sistema Binário é idêntica à do sistema Decimal:



Símbolos Básicos

Repetição dos Símbolos Básicos

O primeiro símbolo da direita é sempre o menos significativo e o primeiro símbolo da esquerda é sempre o mais significativo.

Fig. 2 – Sistema Binário

A representação de um número binário é análoga à de um número Decimal, respeitando-se as Bases:

$$(No.\ Binário)_2 \rightarrow (1001101)_2$$

Binário a Decimal

Dado um número N, binário, para expressá-lo em decimal, deve escrever-se cada número que o compõe (bit), multiplicado pela base do sistema (base = 2), elevado à posição que ocupa. A soma de cada multiplicação de cada dígito binário pelo valor das potências resulta no número real representado.

Exemplo:

$1011_{(2)}$ = _____(10)

$1^3 0^2 1^1 1^0 \rightarrow 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

Portanto, $1010_{(2)} = 11_{(10)}$

Exercícios:

$101100_{(2)}$ = _____ (10)

$1100_{(2)}$ = _____ (10)

$100001_{(2)}$ = _____ (10)

$1010_{(2)}$ = _____ (10)

$1111_{(2)}$ = _____ (10)

$10010_{(2)}$ = _____ (10)

$1001_{(2)}$ = _____ (10)

$11101_{(2)}$ = _____ (10)

$110111_{(2)}$ = _____ (10)

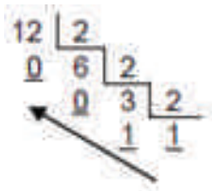
$10110_{(2)}$ = _____ (10)

$11011_{(2)}$ = _____ (10)

$101001_{(2)}$ = _____ (10)

Decimal a Binário

Dado um número decimal, para convertê-lo em binário, basta dividi-lo sucessivamente por 2, anotando o resto da divisão inteira (da direita para a esquerda):



Portanto, $12_{(10)} = 1100_{(2)}$

Exercícios:

$22_{(10)} =$ _____ (2)

$56_{(10)} =$ _____ (2)

$54_{(10)} =$ _____ (2)

$35_{(10)} =$ _____ (2)

$26_{(10)} =$ _____ (2)

$3_{(10)} =$ _____ (2)

$69_{(10)} =$ _____ (2)

$17_{(10)} =$ _____ (2)

$13_{(10)} =$ _____ (2)

$36_{(10)} =$ _____ (2)

$42_{(10)} =$ _____ (2)

$71_{(10)} =$ _____ (2)

$25_{(10)} =$ _____ (2)

$102_{(10)} =$ _____ (2)

Soma de números binários

Recordando as seguintes somas básicas:

1. 0+0 = 0
2. 0+1 = 1
3. 1+1 = 10

Assim, ao somar-se 100110101 com 11010101, tem-se:

	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	
		1_{+1}	0_{+1}	0_{+1}	1_{+1}	1	0_{+1}	1	0_{+1}	1	Transporte
+			1	1	0	1	0	1	0	1	
	1	0	0	0	0	0	1	0	1	0	

Opera-se como em decimal: começa-se a somar desde a direita, no exemplo, 1+1=10, então escreve-se 0 «e vai» 1. Soma-se este 1 à coluna seguinte.

Subtração de números binários

		bit 5	bit 4	bit 3	bit 2	bit 1	
Diminuendo		1	1	0	0	1	
Diminuidor	-	0_{+1}	1_{+1}	0_{+1}	1_{+1}	1	Transporte
Resultado:		0	1	1	1	0	

A subtração bit a bit faz-se da seguinte forma:

Bit 1: 1 - 1 = 0 (transporte 0)

Bit 2: 0 - 1 = 1 (transporte 1)

Bit 3: soma-se o diminuidor com o transporte, isto é,

0 + 1 = 1 (transporte da soma é 0). Depois realiza-se a subtração do diminuendo com o resultado da soma.

0 - 1 = 1 (transporte 1)

Bit 4: soma-se o diminuidor com o transporte, isto é,

1 + 1 = 0 (transporte da soma é 1). Faz-se a subtração do diminuendo com o resultado da soma, isto é,

1 – 0 = 1 (transporte 0)

Bit 5: soma-se o diminuidor com o transporte, isto é,

0 + 1 = 1 (transporte da soma é 0). Faz-se a subtração do diminuendo com o resultado da soma, isto é,

1 – 1 = 0 (transporte 0)

Verificar:

$11001_{(2)} = 25_{(10)}$

$- 1011_{(2)} = 11_{(10)}$

$0110_{(2)} = 14_{(10)}$

Sistema Hexadecimal

Todos os sistemas digitais trabalham com os Números Binários, porém, em computação, por exemplo, as informações são grupos de 8, 16 ou 32 dígitos binários, que combinados formam as instruções e os dados inteligíveis ao sistema. Devido ao grande número de dígitos, a Linguagem de Máquina torna complexo o entendimento entre o homem e a máquina. Para suprir esta complexidade, ou pelo menos atenuá-la, um sistema de numeração de muitos símbolos e ao mesmo tempo proporcional aos grupos de dígitos, seria o adequado, pois diminuiria o número de dígitos da informação, sem alterá-la. Este sistema é conhecido como Sistema Hexadecimal.

O sistema Hexadecimal possui 16 símbolos: 0 1 2 3 4 5 6 7 8 9 A B C D E F

As Letras de A a F, para efeito de conversão, corresponderão aos decimais de 10 a 15 respetivamente.

A construção do sistema Hexadecimal é idêntica a qualquer outro sistema de numeração, consiste na combinação ordenada sequencial dos símbolos básicos de modo que o valor do número depende da posição dos símbolos.

Os números do sistema Hexadecimal também são conhecidos como números na base 16.

A representação dos números Hexadecimais é análoga aos outros sistemas de numeração:

N.º Hexadecimal $_{(16)}$ → C12FA $_{(16)}$

Hexadecimal a Decimal

A base hexadecimal tem mais vantagem do que a octal, pois representa um número com grande quantidade de bits, numa forma simples e reduzida. Por exemplo, o número binário 1001110100 110110 $_{(2)}$ = 9D36 $_{(16)}$.

A base hexadecimal é formada por 16 elementos. Como a base dez apenas tem 10 símbolos, os restantes 6 símbolos são representados pelas primeiras 6 letras do nosso alfabeto: A, B, C, D, E, F.

Para obtermos o equivalente decimal do número $9D36_{(16)}$ na base hexadecimal temos que executar as seguintes operações:

$9D36_{(16)} =$ ———(10)

$9^3D^2 3^1 6^0$ (D=13) → $9 \times 16^3 + 13 \times 16^2 + 3 \times 16^1 + 6 \times 16^0 = 40246_{(10)}$

Exercícios:

$2C_{(16)} =$ ——— (10)

$C_{(16)} =$ ——— (10)

$21_{(16)} =$ ——— (10)

$A_{(16)} =$ ——— (10)

$F_{(16)} =$ ——— (10)

$12_{(16)} =$ ——— (10)

$22_{(16)} =$ ——— (10)

$9_{(16)} =$ ——— (10)

$1D_{(16)} =$ ——— (10)

$37_{(16)} =$ ——— (10)

$16_{(16)} =$ ——— (10)

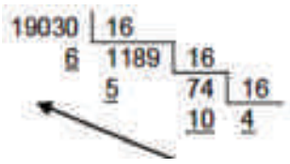
$1B_{(16)} =$ ——— (10)

$29_{(16)} =$ ——— (10)

$18_{(16)} =$ ——— (10)

Decimal a Hexadecimal

Dado um número decimal, para convertê-lo em hexadecimal, basta dividi-lo sucessivamente por 16, anotando o resto da divisão inteira (da direita para a esquerda):



Portanto, $19030_{(10)} = 4A56_{(16)}$

Exercícios:

$12_{(10)} = C_{(16)}$

$16_{(10)} = 10_{(16)}$

$23_{(10)} = 17_{(16)}$

$26_{(10)} = 1A_{(16)}$

$41_{(10)} = 29_{(16)}$

$11_{(10)} = B_{(16)}$

$10_{(10)} = A_{(16)}$

$54_{(10)} = 36_{(16)}$

$52_{(10)} = 34_{(16)}$

$64_{(10)} = 40_{(16)}$

$21_{(10)} = 15_{(16)}$

$47_{(10)} = 2F_{(16)}$

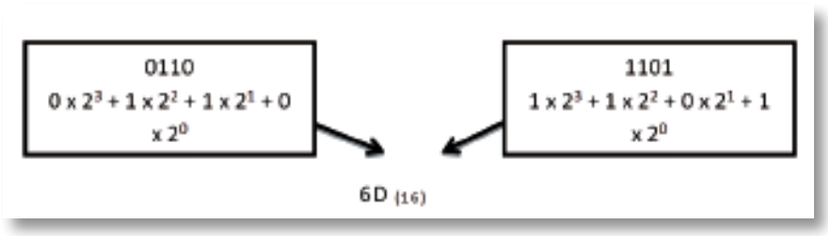
$25_{(10)} = 19_{(16)}$

$19_{(10)} = 13_{(16)}$

Binário-Hexadecimal e vice-versa

Utiliza-se o princípio de que para escrever um dígito em hexadecimal chegam 4 dígitos em binário, dada a relação entre as bases respetivas ser a potência 4, isto é, $16 = 2^4$.

Por exemplo, dado o número $01101101_{(2)}$, podemos efetuar a seguinte conversão:



Fazendo a operação inversa chegamos da base hexadecimal à base binária:

2	4	A	8	Hexadecimal
↓	↓	↓	↓	
0010	0100	1010	1000	Binário

O equivalente na base binária é: $24A8_{(16)} = 1001001010_{(2)}$

Exercícios:

$101100_{(2)}$ = _____ (16)

$1100_{(2)}$ = _____ (16)

$100001_{(2)}$ = _____ (16)

$1010_{(2)}$ = _____ (16)

$1111_{(2)}$ = _____ (16)

$10010_{(2)}$ = _____ (16)

$16_{(16)}$ = _____ (2)

$38_{(16)}$ = _____ (2)

$36_{(16)}$ = _____ (2)

$23_{(16)}$ = _____ (2)

$1A_{(16)}$ = _____ (2)

$3_{(16)}$ = _____ (2)

$45_{(16)}$ = _____ (2)

$1001_{(2)}$ = _____ (16)

$11101_{(2)}$ = _____ (16)

$110111_{(2)}$ = _____ (16)

$10110_{(2)}$ = _____ (16)

$11011_{(2)}$ = _____ (16)

$101001_{(2)}$ = _____ (16)

$11_{(16)}$ = _____ (2)

$D_{(16)}$ = _____ (2)

$24_{(16)}$ = _____ (2)

$2A_{(16)}$ = _____ (2)

$47_{(16)}$ = _____ (2)

$19_{(16)}$ = _____ (2)

$66_{(16)}$ = _____ (2)

Sistema Octal

O sistema Octal assemelha-se ao sistema Hexadecimal e também é muito utilizado na área de digital.

O sistema Octal possui 8 símbolos: 0 1 2 3 4 5 6 7

A construção do sistema Octal é idêntica aos demais sistemas de numeração.

Os números do sistema Octal também são conhecidos como números na base 8.

A representação dos números Octais é análoga aos outros sistemas de numeração:

N.º Octal $_{(8)} \rightarrow 7123_{(8)}$

Octal a Decimal

A base octal utiliza oito algarismos ou dígitos: 0, 1, 2, 3, 4, 5, 6, 7; por isso se diz que a base deste sistema de numeração é oito (octal) e cada dígito também tem um valor posicional. Para obtermos o equivalente decimal do número $24643701_{(8)}$ na base octal temos que executar as seguintes operações.

$24643701_{(8)} = \_\_\_\_\_(10)$

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

$2 \times 8^7 + 4 \times 8^6 + 6 \times 8^5 + 4 \times 8^4 + 3 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = 5457857_{(10)}$

Exercícios:

$54_{(8)} = \_\_\_\_\_ (10)$

$14_{(8)} = \_\_\_\_\_ (10)$

$41_{(8)} = \_\_\_\_\_ (10)$

$12_{(8)} = \_\_\_\_\_ (10)$

$17_{(8)} = \_\_\_\_\_ (10)$

$22_{(8)} = \_\_\_\_\_ (10)$

$42_{(8)} = \_\_\_\_\_ (10)$

$11_{(8)} = \_\_\_\_\_ (10)$

$35_{(8)} = \_\_\_\_\_ (10)$

$67_{(8)} = \_\_\_\_\_ (10)$

$26_{(8)} = \_\_\_\_\_ (10)$

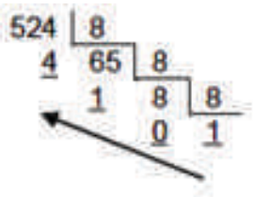
$33_{(8)} = \_\_\_\_\_ (10)$

$51_{(8)} = \_\_\_\_\_ (10)$

$30_{(8)} = \_\_\_\_\_ (10)$

Decimal a Octal

Dado um número decimal, para convertê-lo em octal, basta dividi-lo sucessivamente por 8, anotando o resto da divisão inteira (da direita para a esquerda):



Portanto, $524_{(10)} = 1014_{(8)}$

Exercícios:

$43_{(10)} =$ _____ (8)

$16_{(10)} =$ _____ (8)

$12_{(10)} =$ _____ (8)

$19_{(10)} =$ _____ (8)

$23_{(10)} =$ _____ (8)

$15_{(10)} =$ _____ (8)

$28_{(10)} =$ _____ (8)

$31_{(10)} =$ _____ (8)

$53_{(10)} =$ _____ (8)

$65_{(10)} =$ _____ (8)

$22_{(10)} =$ _____ (8)

$34_{(10)} =$ _____ (8)

$36_{(10)} =$ _____ (8)

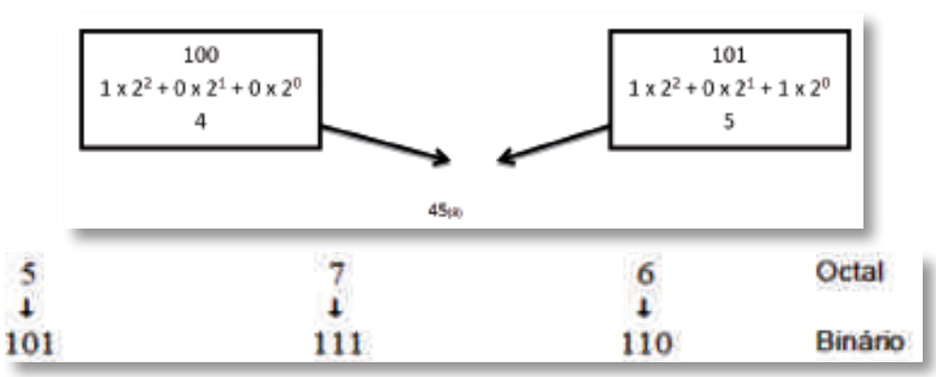
$45_{(10)} =$ _____ (8)

Binário-Octal e vice-versa

Utiliza-se o princípio de que para escrever cada dígito octal são necessários somente 3 dígitos binários, visto a relação entre as bases respetivas ser uma potência 3, isto é, $8 = 2^3$. O maior dígito em octal corresponde ao dígito 7: $111_{(2)} = 7_{(8)}$.

Dado o número $100101_{(2)}$, podemos realizar a seguinte conversão:

Fazendo a operação inversa chegamos da base octal à base binária:



O equivalente na base binaria é $576_{(8)} = 101111110_{(2)}$

Exercícios:

$101100_{(2)}$ = _____ (8)

$1100_{(2)}$ = _____ (8)

$100001_{(2)}$ = _____ (8)

$1010_{(2)}$ = _____ (8)

$1111_{(2)}$ = _____ (8)

$10010_{(2)}$ = _____ (8)

$1001_{(2)}$ = _____ (8)

$11101_{(2)}$ = _____ (8)

$110111_{(2)}$ = _____ (8)

$10110_{(2)}$ = _____ (8)

$11011_{(2)}$ = _____ (8)

$101001_{(2)}$ = _____ (8)

$16_{(8)}$ = _____ (2)

$70_{(8)}$ = _____ (2)

$66_{(8)}$ = _____ (2)

$43_{(8)}$ = _____ (2)

$32_{(8)}$ = _____ (2)

$3_{(8)}$ = _____ (2)

$105_{(8)}$ = _____ (2)

$21_{(8)}$ = _____ (2)

$15_{(8)}$ = _____ (2)

$44_{(8)}$ = _____ (2)

$52_{(8)}$ = _____ (2)

$107_{(8)}$ = _____ (2)

$31_{(8)}$ = _____ (2)

$146_{(8)}$ = _____ (2)

Conversão por decomposição em potências de base

Base N → Decimal

Para converter um número de uma base qualquer em um número decimal, utiliza-se a fórmula genérica, já vista:

$$N^0 = S_y \times B^n + S_{y-1} \times B^{n-1} + \ldots + S_1 \times B^1 + S_0 \times B^0$$

Ou seja:

$$\sum_{i=0}^{n=1} a_i . b^i$$

Onde:

b : é a base

i : a casa onde está o algarismo

$\mathbf{a_i}$: é o algarismo da casa i

n : numero total de algarismos

Exemplos:

$1101)_{(2)} = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 8 + 4 + 0 + 1 = 13$

$1101)_{(2)} = 13_{(10)}$

$10101)_{(2)} = (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 16 + 0 + 4 + 0 + 1 = 21$

$10101_{(2)} = 21_{(10)}$

$2A_{(16)} = 2 \times 16^1 + A \times 16^0 = 2 \times 16^1 + 10 \times 16^0 = 32 + 10 = 42$ (NOTA: A=10)

$2A_{(16)} = 42_{(10)}$

$B1_{(16)} = B \times 16^1 + 1 \times 16^0 = 11 \times 16^1 + 1 \times 16^0 = 176 + 16 = 192$ (B=16)

$B1_{(16)} = 192_{(10)}$

$75_{(8)} = 7 \times 8^1 + 5 \times 8^0 = 56 + 5 = 61$

$75_{(8)} = (61)_{(10)}$

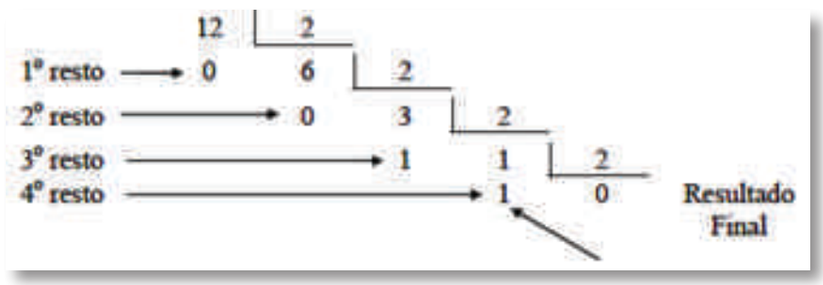
Conversão pela divisão pela base

Decimal → Base

A conversão de números decimais para números de outra base é feita dividindo-se o número decimal pela base até que o resultado seja zero. Os «restos» das divisões no sentido da última divisão para a primeira formam o número convertido.

Exemplo 1: $12_{(10)} \rightarrow ???_{(2)}$

Solução:

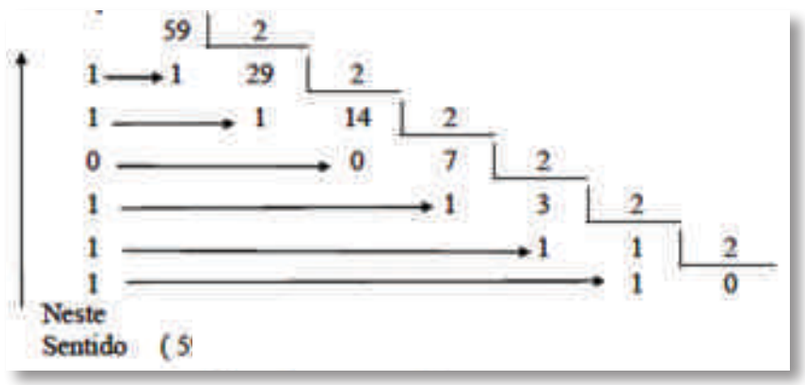


X_2	4º resto	3º resto	2º resto	1º resto
X_2	1	1	0	0

Portanto: $12_{(10)} = 1100_{(2)}$

Exemplo 2: $59_{(10)} \rightarrow ???_{(2)}$

Solução:



Portanto: $58_{(10)} = 111011_{(2)}$

Exemplo 3: $58_{(10)} \rightarrow ???_{(16)}$

Solução:

X_2	2º resto	1º resto
X_2	3	A

Portanto: $(58)_{10} = (3A)_{16}$

Conversão de base B para B^N

Binário → Hexadecimal, Binário → Octal

A conversão B para B^N é feita transformando-se grupos de n dígitos binários, no sentido da direita para a esquerda, e convertendo-os diretamente em números da base B^N

Exemplo 1: $10100110_{(2)} \rightarrow ???_{(16)}$

Solução:

$16 = 2^4$ logo n = 4

1010	0110
A	6

Portanto: $10100110_{(2)} \rightarrow A6_{(16)}$

Exemplo 2: $110011_{(2)} = ???_{(16)}$

Solução:

$16 = 2^4$ logo n = 4

0011	0011
3	3

Portanto: $110011_{(2)} = 33_{(16)}$

Caso o último grupo à esquerda não possua 4 dígitos, deve completar-se com zeros.

Exemplo 3: $110011_{(2)} \rightarrow ???_{(8)}$

Solução:

$8 = 2^3$ logo n = 3

110	011
6	3

Portanto: $110011_{(2)} = 63_{(8)}$

Conversão de base B^N para B

Hexadecimal → Binário, Octal → Binário

A conversão de números de base B^N para base B é feita transformando-se os símbolos da base B^N diretamente em n dígitos da base B.

Exemplo 1: $10D_{(16)} \rightarrow ???_{(2)}$

Solução:

$16 = 2^4$ logo n = 4

1	0	D
0001	0000	1101

Portanto: $10D_{(16)} = 000100001101_{(2)}$ ou $10D_{(16)} = 100001101_{(2)}$

Os zeros à esquerda do último grupo da esquerda podem ser omissos.

Exemplo 2: $C59_{(16)} \rightarrow ???_{(2)}$

Solução:

$16 = 2^4$ logo n = 4

C	5	9
1100	0101	1001

Portanto: $C59_{(16)} = 110001011001_{(2)}$

Exemplo 3: $743_{(8)} \rightarrow ???_{(2)}$

Solução:

$8 = 2^3$ logo n = 3

7	4	3
111	100	011

Portanto: $743_{(8)} = 111100011_{(2)}$

Conversão de base B^N para B^M

Hexadecimal → Octal

A conversão de números de base B^N para base B^M é feita transformando-se os símbolos da base B^N primeiramente para base B e após para a base B^M.

Exemplo 1: $10D_{(16)} \rightarrow ???_{(8)}$

Solução:

$16 = 2^4$ logo n = 4;

1	0	D
0001	0000	1101

$8 = 2^3$ logo m = 3

Portanto: $10D_{(16)} = 000100001101_{(2)}$ ou $10D_{(16)} = 100001101_{(2)}$

100	001	101
4	1	5

Portanto: $100001101_{(2)} = 415_{(8)}$

Solução: $10D_{(16)} = 100001101_{(2)} = 415_{(8)}$

Tabela de Equivalência entre Sistemas de Numeração

Decimal	Binário	Hexadecimal	Octal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Números negativos

Os computadores lidam com números positivos e números negativos, logo, é necessário encontrar uma representação para números com sinal negativo. Uma possibilidade é inverter todos os bits de um número para representar o número correspondente com sinal negativo. Esta representação é designada por complemento para um.

Exemplo 1:

$100_{(10)} = 01100100_{(2)}$ utilizando 8 bits.

Invertendo todos os bits obtemos:

$-100_{(10)} = 10011011_{(2)}$

O problema desta representação é que existem 2 padrões de bits para o 0. Nomeadamente $0_{(10)} = 00000000_{(2)} = 11111111_{(2)}$. A solução encontrada consiste em representar os números em complemento para 2. Para determinar o negativo de um número negam-se todos os seus bits e soma-se uma unidade.

Exemplo 2:

$100_{(10)} = 01100100_{(2)}$ utilizando 8 bits.

Invertendo todos os *bits* obtemos:

$10011011_{(2)}$

Somando uma unidade:

$10011011_{(2)} + 1 = 10011100_{(2)} = -100_{(10)}$

A representação em complemento para 2 tem as seguintes características:

- O bit da esquerda indica o sinal;
- O processo indicado no parágrafo anterior serve para converter um número de positivo para negativo e de negativo para positivo;

- O 0 tem uma representação única: todos os bits a 0;
- A gama de valores que é possível representar com bits é (-2^{n-1} ... $2^{n-1}-1$).

Exemplo 3:

Qual o número representado por $11100100_{(2)}$?

Como o bit da esquerda é 1 este número é negativo.

Vamos inverter:

$$00011011_{(2)}$$

Somando uma unidade:

$$00011011_{(2)} + 1 = 00011100_{(2)} = 28_{(10)}$$

Logo:

$$11100100_{(2)} = -\ 28_{(10)}$$

Exemplo: Representar os seguintes números com 16 *bits*.	
0011101010 Positivo, logo: 0000000011101010	11011110 Negativo, logo: 1111111111011110

Bibliografia

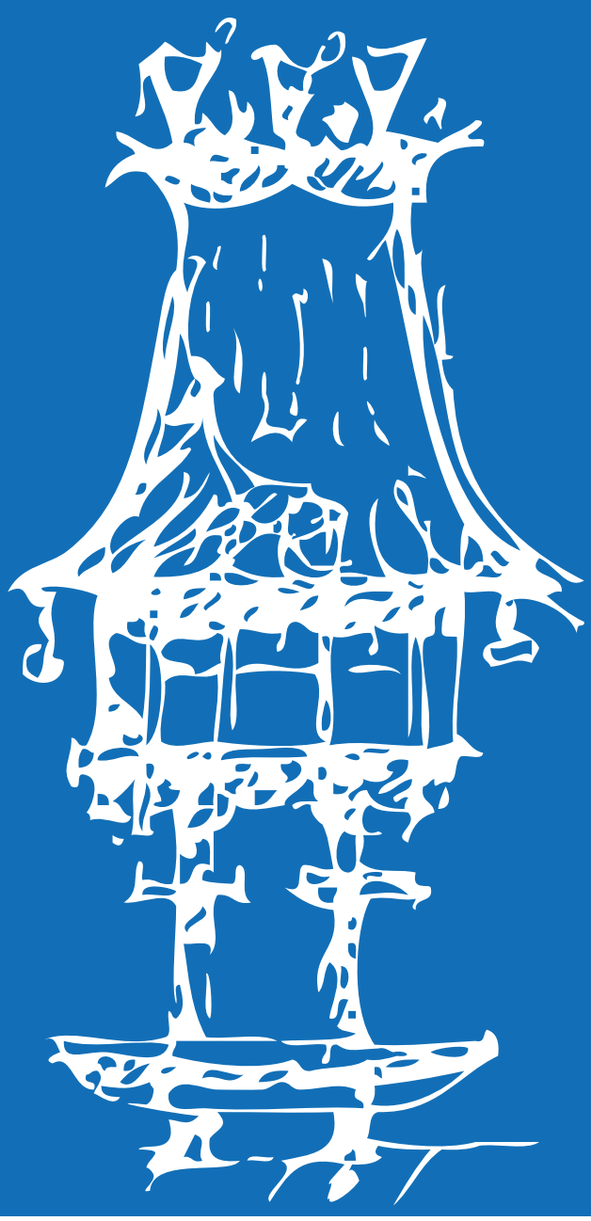
PADILHA, António e outros, *Electrónica Digital*. McGrawHill. (s.d.).

PADILHA, António, *Sistemas Digitais*. McGrawHill. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Analógicos e Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

Notas



Circuitos Lógicos

Módulo 2

Apresentação

Este módulo tem carácter teórico-prático, devendo decorrer em ambiente laboratorial de modo que o aluno possa verificar e comprovar a tabela da verdade das portas lógicas e de circuitos lógicos elementares.

Introdução

A abordagem deste módulo de Circuitos Lógicos leva-nos a uma melhor compreensão do funcionamento de vários tipos de aparelhos, que incorporam circuitos que utilizam estas características, existentes no mercado assim como a melhor escolha deste tipo de equipamentos para que se ajuste às crescentes evoluções disponíveis pelas diversas marcas.

Este módulo requer um conhecimento básico de matemática e análise de circuitos eletrónicos assim como a respetiva compreensão desses circuitos.

Objetivos de aprendizagem

- Álgebra de Boole e funções lógicas:
    - Compreender a noção de estado lógico, variável lógica e nível lógico.
    - Representar as funções lógicas através de tabelas de verdade.
    - Desenhar o logigrama a partir da expressão lógica e vice-versa.
    - Conhecer os postulados e teoremas da Álgebra de Boole.
    - Simplificar funções lógicas através dos teoremas e postulados da Álgebra de Boole e pelo método de Karnaugh.
    - Desenhar circuitos de lógica combinatória a partir da tabela de verdade ou da expressão de saída.
- Portas Lógicas:
    - Identificar os símbolos das portas lógicas.
    - Conhecer o funcionamento das portas lógicas básicas.
    - Reconhecer a universalidade das portas NAND e NOR.
    - Utilizar portas NAND e NOR para implementar qualquer função lógica.

- Famílias Lógicas:
    - Conhecer as características das famílias lógicas mais usadas nos circuitos digitais (TTL e CMOS).

Âmbito de conteúdos

- *Álgebra de Boole.*
- Funções Lógicas.
- Portas Lógicas.
- Famílias Lógicas.

Portas Lógicas e Álgebra Booleana

Como foi mencionado no módulo 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 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 permite descrever a relação entre a(s) saída(s) de um circuito lógico e suas entradas, através de uma equação (expressão booleana). Neste módulo, 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 que permita que este circuito possa ser reconstruído, 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 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.

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 mais fácil do que a álgebra convencional).

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 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 cúbicas, logaritmos, números imaginários e assim por diante.

Na verdade, na álgebra booleana existem apenas três operações básicas: OR (OU), AND (E) e NOT (NAO).

Essas operações básicas são chamadas operações lógicas. Circuitos digitais chamados portas lógicas podem ser construídos a partir de díodos, transístores e resistências ligadas 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 estas 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 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 dependera 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 binária, assim, torna-se bastante simples escrever todas as combinações possíveis sem esquecer nenhuma.

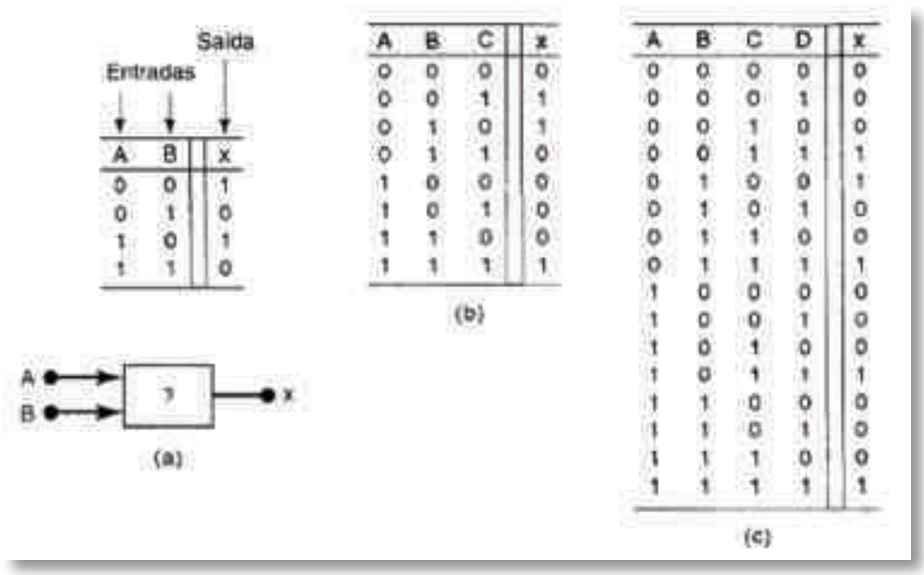


Fig. 1: *(a) Exemplo de tabelas de verdade para circuitos de duas entradas, (b) de três entradas e (c) de quatro entradas.*

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 considerarem 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 do que 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.

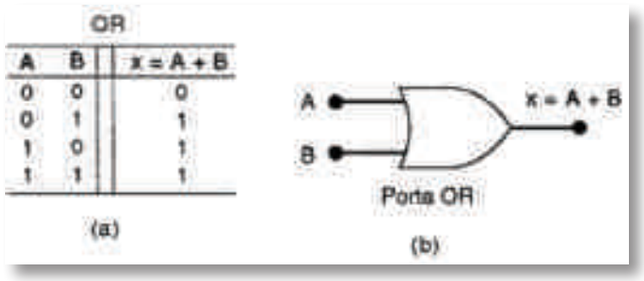


Fig. 2: (a) Tabela de verdade que define a operação OR; (b) símbolo para uma porta OR de duas entradas.

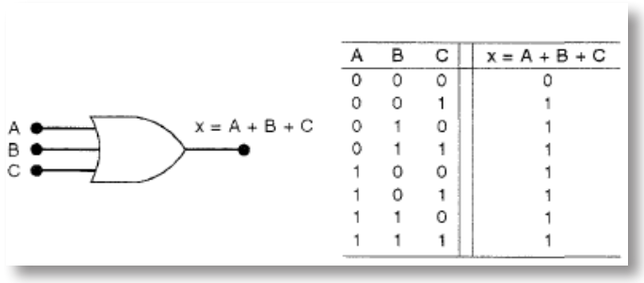
Porta OR

Em circuitos digitais, uma porta OR é um circuito que possui duas ou mais entradas e cuja saída é 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. Noutras 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, então, 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 a operação OR e as 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.

Exercício 1:

Em muitos sistemas industriais de controlo, 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 ativo 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 ou a temperatura ou a pressão seja 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.

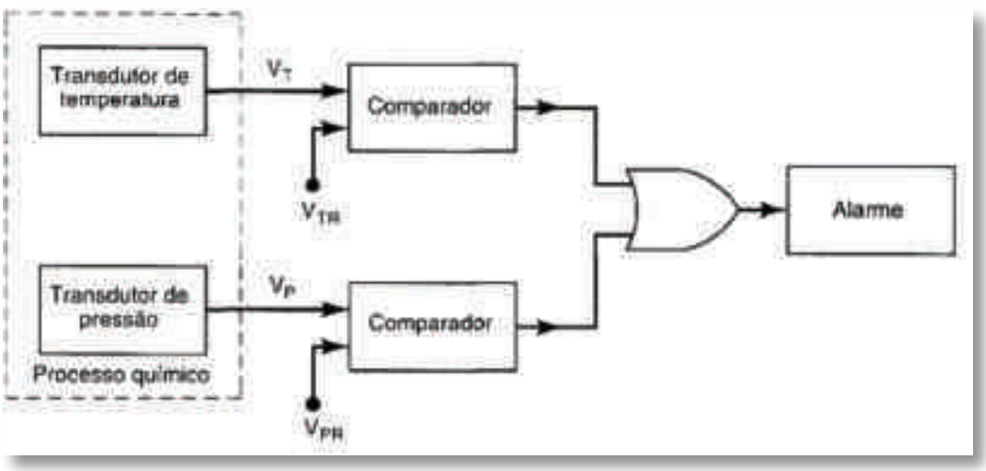


Fig. 4: Exemplo de utilização da porta OR em um sistema de alarme

Exercício 2:

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 por sucessivamente.

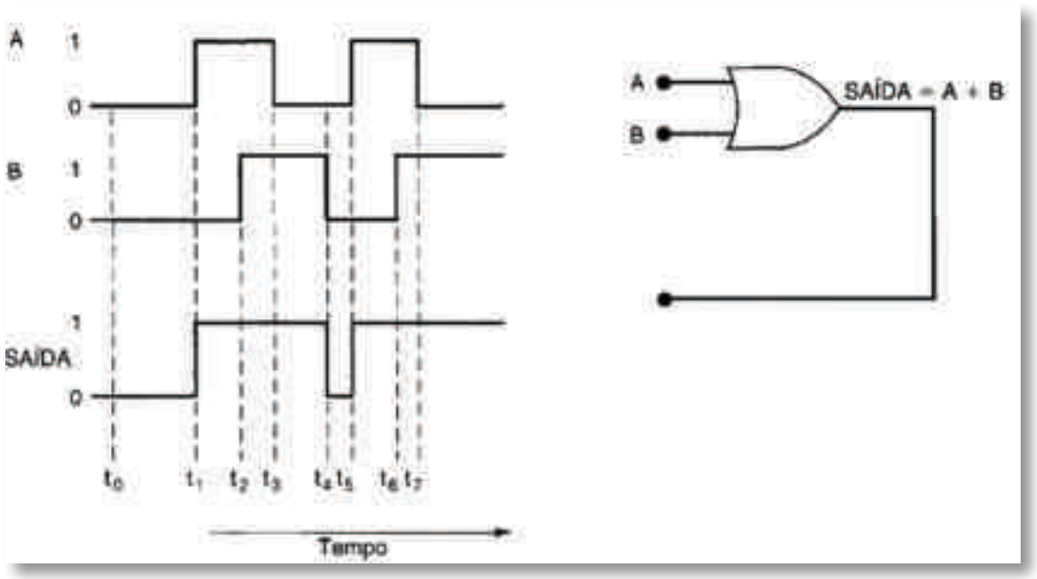


Fig. 5: Exemplo 2

Exemplo 3A:

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

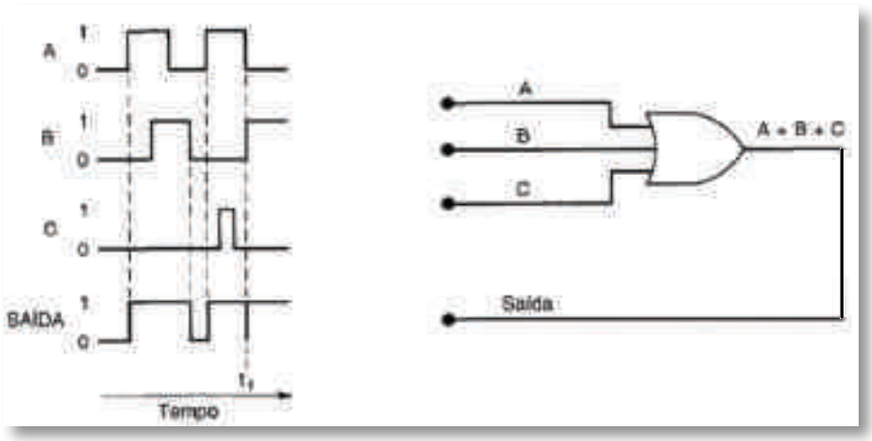


Fig. 6: Exemplos 3A e 3B

Exemplo 3B:

O que aconteceria ao glitch 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* . *B*

Nesta expressão, o sinal • expressa a operação AND, e não a multiplicação normal. Entretanto, 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 do que duas entradas. Por exemplo, quando a operação AND é realizada sobre três entradas, temos x = A.B.C = ABC. O único momento em que pode ser igual a 1 é quando A = B = C = 1.

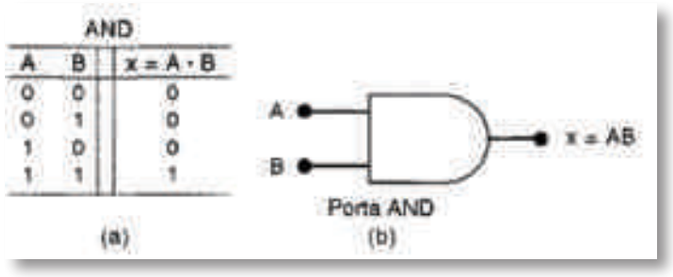


Fig. 7: (a) Tabela de verdade para a operação AND; (b) símbolo da porta AND.

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. Noutras 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 do que 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 OR, isto significa que a saída estará em ALTO quando qualquer uma das entradas estiver em ALTO.

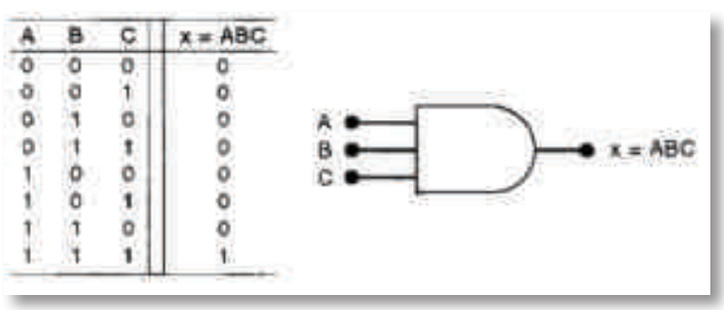


Fig. 8: Tabela de verdade e o símbolo para uma porta AND de três entradas.

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.

Exemplo 1:

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

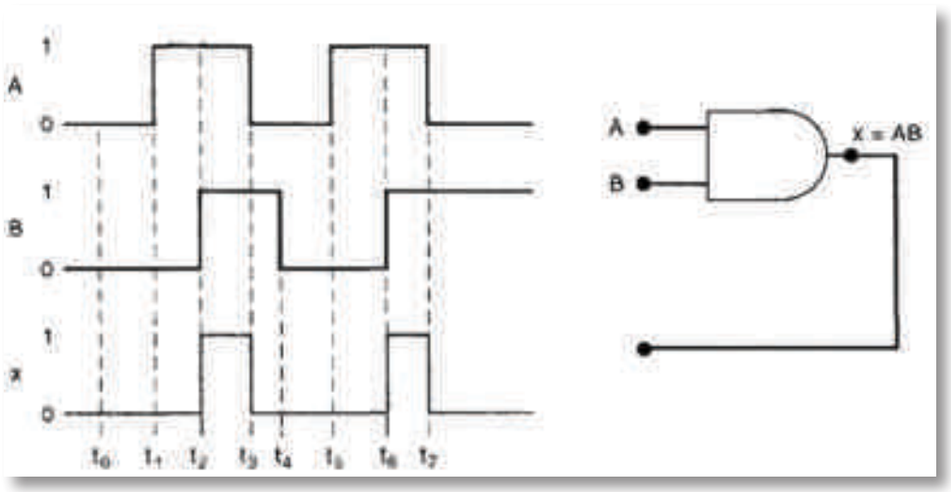


Fig. 9: Exemplo 1

Exemplo 2A:

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

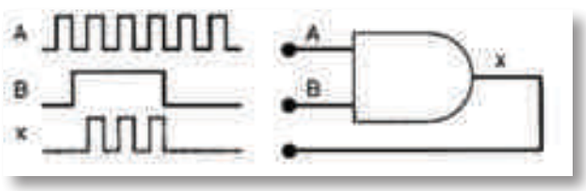


Fig. 10: Exemplo 2A e 2B

Exemplo 2B:

O que acontecera 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 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:

$\overline{1}$; Porque NOT 1 é 0

$\overline{0}$; Porque NOT 0 é 1

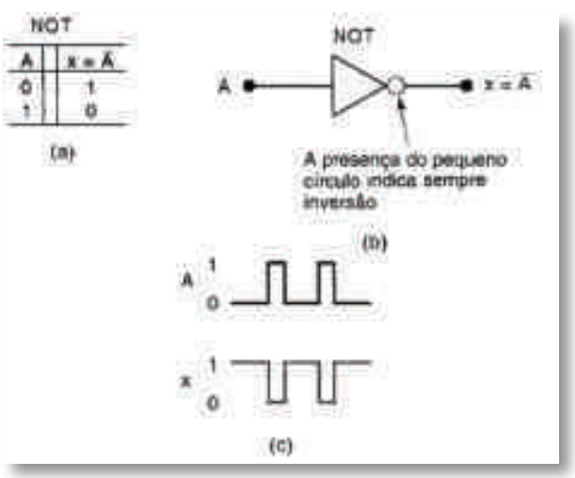


Fig. 11: (a) Tabela de verdade; (b) símbolo para o INVERSOR (NOT); (c) formas de onda.

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 da 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 na tabela 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	
1 + 1 = 1	1 . 1 = 1	$\overline{0} = 1$

Tabela 2 – Operações

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 parenteses na expressão, neste caso, a operação dentro dos parenteses é realizada primeiro. Esta é a mesma regra usada na álgebra comum para determinar a ordem das operações.

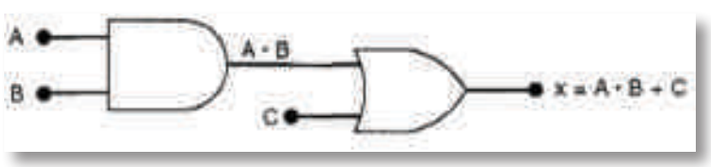


Fig. 12:Circuito lógico com a sua expressão booleana.

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.

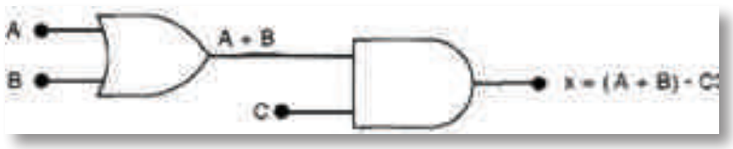


Fig. 13: Circuito lógico cuja expressão requer parenteses.

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 é primeiramente invertido e depois é feita uma operação OR com B.

Na Fig.14 (b), a saída da porta OR e 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.

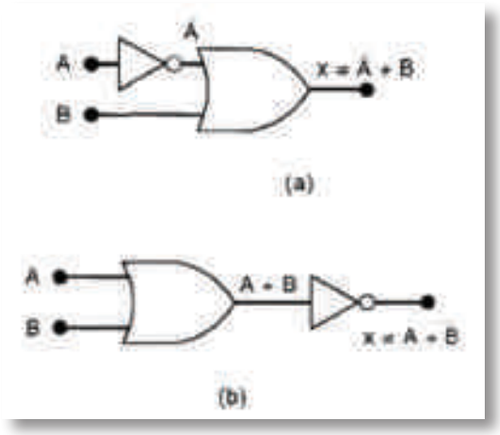


Fig. 14: Circuitos que usam inversores

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 esta ligada como entrada em duas portas diferentes.

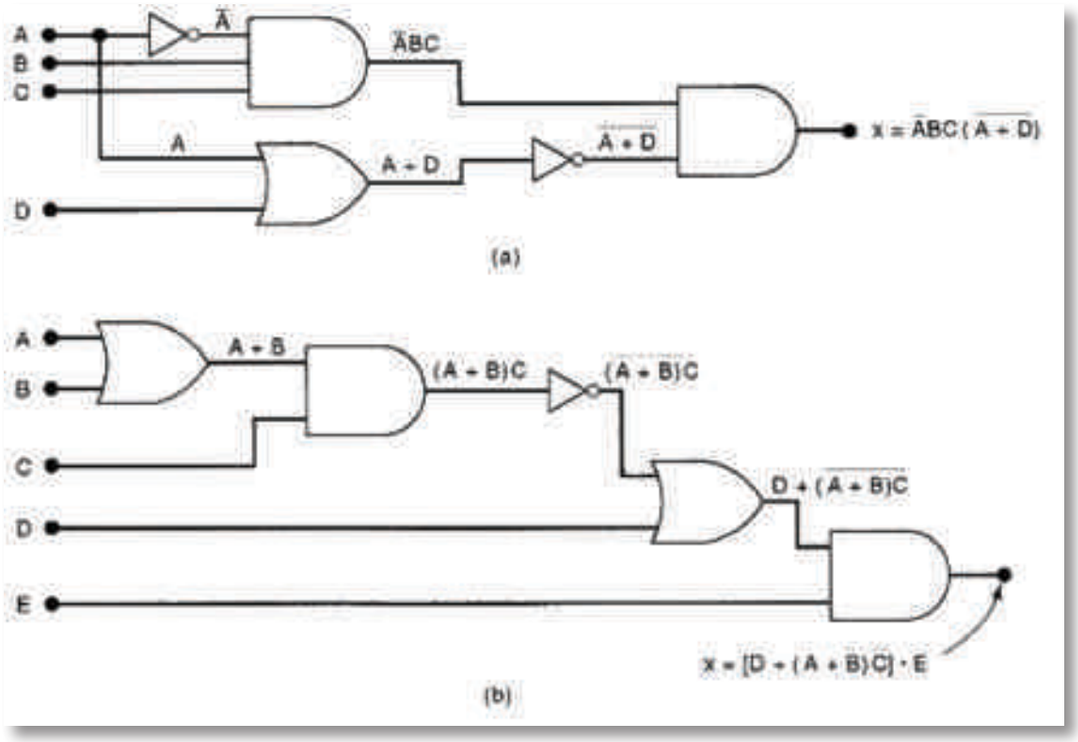


Fig. 15: Mais exemplos

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 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 pode ser encontrado substituindo-se os valores das variáveis na expressão e fazendo as operações como se segue:

$$x = \overline{A}BC\,(\overline{A+D})$$

$$x = \overline{0}.1.1.(\overline{0+1})$$

$$x = 1.1.1.(\overline{0+1})$$

$$x = 1.1.1.(\overline{1})$$

$$x = 1.1.1.0$$

$$x = 0$$

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.

$$x = [D+\overline{(A+B)}C].E$$

$$x = [1+(\overline{0+0}).\overline{1}].1$$

$$x = [1+\overline{0+1}].1$$

$$x = [1+\overline{0}].1$$

$$x = [1+1].1$$

$$x = 1.1$$

$$x = 1$$

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, O = 1 ou 1 = 0.

2. A seguir, faça todas as operações que estão dentro dos parenteses.

3. Faca a operação AND antes da operação OR, a não ser que os parenteses 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.

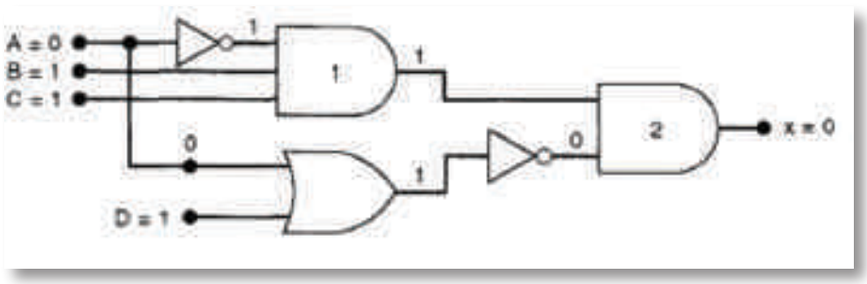


Fig. 16: Determinação do nível lógico de saída a partir do diagrama do circuito

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 e definido pela expressão X = A . B . 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 numa 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 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 vamos ver mais tarde que existem outras técnicas melhores e mais eficientes que podem ser empregues.

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

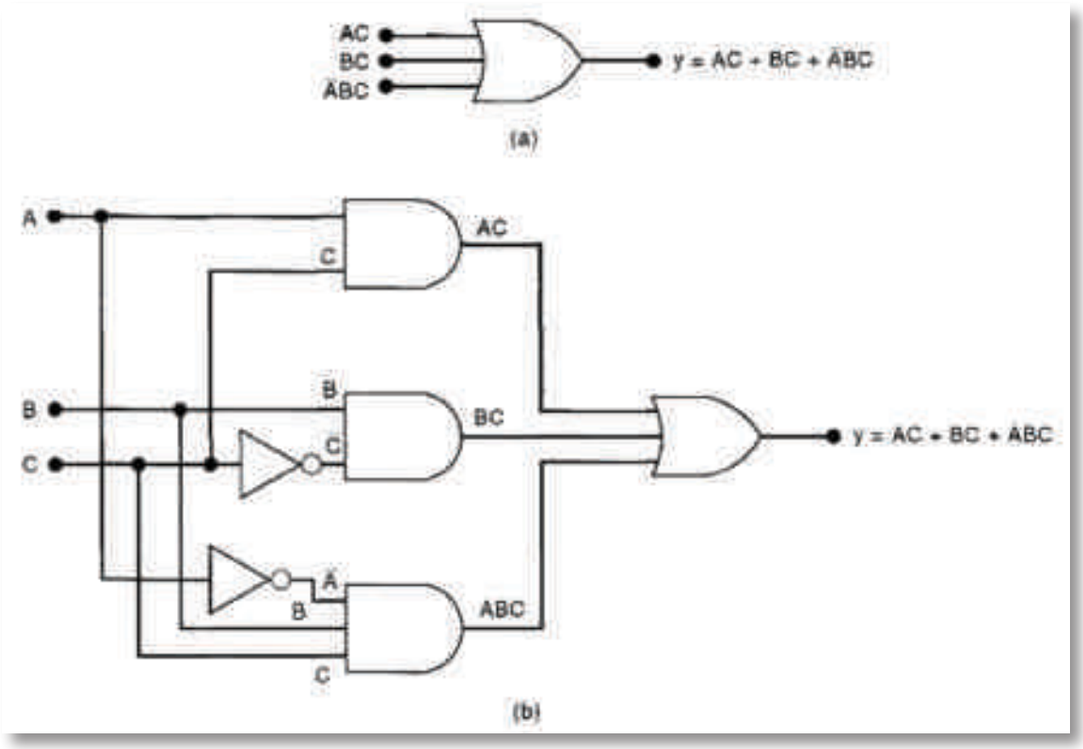


Fig. 17: Construindo um circuito lógico a partir de uma expressão booleana.

Exercício 1:

Desenhe o circuito que implementa a expressão = 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.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 do que duas entradas.

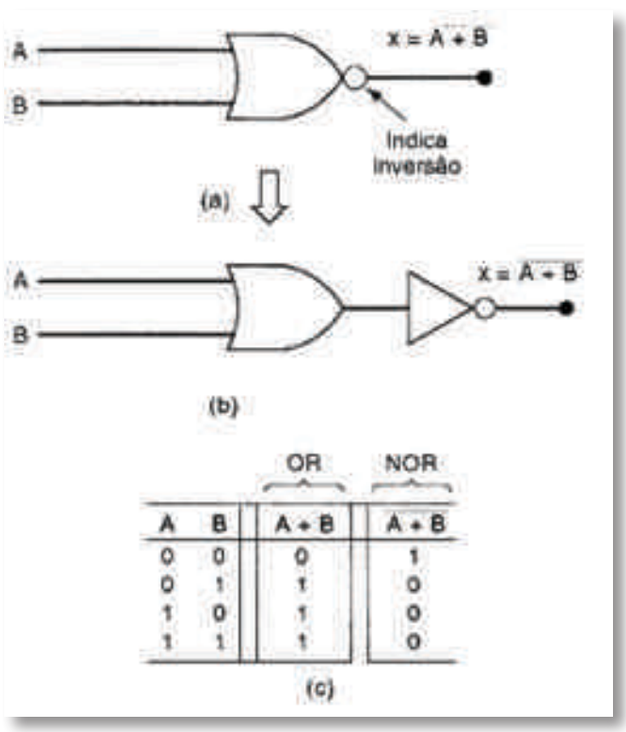


Fig. 18: (a) Símbolo para porta NOR; (b) circuito equivalente.

Exercício 1:

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

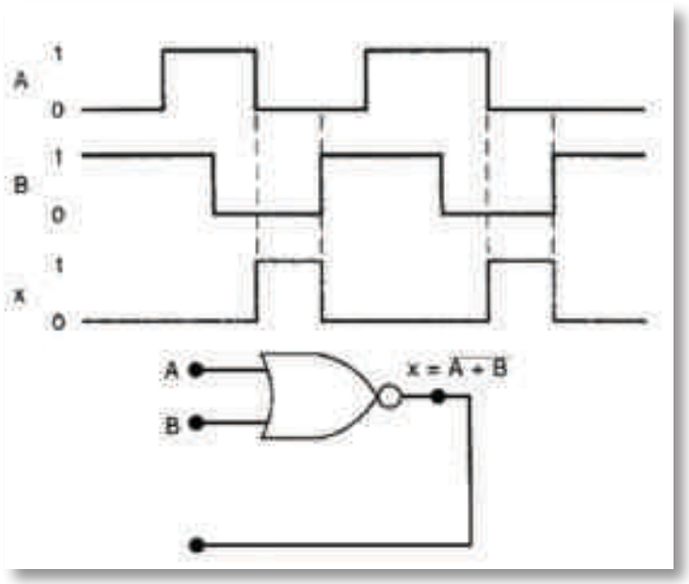


Fig. 19: Exemplo 1

Exercício 2:

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

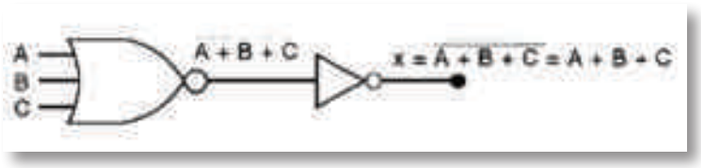


Fig. 20: Exemplo 2

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 na 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 = $\overline{AB}$.

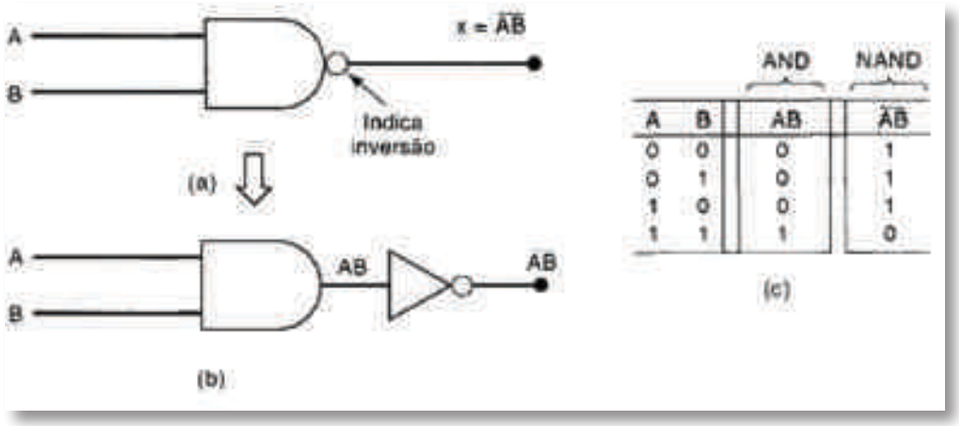


Fig. 21: (a) Símbolo para porta NAND; (b) Tabela de Verdade

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 do que 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.

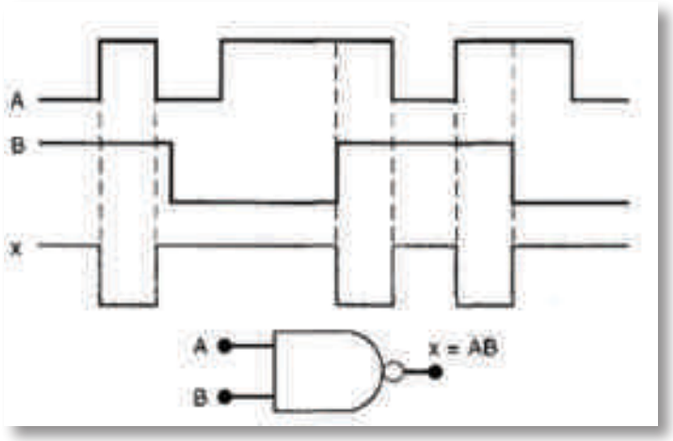


Fig. 22: Exemplo 1

Exercício 2:

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

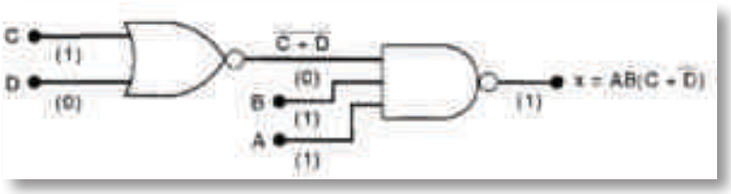


Fig. 23: Exemplo 2 e 3

Exercício 3:

Determine o nível lógico da saída na Fig.23 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.24. 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 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.

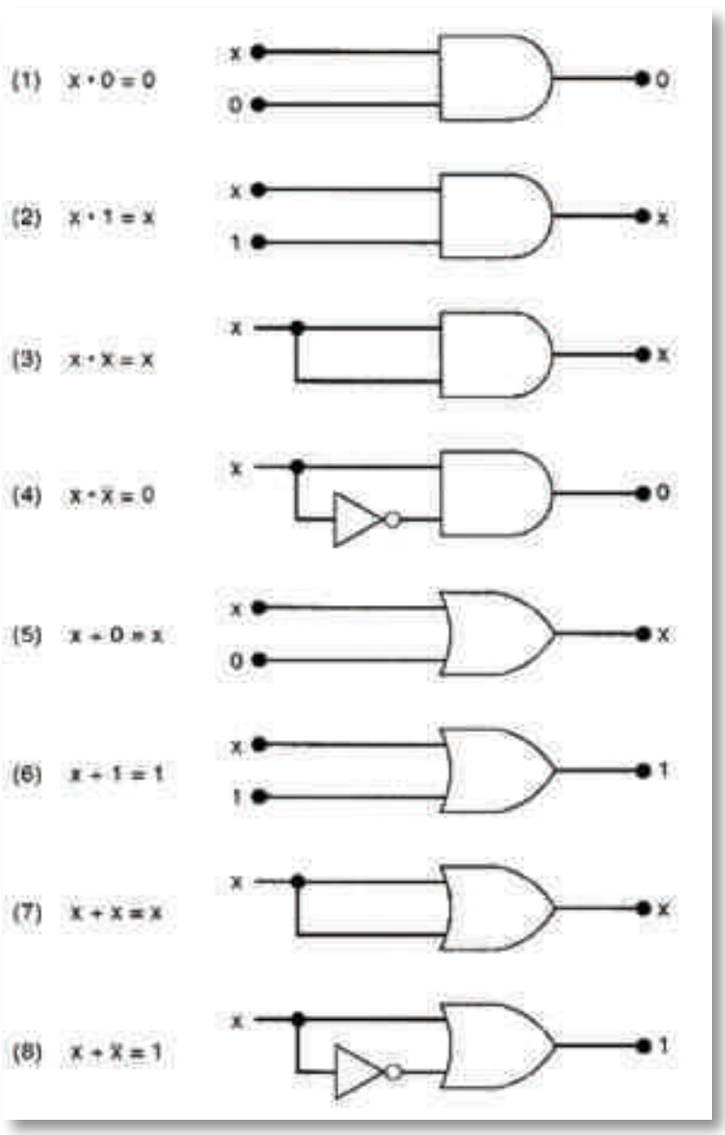


Fig. 24: Teoremas de uma variável

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

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. Entretanto, podemos raciocinar que em qualquer instante ou X ou seu inverso X deve ser igual a 0 e, então, uma operação AND de X com seu inverso sera 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, ou podemos raciocinar que em qualquer instante X ou seu inverso X estará em nível lógico 1, então teremos sempre 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 do que uma variável. Por exemplo, se tivermos a expressão A $\overline{B}$ ($A\overline{B}$), podemos aplicar o teorema (4) se fizermos X = A $\overline{B}$. Então podemos dizer que A $\overline{B}$ ($A\overline{B}$) = 0.

Este mesmo raciocínio pode ser aplicado para o uso de qualquer um destes teoremas.

Teoremas com mais do que uma Variável

Os teoremas apresentados a seguir envolvem o uso de mais do que 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, 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 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 evidencia 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 tem as variáveis A e B em comum, e portanto A e B podem ser evidenciados, como vemos a seguir.

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

Os teoremas (9) a (13) são fáceis de lembrar porque são idênticos aos 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

x + xy = x

0 + 0.0 = 0

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

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 uteis para simplificar expressões nas quais o produto (AND) ou a soma (OR) das variáveis é invertido. Os dois teoremas são:

$$(16)\ \overline{(x+y)} = \bar{x}\,.\,\bar{y}$$
$$(17)\ \overline{(x.y)} = \bar{x}+\bar{y}$$

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 será feita como exercício no final do capítulo.

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

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

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

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

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

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

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

Exercício 1:

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

O Exercício 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
$z = \overline{\bar{A} + \bar{B} \,.\, C}$ $z = \bar{\bar{A}} \,.\, (\overline{\bar{B} \,.\, C})$ $z = \bar{\bar{A}} \,.\, (\bar{\bar{B}} + \bar{C})$	$w = \overline{(\bar{A} + BC).(D + \bar{E}F)}$ $w = (\bar{\bar{A}} + \bar{BC}) + (\bar{D} + \bar{EF})$ $w = (\bar{\bar{A}} \,.\, \bar{BC}) + (\bar{D} \,.\, \bar{EF})$ $w = [\bar{\bar{A}} \,.\, (\bar{\bar{B}} + \bar{C})] + [\bar{D} \,.\, (\bar{E} + \bar{F})]$ $w = \bar{\bar{A}}\bar{B} + \bar{\bar{A}}\bar{C} + \bar{D}\bar{E} + \bar{D}\bar{F}$

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

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

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):

$$\overline{(x+y)} = \bar{x}\,.\,\bar{y}$$

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.25 (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.25 (b), onde os pequenos círculos nas entradas representam a operação de inversão.

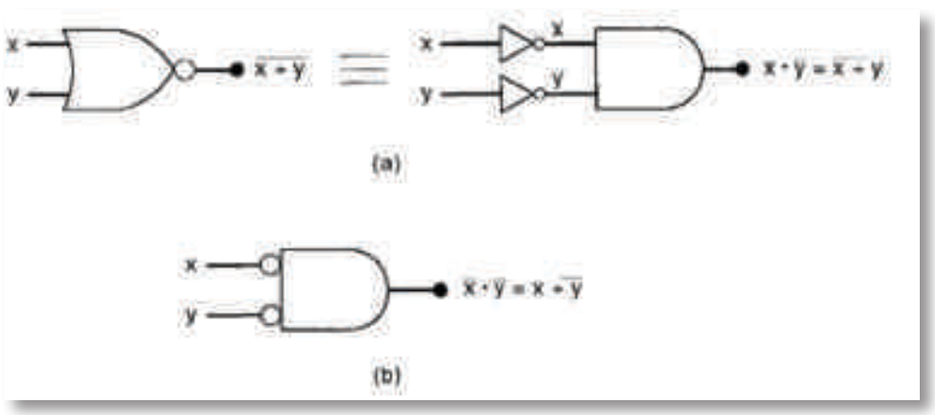


Fig. 25: (a) Circuitos equivalentes obtidos pela aplicação do teorema (16); (b) símbolo alternativo para a função NOR

Agora considere o teorema (17):

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

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.26(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.26 (b), onde os círculos mais uma vez representam inversão.

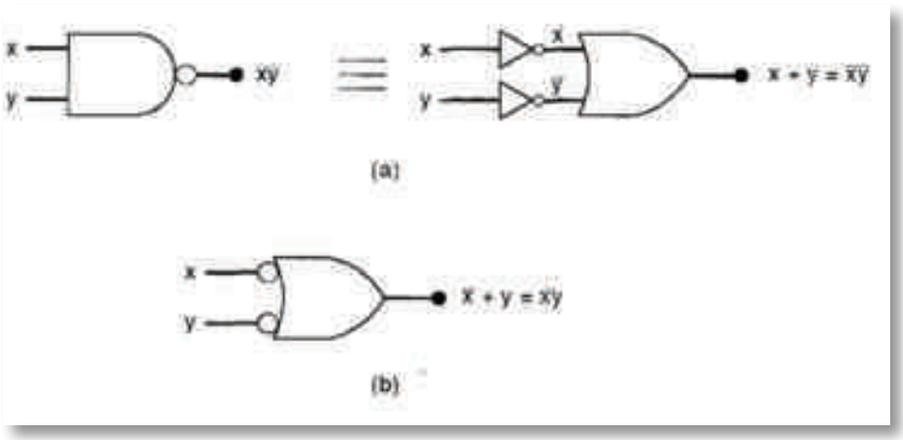


Fig. 26: (a) Circuitos equivalentes obtidos pela aplicação do teorema (17); (b) símbolo alternativo para a função NAND.

Exercício 1:

Determine a expressão lógica para a saída do circuito da Fig.27 e simplifique-a usando os teoremas de DeMorgan.

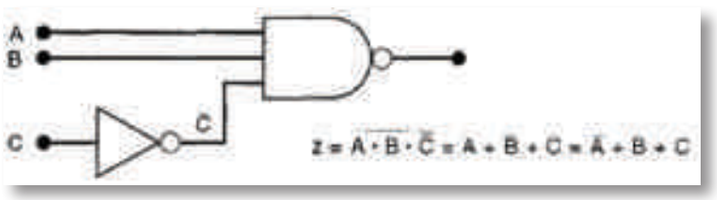


Fig. 27: Exemplo 1

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.28.

Em primeiro lugar, na Fig.28 (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. 28 (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.28 (c). Neste caso, as portas NAND número 1 e 2 são usadas como Inversoras para inverter as entradas, de modo que o resultado final seja $x = \overline{\overline{A}.\overline{B}}$, que pode ser simplificado para 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. 29. 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. 29 (b), duas portas NOR são combinadas de modo que se implemente 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.29 (c). Neste caso, as portas NOR 1 e 2 são usadas como inversores para inverter as entradas, de modo que a saída seja igual a $x = \overline{\overline{A}+\overline{B}}$. que pode ser simplificado para $= A \,.\, B$, 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.

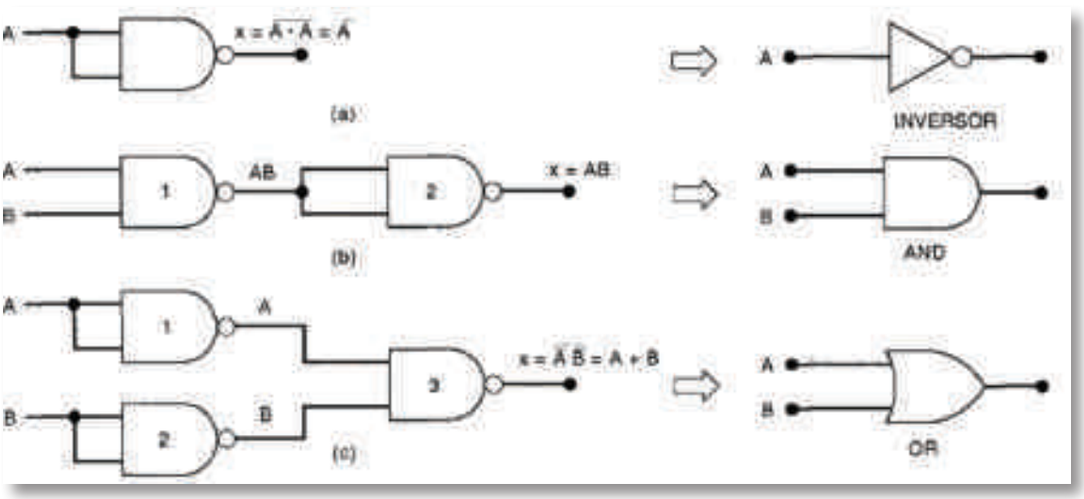


Fig. 28: Portas NAND podem ser usadas para implementar qualquer função booleana

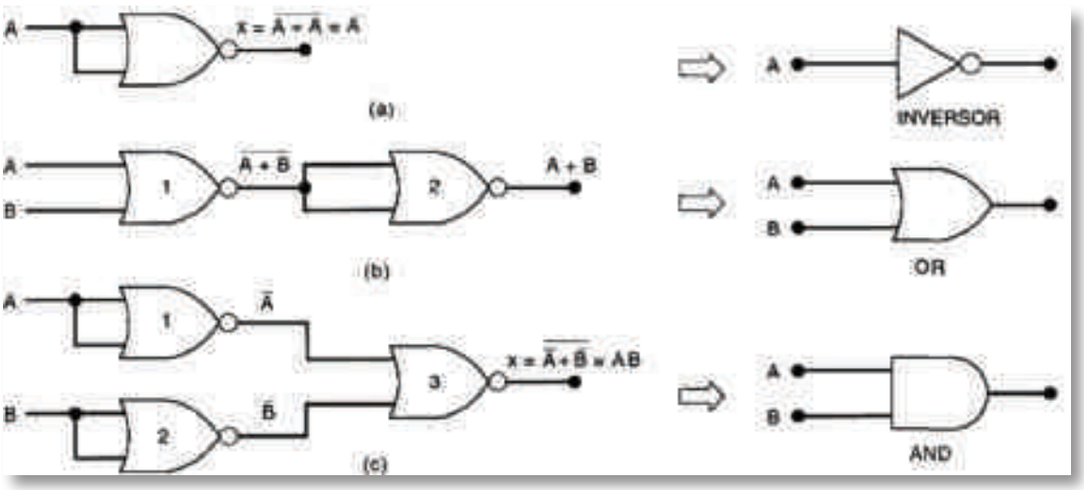


Fig. 29: Portas NOR podem ser usadas para implementar qualquer função booleana.

Exemplo 1:

Num determinado processo de fabricação, uma esteira de transporte deve ser desligada sempre que determinadas condições ocorrerem. 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 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 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.30 estão disponíveis.

Cada Cl é quadruplo, 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.31 (a). Esta implementação utiliza duas portas do Cl 74LS08 e uma única porta do Cl 74LS32. Os números entre parenteses, 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.31 (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.31 (b).

À primeira vista, esse novo circuito parece precisar de sete portas NAND. Entretanto, 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.31 (c).

Esse circuito é mais eficiente do que o da Fig. 31 (a) porque utiliza três portas NAND de duas entradas que podem ser implementadas por um Cl, o 74LS00.

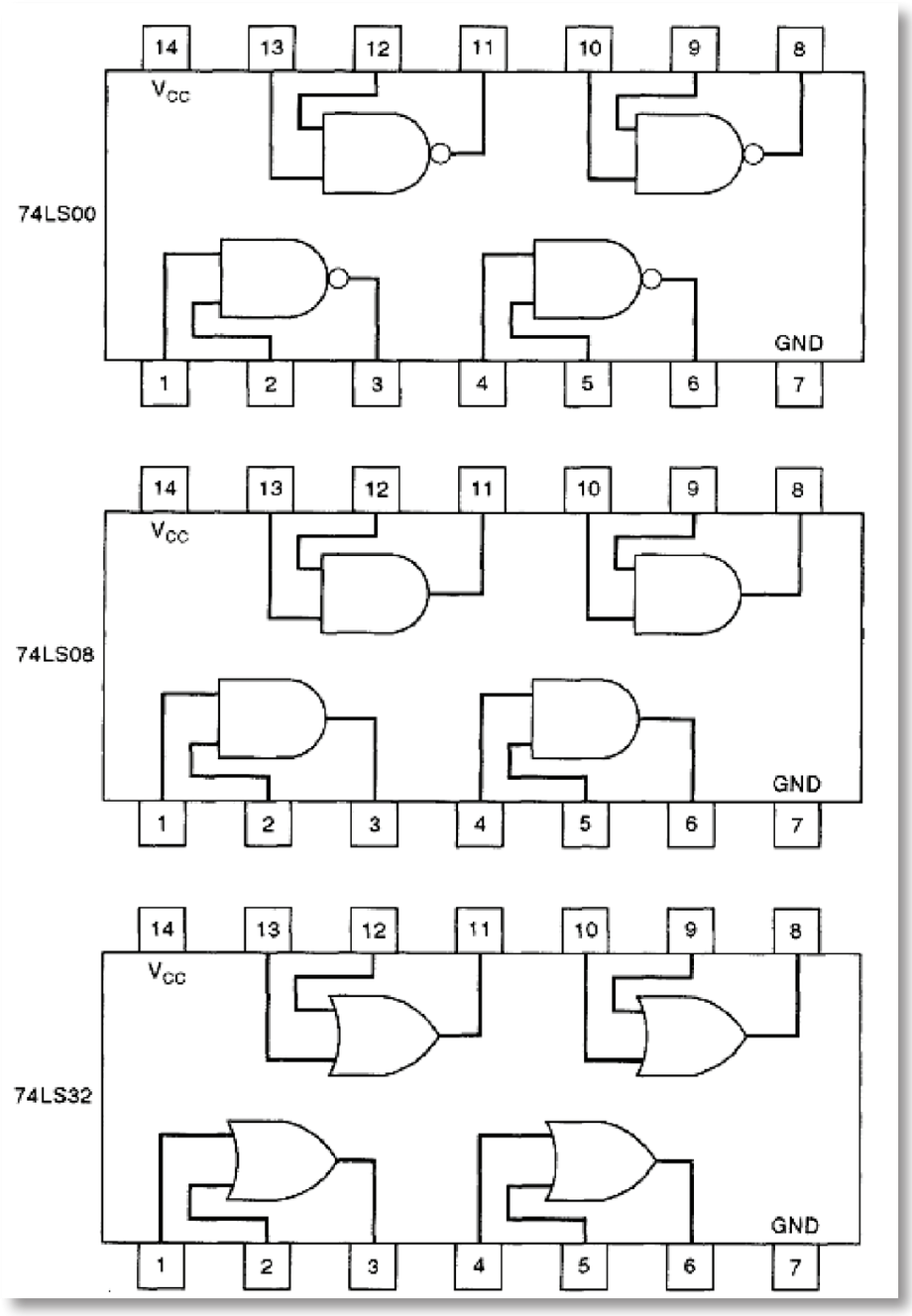


Fig. 30: Exemplo 1

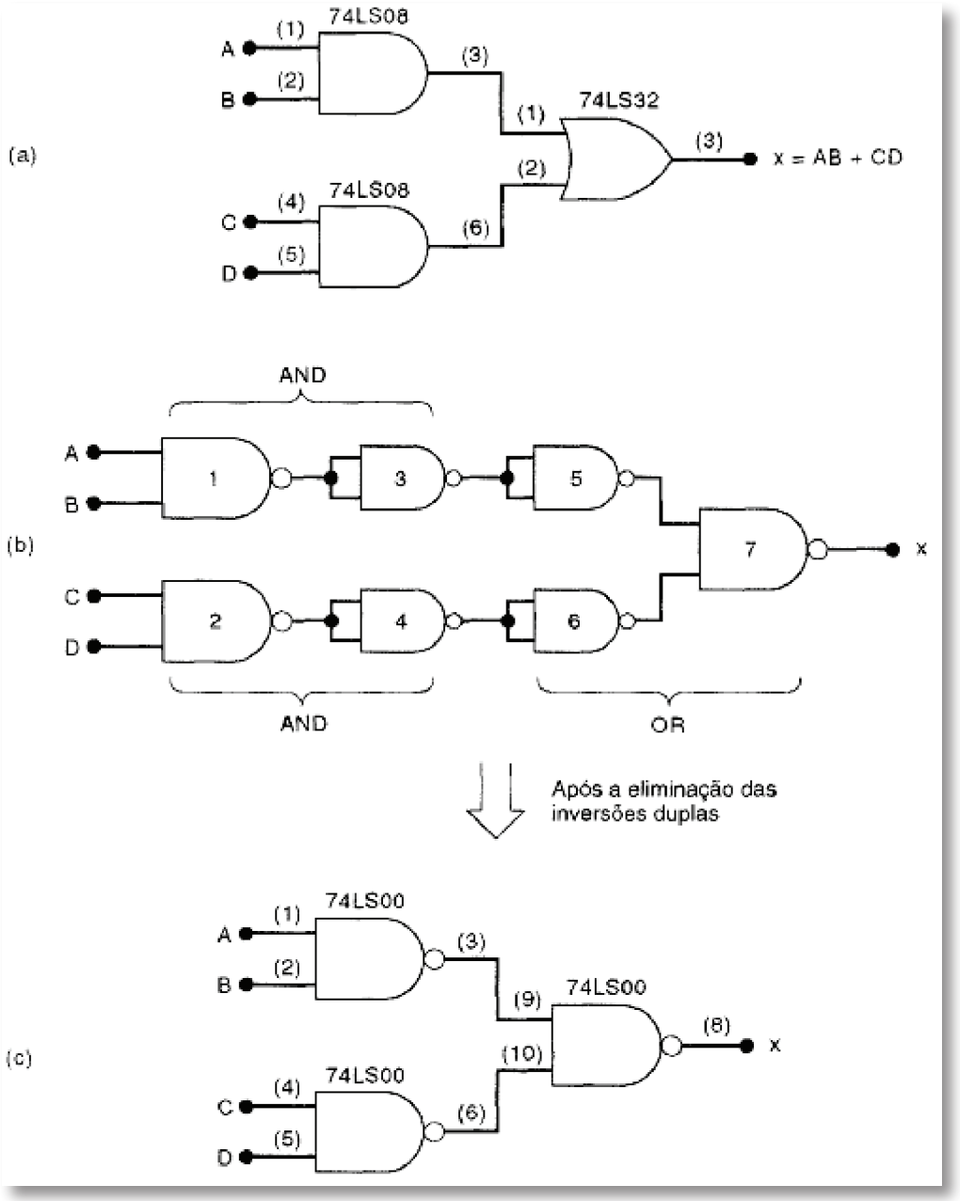


Fig. 31: Implementações possíveis para o exemplo 1

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. 32 (a). A expressão de saída deste circuito é:

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

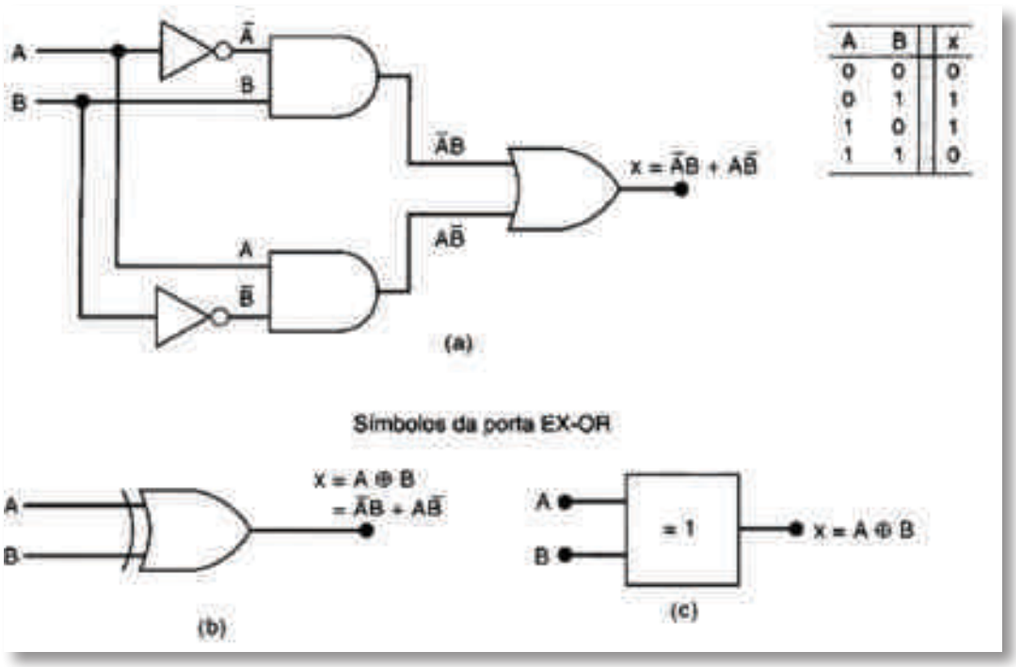


Fig. 32: (a) Tabela DE verdade e circuito exclusive-OR; (b) símbolo tradicional da porta EX-OR; (c) símbolo IEEE/ANSI para a porta EX-OR.

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.33 (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 de 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. 33 (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: $x = \bar{A}B + A\bar{B} = A \oplus B$

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)

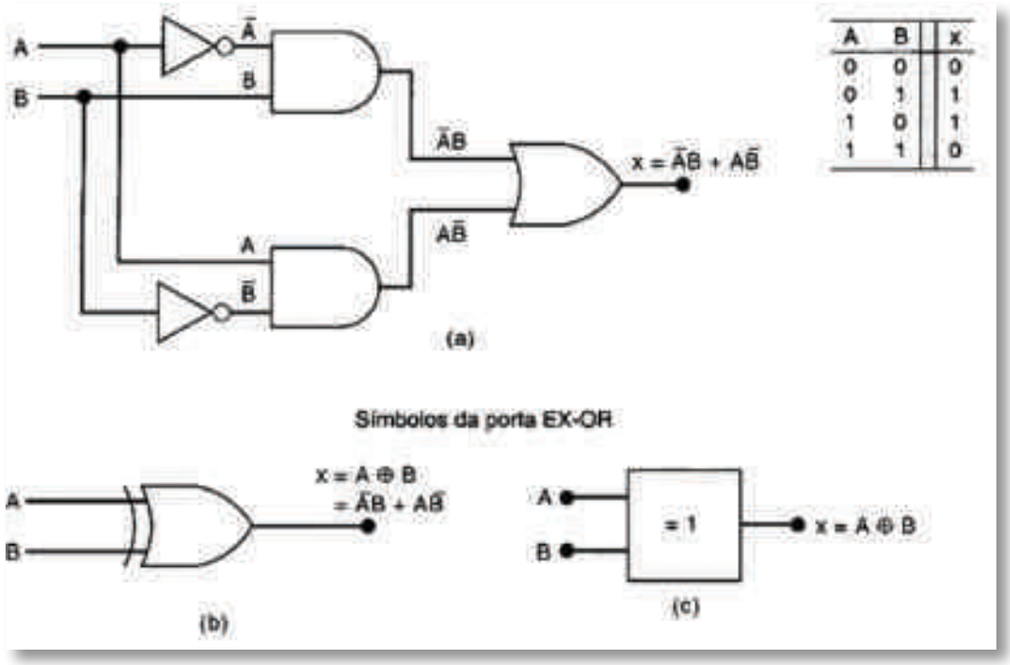


Fig. 33: (a) Tabela de verdade e circuito exclusive-OR; (b) símbolo tradicional da porta XOR; (c) símbolo IEEE/ANSI para a porta XOR.

X NOR

O circuito exclusive - NOR (abreviado como XNOR) opera ao contrário do circuito XOR. A Fig. 34(a) mostra um circuito XNOR e sua respetiva tabela de verdade. A expressão de saída é:

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

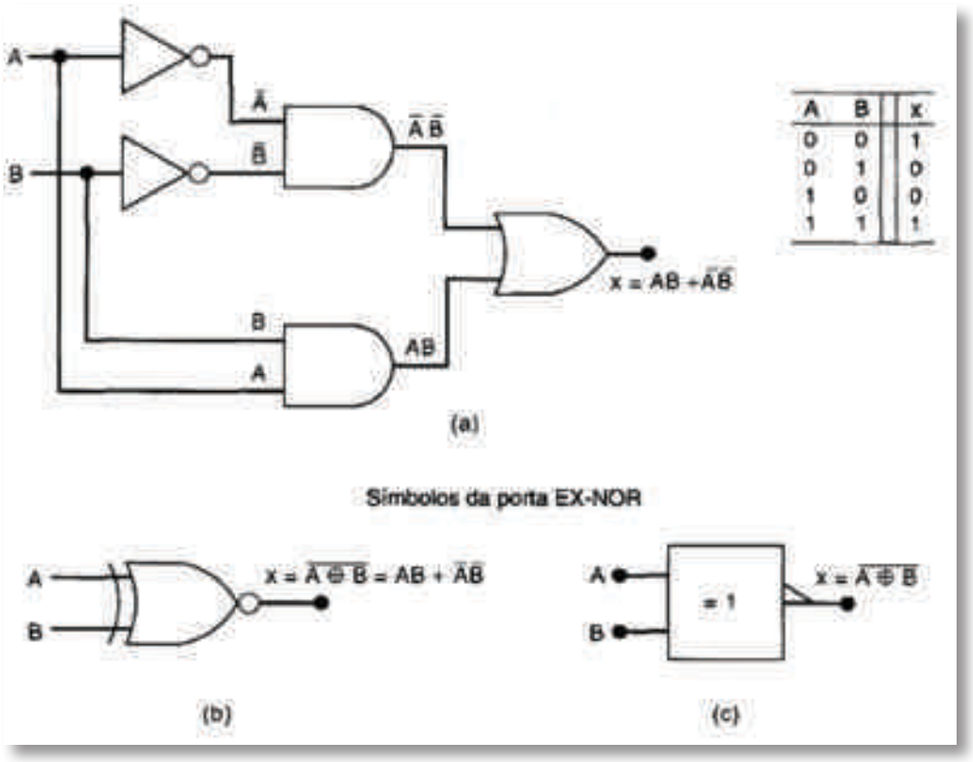
que indica juntamente com a tabela de verdade que é 1 para dois casos: A = B = 1 (o termo AB) e A = B = 0 (o termo $\bar{A}\bar{B}$). Em outras palavras:

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 é exatamente o inverso da saída de um circuito XOR. O símbolo tradicional de uma porta XNOR é obtido simplesmente adicionando-se um pequeno circulo à saída do símbolo do XOR Fig.34 (b). O símbolo adiciona um pequeno triângulo à 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:

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



A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Fig. 34: (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 a sua saída é

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

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. 35.

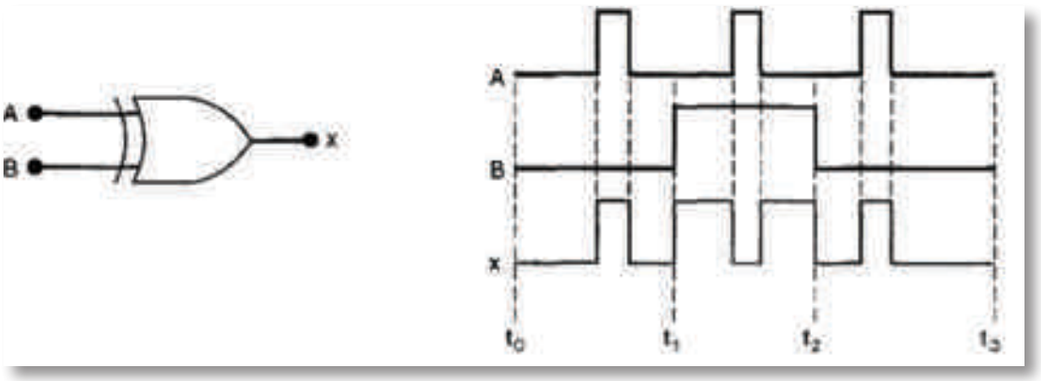


Fig. 35: Exemplo 1

Exercício 2:

$x_1 x_0$, 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, $y_1 y_0$ representa um outro número binário de dois bits. Projete um circuito lógico, usando as entradas $x_1 x_0$ e $y_1 y_0$, cuja saída vai para ALTO somente quando os dois números binários $x_1 x_0$ e $y_1 y_0$ são iguais.

Exercício 3:

Quando se simplifica a expressão para a saída de um circuito lógico convencional, pode-se encontrar operações XOR ou XNOR durante a evidenciação. Isto frequentemente conduz ao uso de portas XOR ou XNOR na implementação do circuito final. Para ilustrar, simplifique o circuito da Fig. seguinte.

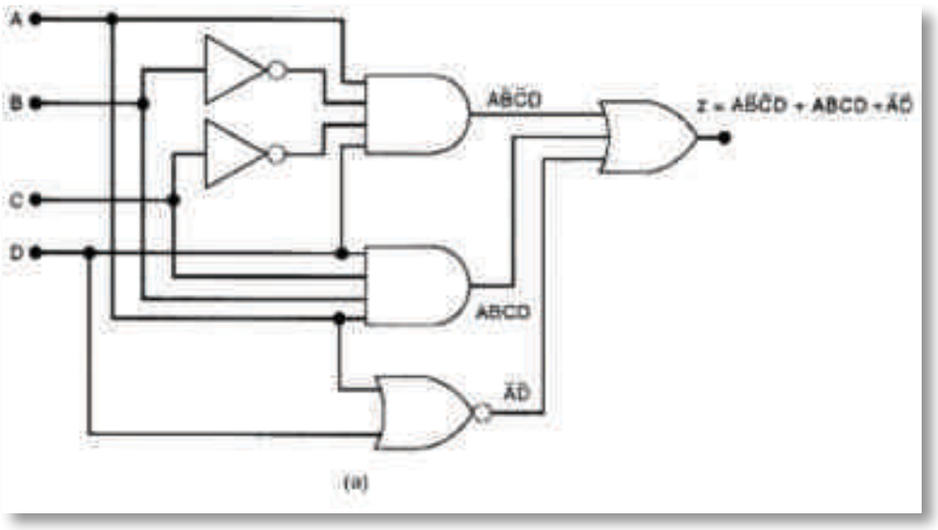


Fig. 36: O exercício 3 mostra como uma porta XNOR pode ser usada para simplificar a implementação de circuitos.

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 envolvam 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 demasiadamente 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.37 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:

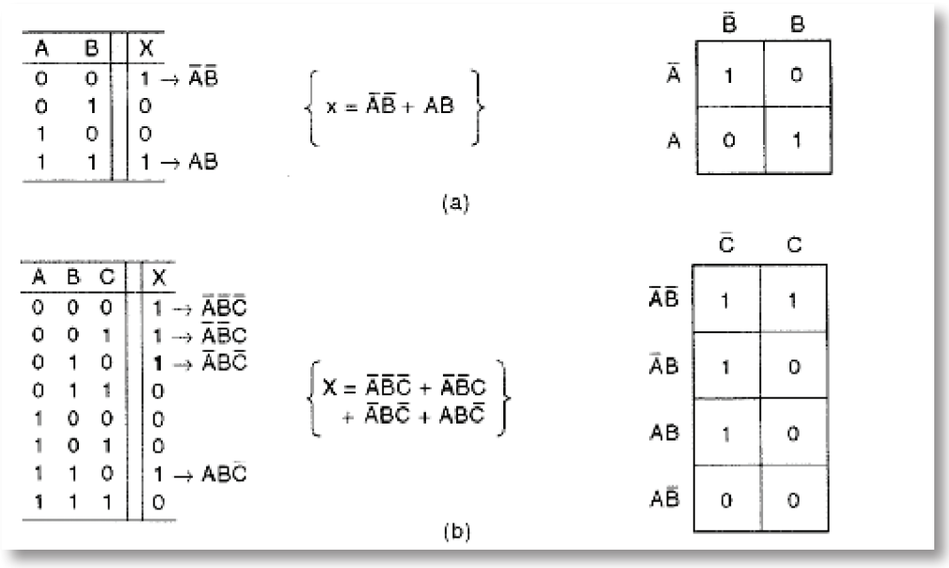
1. A tabela de verdade fornece o valor da saída X para cada combinação de valores da entrada. O mapa K fornece a mesma informação num formato diferente. Cada linha na tabela de verdade corresponde a um quadrado no mapa K. Por exemplo, na Fig. 37 (a), a condição A = 0, B = 0, na tabela de verdade, corresponde ao quadrado $\overline{A}\overline{B}$ 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 difiram 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 à direita.

3. Para que os quadrados adjacentes, tanto na horizontal quanto na vertical, difiram em apenas numa 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 à 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 os quadrados que contêm 1. No mapa de três variáveis da Fig. 37(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 = $\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC$.



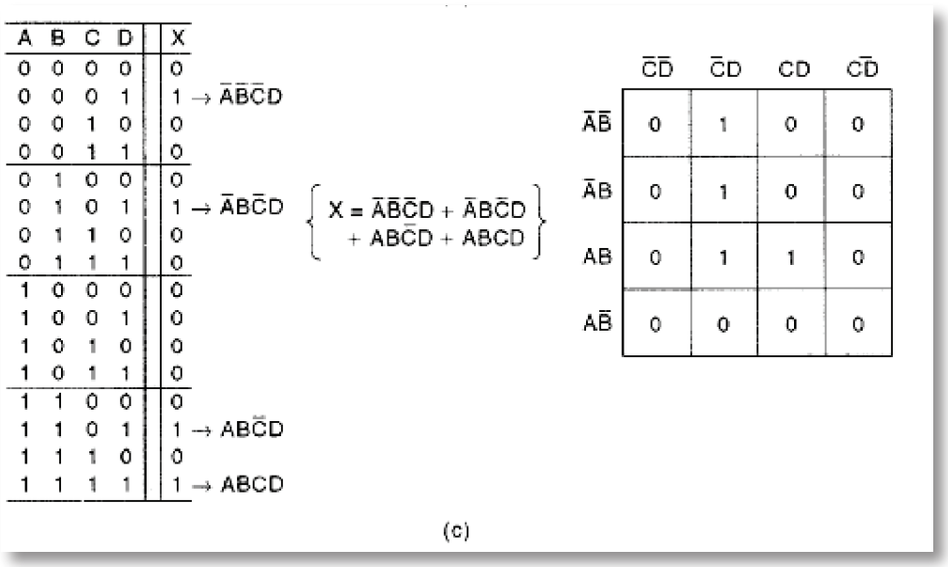


Fig. 37:Mapas de Karnaugh e tabelas de verdade para (a) duas, (b) três e (c) quatro variáveis.

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. 38 (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 $\bar{A}B\bar{C}$ e o segundo representa $AB\bar{C}$. Repare que nestes dois termos apenas a variável *A* aparece tanto na forma normal quanto na complementar (*B* e $\bar{C}$, 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:

$$x = \bar{A}B\bar{C} + AB\bar{C}$$
$$x = B\bar{C}(\bar{A} + \bar{C})$$
$$x = B\bar{C}(1) = B\bar{C}$$

Este mesmo princípio permanece válido para qualquer par de 1s adjacentes na vertical ou na horizontal. A Fig. 38 (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 $X = \overline{A}B$.

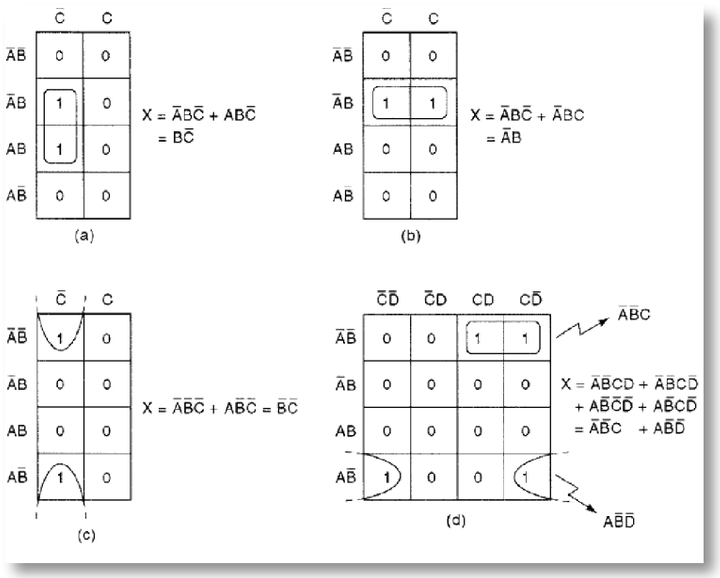


Fig. 38: Exemplos de agrupamentos de pares de 1s adjacentes.

Um outro exemplo está ilustrado na Fig. 38 (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 $\overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} = \overline{B}\overline{C}$.

A Fig. 38 (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, em um 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. Resumindo: 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.39 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. 39 (c) contém quatro 1s formando um quadrado, e eles são considerados adjacentes entre si. Os quatro 1s na Fig. 39 (d) também são adjacentes, assim como os da Fig. 39 (e) porque, conforme apresentado anteriormente, as linhas superior e inferior são consideradas adjacentes entre si, do mesmo modo que as colunas mais à esquerda e mais à 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. 39 (a), os quatro quadrados que contêm 1s são $\bar{A}\bar{B}C$, $\bar{A}BC$, ABC e $A\bar{B}C$. 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:

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

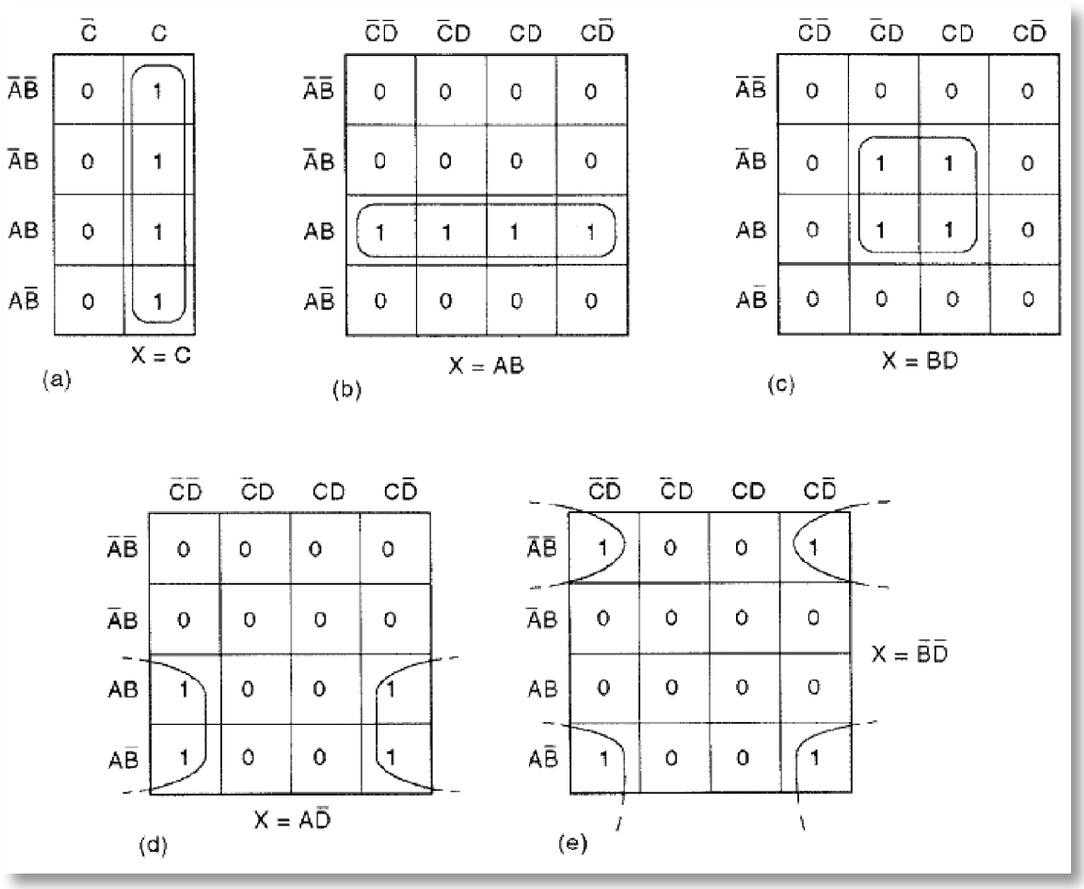


Fig. 39: Exemplos de agrupamentos de quatro 1s (quartetos).

Outro exemplo, considere a Fig. 39 (d), onde os quatro quadrados que contém 1s são: $AB\overline{C}\overline{D}$, $A\overline{B}\overline{C}\overline{D}$, $ABC\overline{D}$, e $A\overline{B}C\overline{D}$. Um exame destes termos indica que somente as variáveis *A* e *D* permanecem inalteradas, portanto a expressão simplificada para *X* é:

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

Isto pode ser provado da mesma maneira que foi feito anteriormente. Deve analisar-se cada um dos casos na Fig.39 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)

Um grupo de oito 1s que são adjacentes entre si é chamado de octeto. Muitos exemplos de octetos são mostrados na Fig.40. 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. 40 (a) mostra que somente a variável *B* está na mesma forma para todos os oito quadrados; as outras variáveis aparecem nas formas complementada e não-complementada. Portanto, para este mapa, X=*B*. Podem verificar-se os resultados para os outros exemplos na Fig. 40.

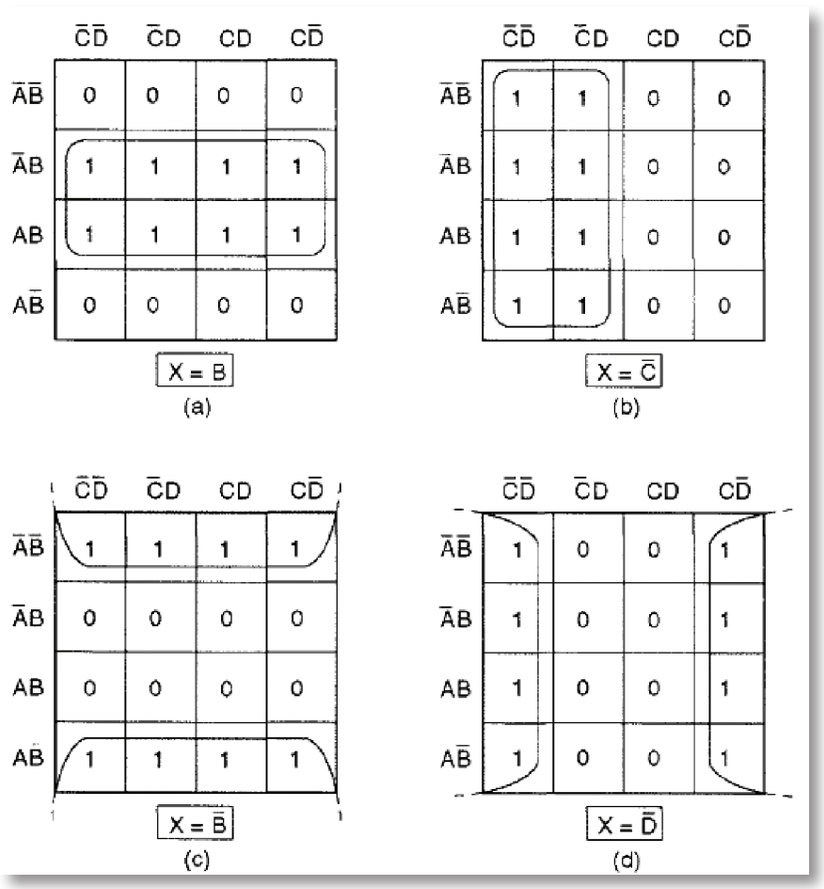


Fig. 40: Exemplos de agrupamentos de oito 1s (octetos).

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. Para ser exato, 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 contêm 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. 41 (a) mostra o mapa K para um problema de quatro variáveis. Vamos supor que o mapa foi obtido a partir da tabela de 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 $\overline{A}\overline{B}C\overline{D}$. O grupo 11, 15 e *ACD* (a variável *B* foi eliminada). O grupo 6, 7, 10, 11 e BD (A e C foram eliminadas).

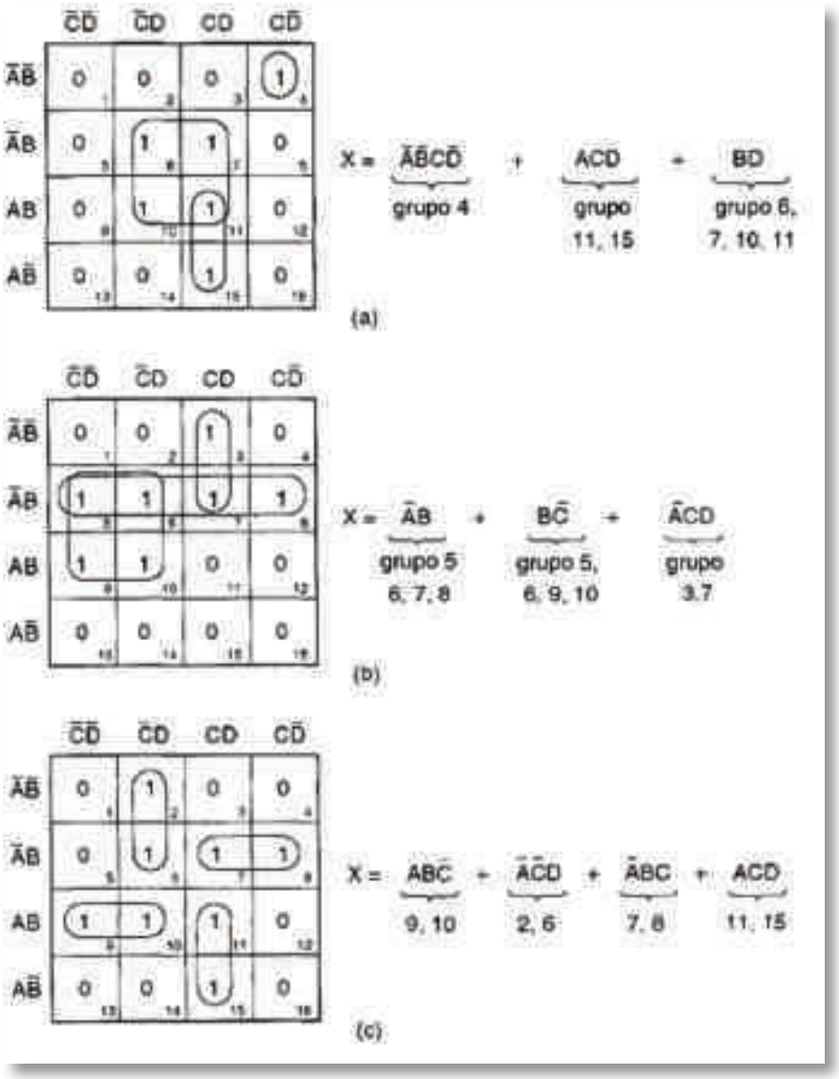


Fig. 41: Exemplo 1, 2 e 3

Exemplo 2:

Considere o mapa K na Fig.41 (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 contém dois quadrados que não tinham sido combinados anteriormente. Este grupo produz $B\overline{C}$.

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.41 (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. 42 (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. 42 (a) mostra uma solução e a expressão resultante. A Fig. 42 (b) mostra a outra. Note que, ambas as expressões têm a mesma complexidade, e portanto nenhuma é melhor do que a outra.

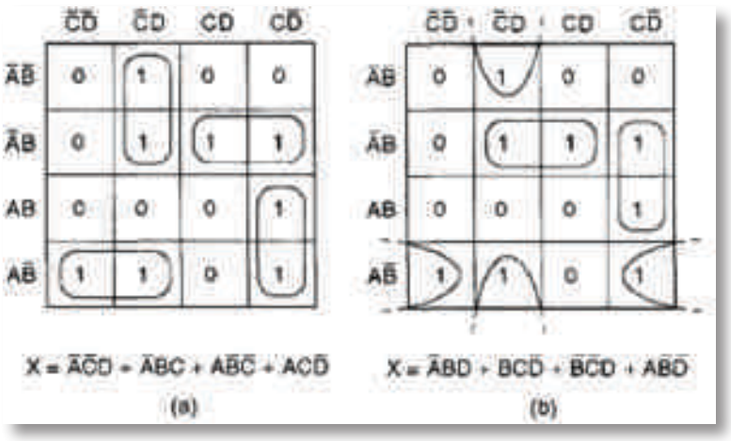


Fig. 42: O mesmo mapa K com duas soluções igualmente boas.

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.43). O segundo termo, $\bar{B}C$, indica que um 1 deve ser colocado em cada quadrado que contém $\bar{B}C$ no seu rotulo. Na Fig. 43, 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.

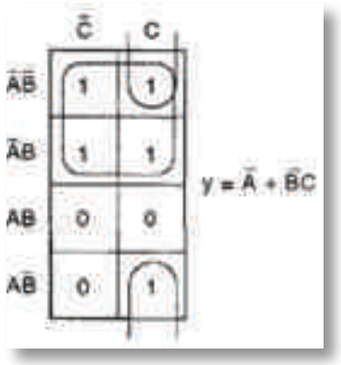


Fig. 43: Exemplo 5

Exercício 1:

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

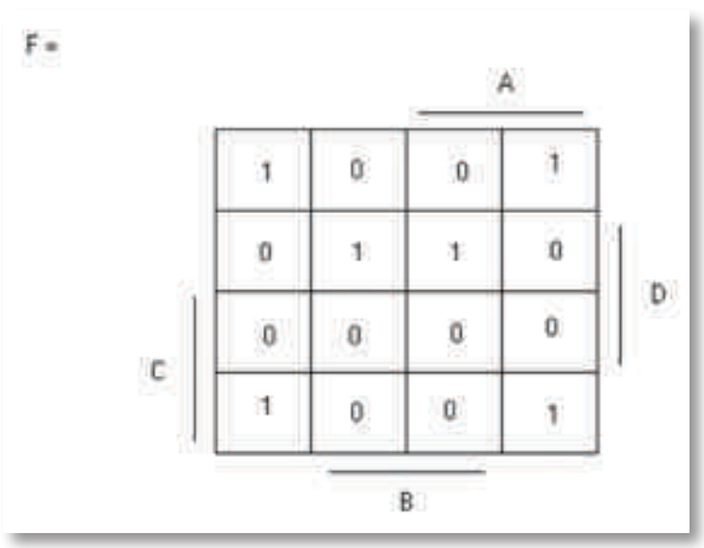
a) $F = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,C$

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

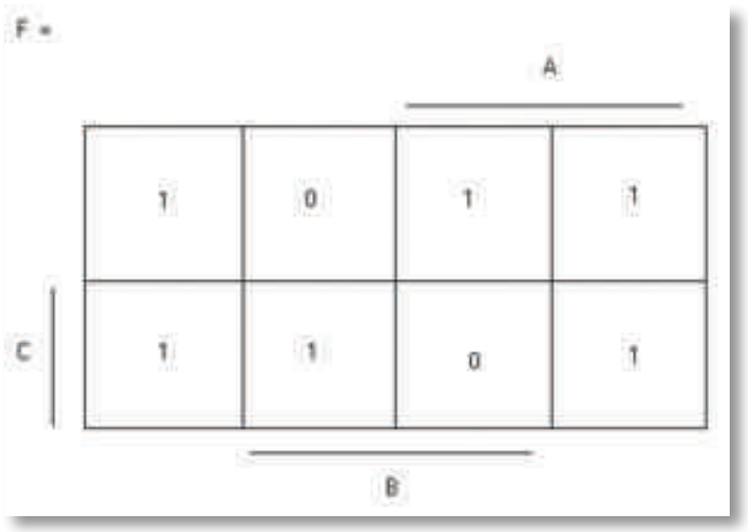
Exercício 2:

1. Extraia as expressões simplificadas dos seguintes mapas de Karnaugh:

a)



b)



Bibliografia

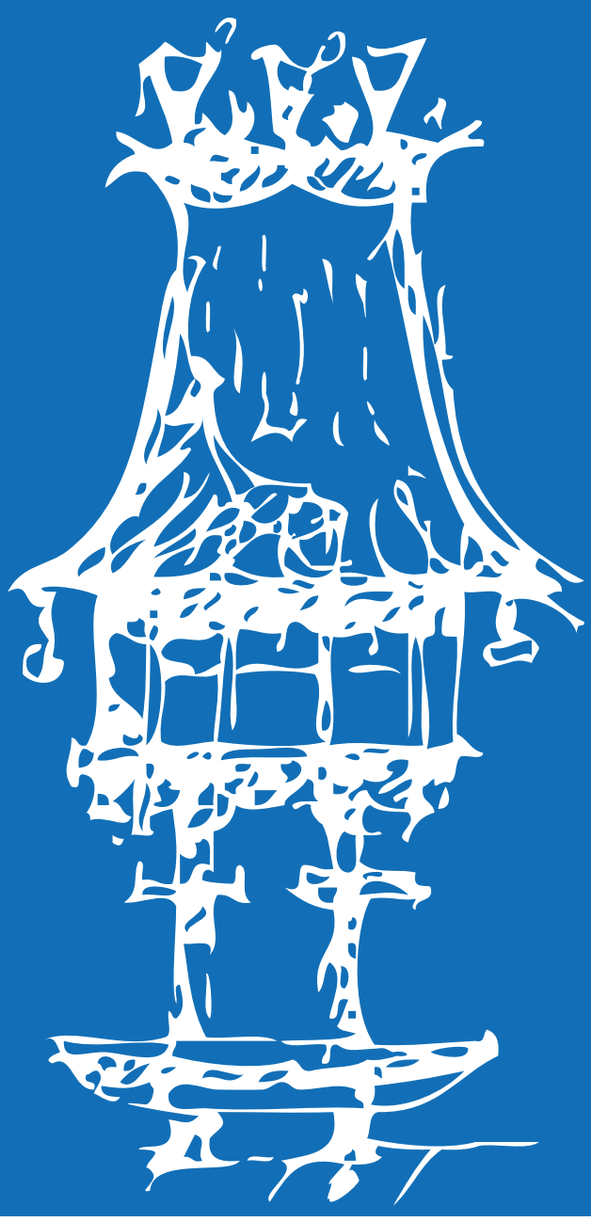
PADILHA, António e outros, *Electrónica Digital*. McGrawHill. (s.d.).

PADILHA, António, *Sistemas Digitais*. McGrawHill. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Analógicos e Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

Notas



Circuitos Combinatórios

Módulo 3

Apresentação

Este módulo tem carácter teórico-prático, devendo decorrer essencialmente em ambiente laboratorial de modo que os alunos possam ensaiar e comprovar as características e funcionamento dos circuitos combinatórios estudados na teoria.

Introdução

A abordagem deste módulo de Circuitos Combinatórios leva-nos a uma melhor compreensão do funcionamento de vários tipos de aparelhos, que incorporam circuitos que útilizam estas características, existentes no mercado assim como a melhor escolha deste tipo de equipamentos para que se ajuste às crescentes evoluções disponíveis pelas diversas marcas.

Este módulo requer um conhecimento básico de matemática e análise de circuitos eletrónicos assim como a respetiva compreensão desses circuitos.

Objetivos de aprendizagem

- Em relação aos circuitos codificadores / descodificadores, multiplexers / desmultiplexers, comparadores e somadores / subtratores os alunos devem:
    - Conhecer o seu funcionamento e aplicações.
    - Obter a tabela de verdade.
    - Implementar os respetivos circuitos com portas elementares ou CI.

Âmbito de conteúdos

- Codificadores e descodificadores.
- Multiplexers e desmultiplexers.
- Circuitos comparadores.
- Somadores e subtratores.

Descodificadores

Um descodificador é um circuito lógico que aceita um conjunto de entradas que representa um número binário e ativa somente uma saída, que corresponde ao número da entrada.

Por outras palavras, um circuito descodificador analisa as suas entradas, determina qual o número binário que está presente e ativa a saída correspondente a esse número; todas as outras saídas permanecem desativadas. O diagrama para um descodificador geral com *N* entradas e *M* saídas é apresentado na Fig.1. Como cada uma das N entradas pode ser 0 ou 1, existem 2^N combinações ou códigos de entrada possíveis. Para cada uma destas combinações de entrada, apenas uma das *M* saídas estará ativa (ALTO), todas as outras saídas estarão em BAIXO. Muitos descodificadores são projetados para produzir saídas ativas em BAIXO, onde apenas a saída selecionada fica em BAIXO e todas as outras permanecem em ALTO. Isto será indicado pela presença de pequenos círculos nas linhas de saída no diagrama do descodificador.

Alguns descodificadores não útilizam todos os 2^N códigos de entrada possíveis, mas apenas alguns. Por exemplo, um descodificador BCD para decimal tem um código de entrada de quatro bits e dez linhas de saída, que correspondem aos dez códigos BCD de 0000 até 1001. Descodificadores deste tipo frequentemente são projetados, de modo que caso um código não usado seja aplicado na entrada, nenhuma saída será ativada.

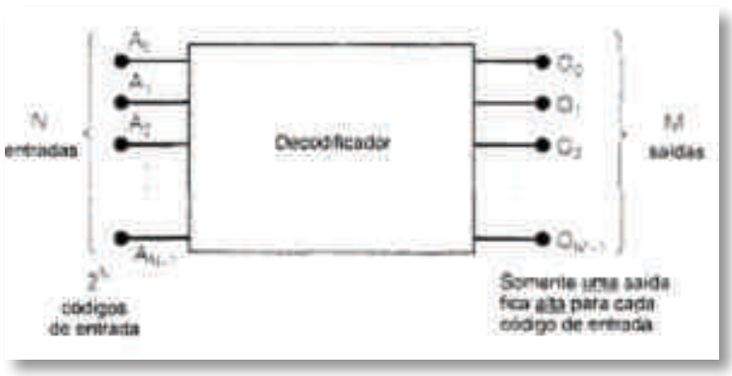
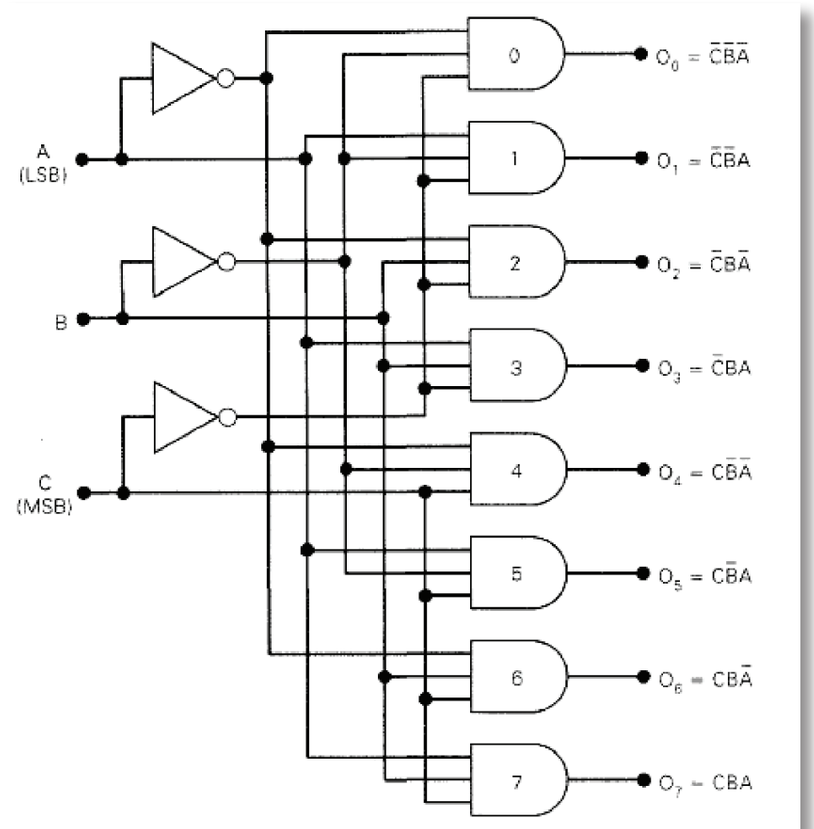


Fig. 1: Diagrama de um descodificador genérico

A Fig.2 mostra o circuito para um descodificador de três entradas e 2^3 = 8 saídas. Útiliza somente portas AND, e portanto as saídas são ativas em ALTO. Note que, para

um determinado código de entrada, a única saída que está ativa (ALTO) é aquela que corresponde ao equivalente decimal do código binário de entrada (isto é, a saída O_6 vai para ALTO somente quando *CBA* = $110_2 = 6_{10}$).

Este descodificador pode ser identificado de várias maneiras. Pode ser chamado descodificador de 3 linhas para 8 linhas, porque tem três linhas de entrada e oito linhas de saída. Também pode ser chamado de descodificador ou conversor binário para octal, porque recebe um código binário de entrada de três bits e ativa uma entre as oito (octal) saídas correspondente para aquele código. Ele também é denominado de descodificador 1 de 8, porque somente 1 das 8 saídas é ativada de cada vez.



C	B	A	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Fig. 2: Descodificador de 3 linhas para 8 linhas (ou de 3 para 8)

Habilitação de entradas

Alguns descodificadores têm uma ou mais entradas de HABILITAÇÃO (ENABLE), que são usadas para controlar a operação do descodificador. Por exemplo, imagine que o descodificador da Fig.2 tem uma linha de habilitação comum ligada numa quarta entrada de cada porta. Com esta linha de habilitação mantida em ALTO, o descodificador funciona normalmente e o código de entrada, *A, B* e *C,* determina qual das saídas fica em ALTO. Entretanto, com a habilitação mantida em BAIXO, todas as saídas são forçadas para o estado BAIXO, não importando os níveis nas entradas *A, B* e *C.* Assim, o descodificador é habilitado somente quando habilitação está em ALTO.

A Fig.3 (a) mostra o diagrama lógico para o descodificador 74LS138, conforme ele aparece no *Manual TTL (TTL Data Book)* da *Fairchild*. Ao examinar este diagrama cuidadosamente, podemos determinar exatamente como este descodificador funciona. Primeiramente, note que ele tem saídas com portas NAND, de modo que suas saídas são ativas em BAIXO. Outra indicação é a identificação das saídas como $\overline{O}_7$, $\overline{O}_6$, $\overline{O}_5$ e assim por diante; a barra indica saídas ativas em BAIXO.

O código de entrada é aplicado em A_2, A_1 e A_0, onde A_2 é o MSB. Com três entradas e oito saídas, este é um descodificador 3 para 8 ou, de modo equivalente, um descodificador 1 de 8.

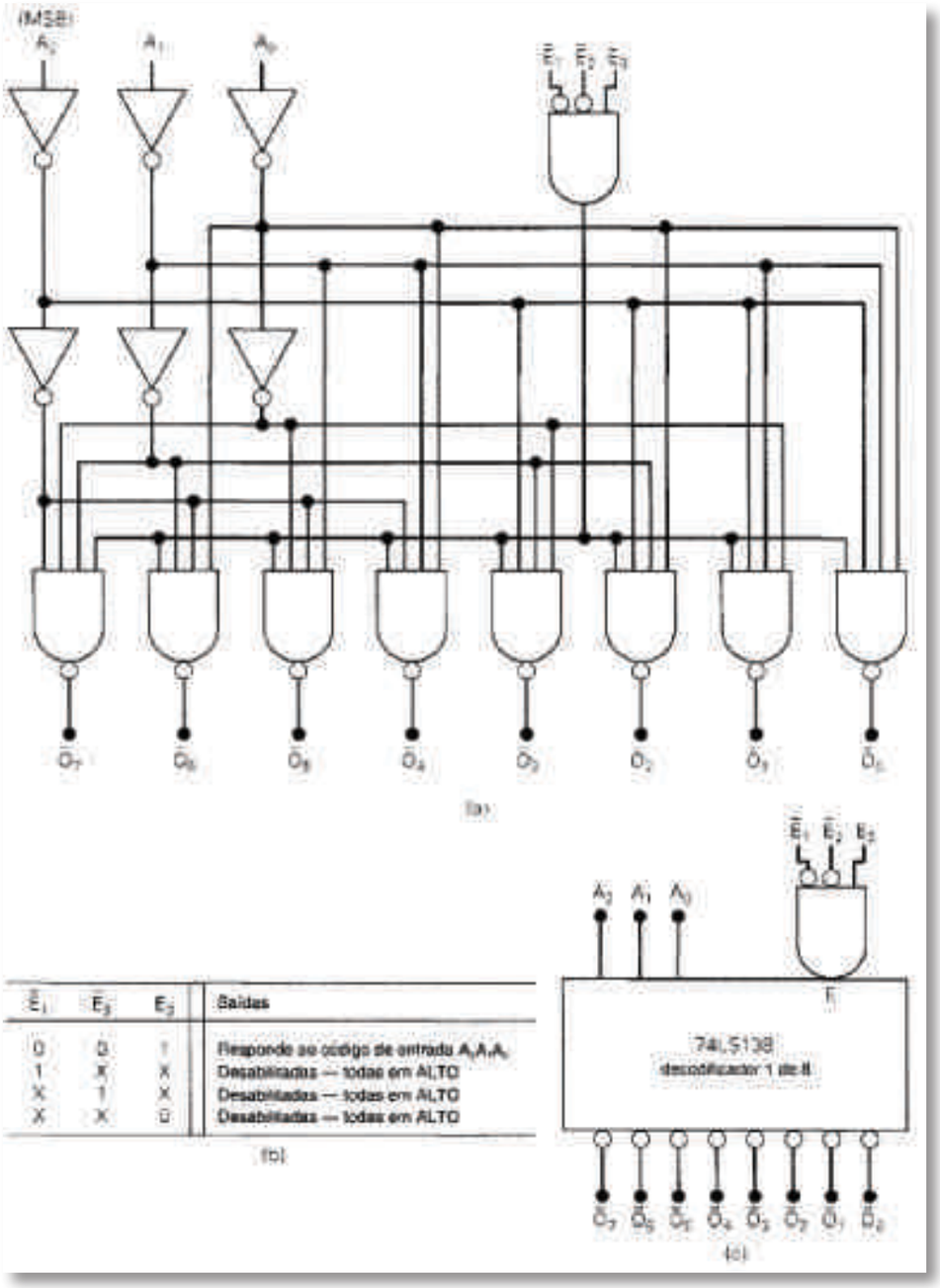


Fig. 3: (a) Diagrama lógico para o descodificador 74LS138; (b) tabela de verdade; (c) símbolo lógico.

As entradas $\overline{E}_1$, $\overline{E}_2$ e E_3 são entradas de habilitação separadas que são combinadas na porta AND. Para habilitar as portas NAND de saída para responder ao código de entrada em A_2, A_1 e A_0, a saída desta porta AND deve estar em ALTO. Isto só ocorre quando $\overline{E}_1 = \overline{E}_2 = 0$ e $E_3 = 1$. Em outras palavras, $\overline{E}_1$ e $\overline{E}_2$ são ativos em BAIXO, E_3 é ativo em ALTO, e todos os três devem estar ativos para habilitar as saídas do descodificador. Se uma ou mais

entradas de habilitação estão no seu estado inativo, a saída da porta AND fica em BAIXO, o que força todas as saídas NAND para os seus respetivos estados inativos em ALTO, não importando o código de entrada. Esta operação está resumida na tabela verdade da Fig.3 (b). De lembrar que X representa a condição *don't care*.

O símbolo lógico para o 74LS138 é apresentado na Fig.3 (c). Note como as saídas ativas em BAIXO e como as entradas de habilitação são representadas. Embora a porta AND de habilitação esteja indicada externamente ao bloco descodificador, ela faz parte do circuito interno do CI. O 74HC138 é a versão CMOS de alta velocidade deste descodificador.

Exercício 1:

Indique os estados das saídas do 74LS138 para cada um dos seguintes conjuntos de entrada.

(a) $E_3 = \overline{E}_2 = 1$; $\overline{E}_1 = 0$; $A_2 = A_1 = 1$; $A_0 = 0$

(b) $E_3 = 1$; $\overline{E}_2 = \overline{E}_1 = 0$; $A_2 = 0$; $A_1 = A_0 = 1$

Exercício 2:

A Fig.4 mostra como quatro 74LS138s e um INVERSOR podem ser interligados para funcionarem como um descodificador 1 de 32. Os descodificadores foram identificados de Z_1 até Z_4 para facilitar a consulta, e as oito saídas de cada um são combinadas em 32 saídas. As saídas de Z_1 são O_0 a O_7; as saídas de Z_2 foram renomeadas para O_8 a O_{15}, respetivamente; as saídas de Z_3 foram renomeadas para O_{16} a O_{23}; e as saídas de Z_4 foram renomeadas para O_{24} a O_{31}.

Um código de entrada de cinco bits, $A_4A_3A_2A_1A_0$, ativa apenas uma destas 32 saídas para cada um dos 32 códigos de entrada possíveis.

(a) Que saída será ativada para $A_4A_3A_2A_1A_0 = 01101$?

(b) Que faixa de códigos de entrada ativará o chip Z_4?

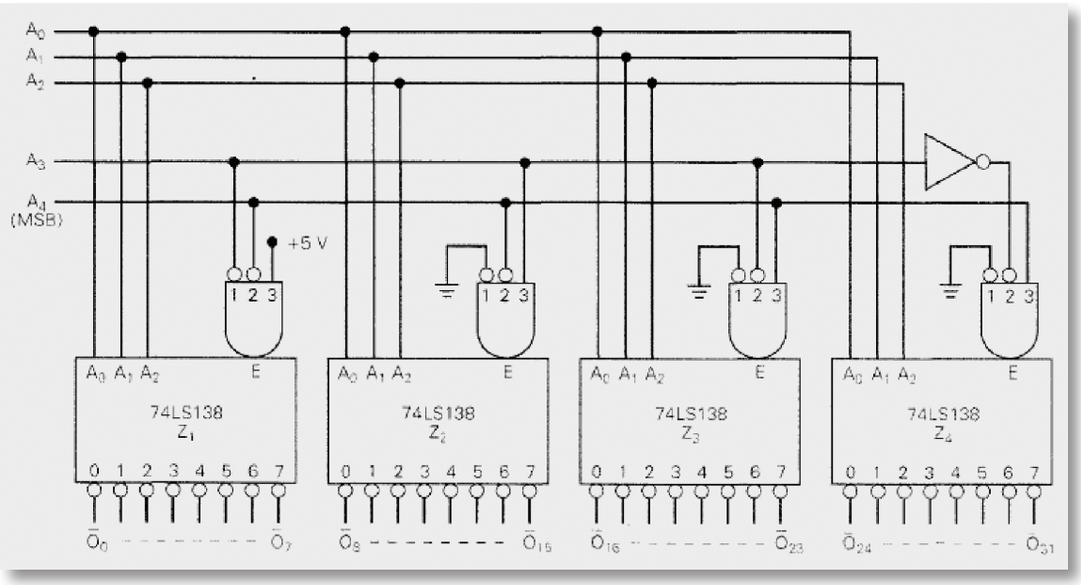


Fig. 4: Quatro 74LS138s formando um descodificador 5 para 32 (ou 1 de 32),

Descodificadores BCD para Decimal

A Fig.5 (a) mostra o diagrama lógico para o descodificador BCD para decimal 7442. Ele também está disponível como um 74LS42 ou um 74HC42. Cada saída vai para BAIXO apenas quando a entrada BCD correspondente é aplicada. Por exemplo, $\overline{O}_5$ vai para BAIXO somente com as entradas *DCBA* = 0101; $\overline{O}_8$ vai para BAIXO somente com *DCBA* = 1000. Para combinações de entrada que são inválidas para BCD, nenhuma das saídas será ativada. Este descodificador pode também ser denominado descodificador 4 para 10 ou um descodificador 1 de 10. O símbolo lógico e a tabela de verdade para o 7442 também são mostrados na figura.

Note que este descodificador não tem uma entrada de habilitação. No Problema 7 vamos ver como o 7442 pode ser útilizado como um descodificador 3 para 8 com a entrada *D* sendo usada como entrada de habilitação.

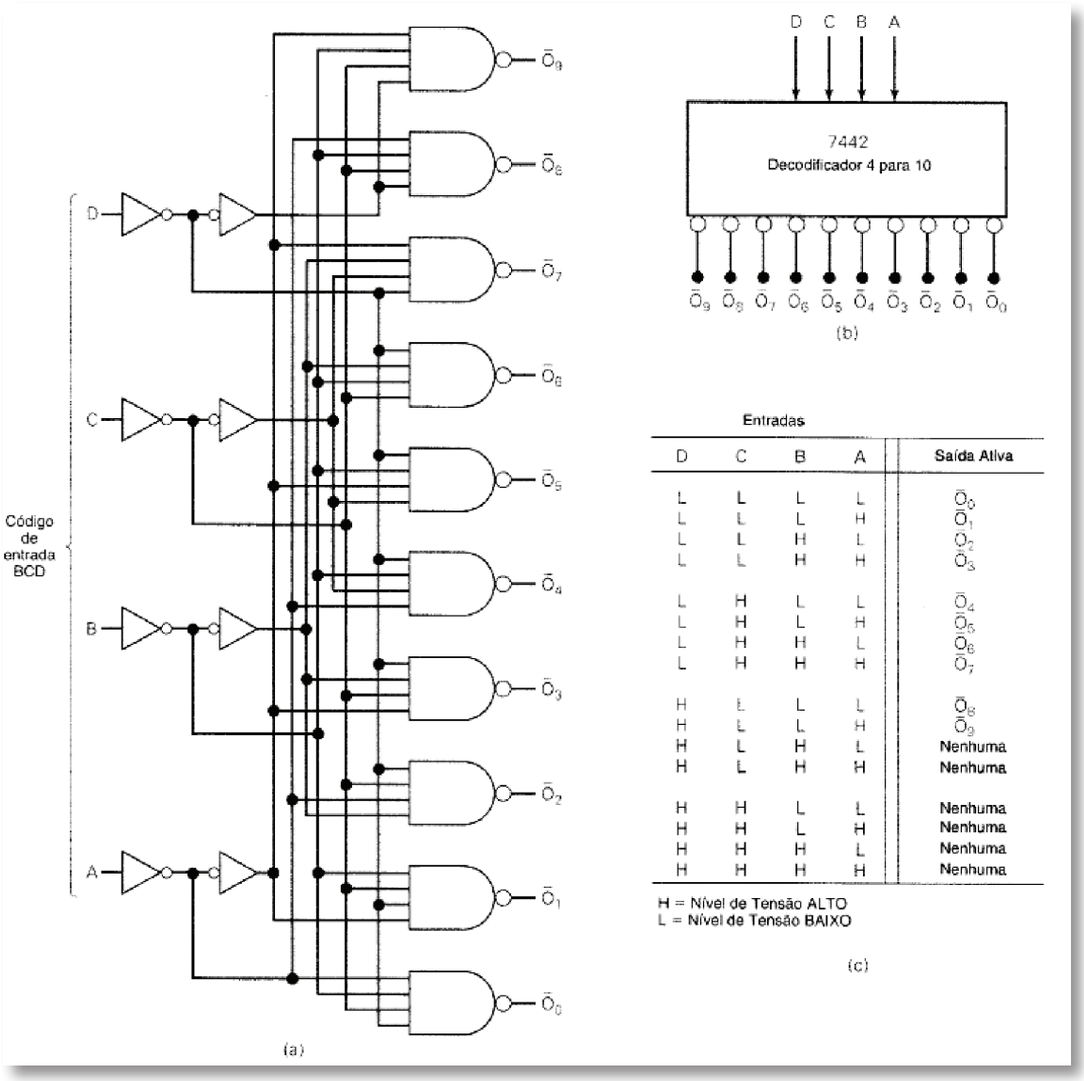


Fig. 5: (a) Diagrama lógico para o descodificador BCD para decimal 7442; (b) tabela de Verdade

O chip TTL 7445 e um descodificador/Driver BCD para decimal.

O termo driver é adicionado na sua descrição porque este CI tem saídas em coletor aberto que podem operar com tensões e correntes mais altas do que os limites de uma saída normal TTL. As saídas do 7445 podem absorver até 80 mA no estado BAIXO e podem ser levadas até 30 V no estado ALTO. Isto faz com que sejam apropriadas para acionar diretamente cargas tais como lâmpadas e LEDs indicadores, reles ou motores DC.

Aplicações de Descodificadores

Descodificadores são usados sempre que uma saída ou grupo de saídas são ativadas somente na ocorrência de uma combinação específica de níveis de entrada. Estes níveis de entrada são frequentemente fornecidos pelas saídas de um contador ou de um registador. Quando as entradas do descodificador vem de um contador que está a ser acionado continuamente, as saídas do descodificador serão ativadas sequencialmente, e elas podem ser utilizadas como sinais de temporização ou sequenciamento para ligar ou desligar dispositivos em determinados momentos. Um exemplo desta operação é apresentado na Fig.6 com um contador 74LS293 e o descodificador/driver 7445 descrito anteriormente.

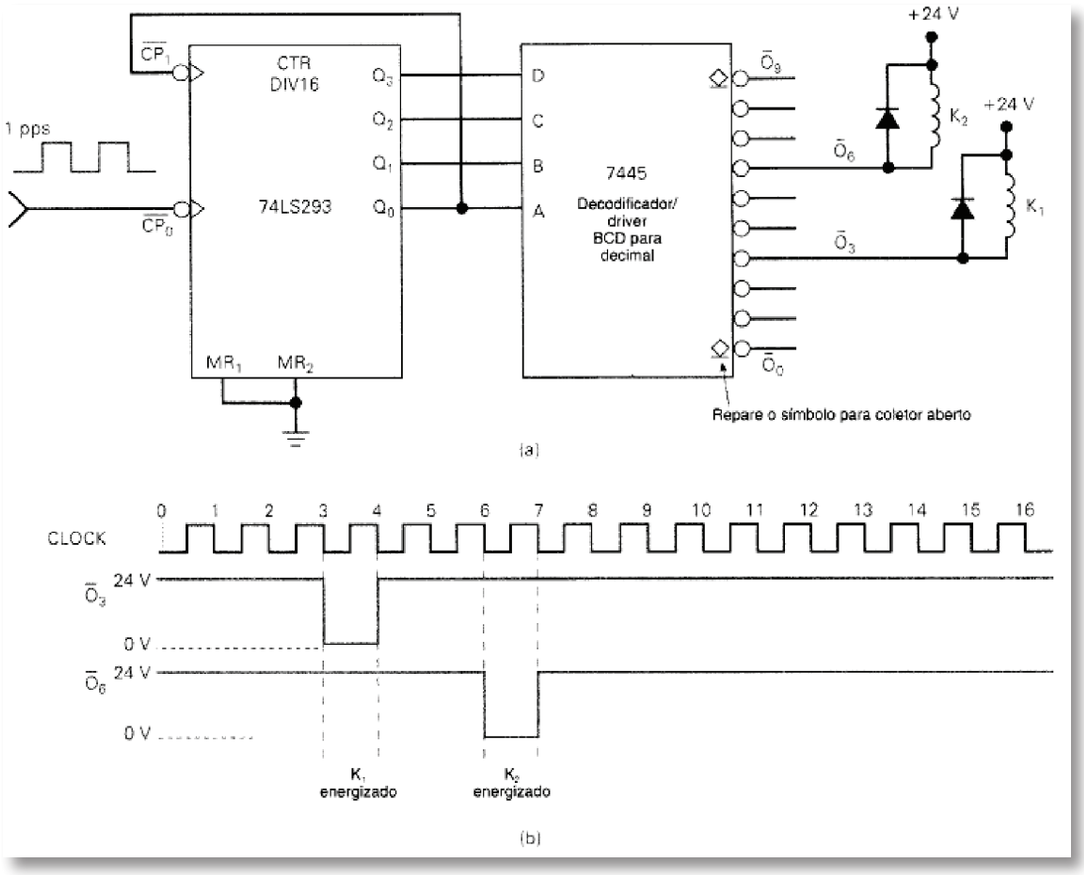


Fig. 6: Exemplo 3: combinação contador/descodificador útilizado para realizar operações de temporização e sequenciamento.

Exercício 1:

Descreva a operação do circuito na Fig.6 (a).

Nota:

Os descodificadores são amplamente útilizados nos sistemas de memória dos computadores, onde respondem aos endereços criados pelo processador central para aceder a uma posição de memória específica. Cada CI de memória contem muitos registadores que podem armazenar números binários (dados). Cada registador precisa ter o seu próprio endereço único, para distingui-lo de todos os outros registadores. Um descodificador é construído nos circuitos internos dos CIs de memória e permite que um determinado registador de armazenamento seja ativado quando uma única combinação de entrada (isto é, o seu endereço) é aplicada.

Num sistema, existem usualmente diversos CIs de memória combinados para implementar a capacidade de armazenamento completa. Um descodificador é usado para selecionar um chip de memória, em resposta a uma faixa de endereços, descodificando os bits mais significativos de endereço do sistema e habilitando (selecionando) um determinado chip. Em sistemas de memória mais complicados, os chips de memória são organizados em múltiplos bancos, os quais devem ser selecionados individualmente ou simultaneamente, dependendo de se o microprocessador deseja um ou mais bytes por vez. Isto significa que, sob certas circunstâncias, mais do que uma saída do descodificador deve ser ativada. Para sistemas deste tipo, um dispositivo de lógica programável frequentemente é usado para implementar o descodificador, pois um simples descodificador 1 de 8 sozinho não é suficiente. Dispositivos de lógica programável podem ser facilmente útilizados para aplicações específicas de descodificação.

Descodificadores/drivers BCD para 7 Segmentos

A maioria dos equipamentos digitais tem alguma maneira de mostrar as informações num formato, que possa ser prontamente compreendido pelo utilizador ou operador. Estas informações, frequentemente são dados numéricos, mas também podem ser alfanuméricos (números e letras). Um dos métodos mais simples e populares para a apresentação de dígitos numéricos, usa uma configuração de 7 segmentos Fig.7 (a) para formar os caracteres decimais de 0 a 9, e algumas vezes os caracteres hexadecimais de A até F. Um arranjo comum utiliza díodos emissores de luz (LEDs) para cada segmento. Controlando-se a corrente através de cada LED, alguns segmentos acendem e outros permanecem apagados, de modo que o padrão de caracter desejado seja criado. A Fig.7 (b) mostra os padrões de segmentos usados para formar os vários dígitos. Por exemplo, para mostrar um «6», os segmentos *a, c, d, e, f e g* são acesos, enquanto o segmento *b* fica apagado. Um descodificador /driver BCD para 7 segmentos é usado para receber uma entrada BCD de quatro bits e fornecer as saídas que ativam os segmentos apropriados para mostrar o dígito decimal. A lógica para este descodificador é mais complicada do que a lógica dos descodificadores que analisamos até agora, porque cada saída é ligada para mais do que uma combinação de entrada. Por exemplo, o segmento (*e*) deve, ser ativado para qualquer um dos dígitos 0, 2, 6 e 8, o que significa sempre que qualquer um dos códigos 0000, 0010, 0110 ou 1000 ocorrer.

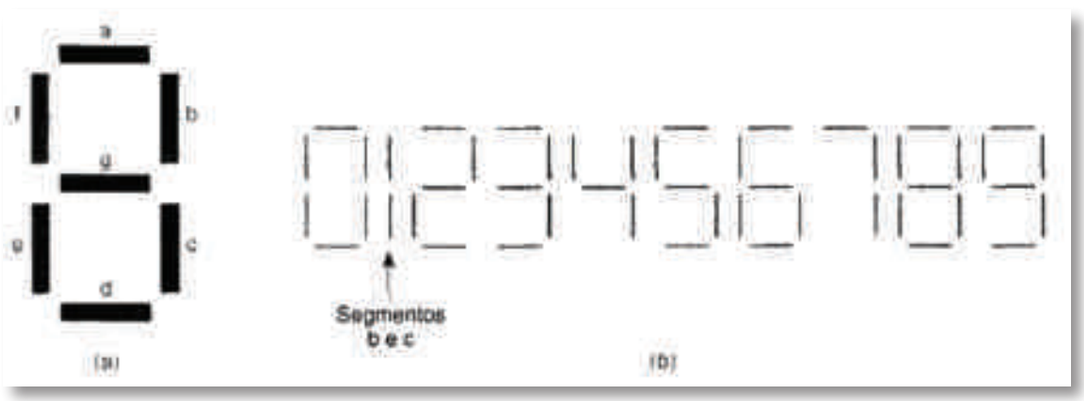


Fig. 7: (a) Configuração dos 7 segmentos; (b) segmentos ativos para cada dígito.

A Fig.8 (a) mostra um descodificador/driver BCD para 7 segmentos (TTL 7446 ou 7447) sendo usado para ligar um display a LEDs de 7 segmentos. Cada segmento consiste num ou dois LEDs. Os ânodos dos LEDs estão ligados em *Vcc* (+5 V). Os cátodos dos LEDs são ligados através de resistências limitadores de corrente nas saídas apropriadas do descodificador/driver. O descodificador/driver tem saídas ativas em BAIXO, de coletor aberto, com transístores de ativação que podem absorver uma corrente razoavelmente grande. Isto é necessário porque os *displays* a LED podem precisar de 10 a 40 mA por segmento, dependendo do tipo e tamanho.

Para ilustrar a operação deste circuito, vamos supor que a entrada BCD seja *D* = 0, *C = 1, B = 0, A* = 1, que é 5 em BCD. Com estas entradas, as saídas *a, f , g, c* e *d* do descodificador/driver ficavam acionadas em BAIXO (ligadas à terra), permitindo assim a corrente fluir através dos segmentos de LED *a, f g, c* e *d,* e portanto apresentava o número 5. As saídas (*b*) e (*e*) estavam em ALTO (em aberto), de modo que os segmentos de LED *b e e* não acendessem.

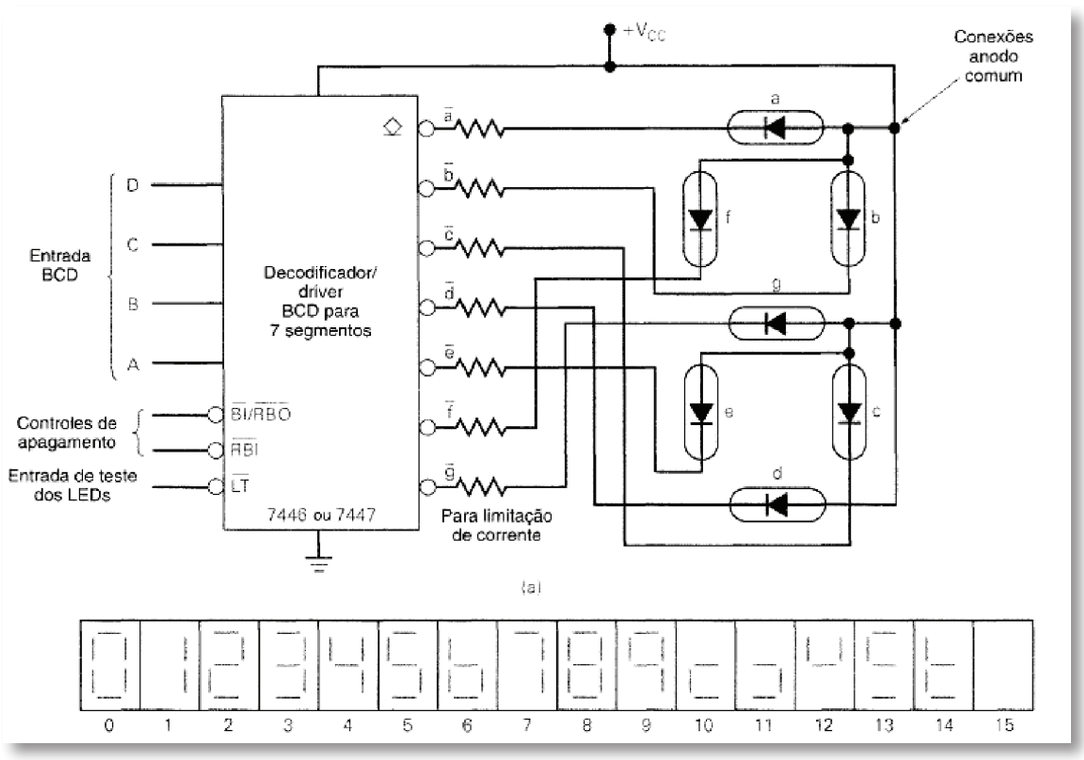


Fig. 8: (a) Descodificador/driver BCD para 7 segmentos a ligar um display de 7 segmentos a LEDs de ânodo comum; (b) padrões de segmentos para todos os códigos de entrada possíveis.

Os descodificadores/drivers 7446/47 são projetados para ativar segmentos específicos mesmo para códigos de entrada não-BCD (maiores do que 1001). A Fig.8 (b) mostra os padrões de segmentos ativos para todos os códigos de entrada possíveis desde 0000 até 1111. Note que um código de entrada 1111 (15) apaga todos os segmentos.

Descodificadores/drivers de 7 segmentos, tais como o 7446/47, são exceções para a regra de circuitos descodificadores ativarem apenas uma saída para cada combinação de entrada. Eles ativam um padrão único de saída para cada combinação de entrada.

Displays a LED de Cátodo Comum versus Ânodo Comum

O display a LEDs usado na Fig.8 é do tipo ânodo comum porque os ânodos de todos os segmentos são ligados juntos em *Vcc.* Um outro tipo de display a LEDs de 7 segmentos utiliza um arranjo com cátodo comum, onde os cátodos de todos os segmentos são ligados juntos à terra. Este tipo de display deve ser acionado por um descodificador/driver BCD para 7 segmentos com saídas ativas em ALTO, que aplicam uma tensão nos ânodos daqueles segmentos que devem ser ativados. Tendo em vista que cada segmento precisa de uma corrente de 10 a 20 mA para acender, dispositivos TTL e CMOS não são normalmente útilizados para acionar diretamente *displays* de cátodo comum.

Um circuito de interface com transístores é frequentemente usado entre os chips de descodificação e o display de cátodo comum.

Exercício 1:

Cada segmento de um display típico a LEDs de 7 segmentos opera com 10 mA e 2,7 V para um brilho normal. Calcule o valor da resistência limitadora de corrente necessária para produzir aproximadamente 10 mA por segmento.

Displays de Cristal Líquido

Um display a LEDs gera ou emite energia luminosa conforme a corrente que passa por cada segmento. Um display de cristal líquido (LCD — *liquid crystal display)* controla a reflexão da luz disponível. A luz disponível pode ser simplesmente a luz ambiente, tal como a luz do sol ou a iluminação normal.

LCDs reflexivos útilizam luz ambiente. A luz também pode ser fornecida por uma pequena fonte luminosa que faz parte da unidade de display; LCDs *backlit* utilizam este método. Em qualquer dos casos, os LCDs obtiveram ampla aceitação devido ao baixíssimo consumo de energia quando comparado aos LEDs, especialmente em equipamentos que operam com baterias, tais como calculadoras, relógios digitais e instrumentos eletrónicos portáteis de medição. Os LEDs têm a vantagem de proporcionarem um display com brilho muito mais intenso, que, ao contrário dos LCDs reflexivos, são facilmente visíveis em áreas escuras ou pouco iluminadas.

Basicamente, os LCDs operam com sinais AC de baixa tensão (tipicamente de 3 a 15 V rms) e baixa frequência (25 a 60 Hz) e consomem uma corrente muito pequena. Frequentemente são configurados como *displays* de 7 segmentos para leituras numéricas conforme é apresentado na Fig.9 (a).

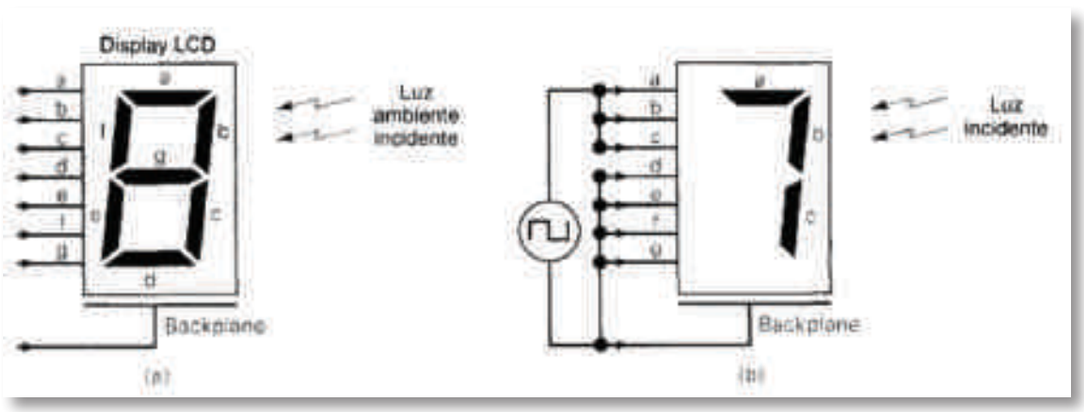


Fig. 9: Display de cristal líquido: (a) configuração básica; (b) aplicando-se uma tensão entre o segmento e o backplane, o segmento e ligado. Uma tensão zero desliga o segmento.

A tensão AC necessária para ligar um segmento é aplicada entre o segmento e o backplane, que é comum a todos os segmentos. O segmento e o backplane formam um condensador que consome uma corrente muito baixa enquanto a frequência AC é mantida baixa. Geralmente não é inferior a 25 Hz, porque poderia produzir uma cintilação visível.

Uma explicação, certamente simplificada, de como um LCD funciona seria a seguinte. Quando não há diferença de tensão entre um segmento e o backplane, diz-se que o segmento está desativado (OFF). Os segmentos *d, e, f e g,* na Fig.9 (b), estão OFF e refletirão a luz incidente, de modo que pareçam invisíveis contra o fundo. Quando uma tensão AC apropriada é aplicada entre um segmento e o backplane, o segmento é ativado (ON). Os segmentos *a, b* e c, na Fig.9 (b), estão ON e não vão refletir a luz incidente, e portanto vão parecer escuros contra o fundo.

Como ativar um LCD

Um segmento de LCD ligará quando uma tensão AC for aplicada entre o segmento e o backplane, e desligará quando não existir tensão entre os dois. Em vez de gerar um sinal AC, é prática comum produzir a tensão AC necessária aplicando-se ondas quadradas fora de fase ao segmento e ao backplane. Isto está ilustrado na Fig.10 para um segmento.

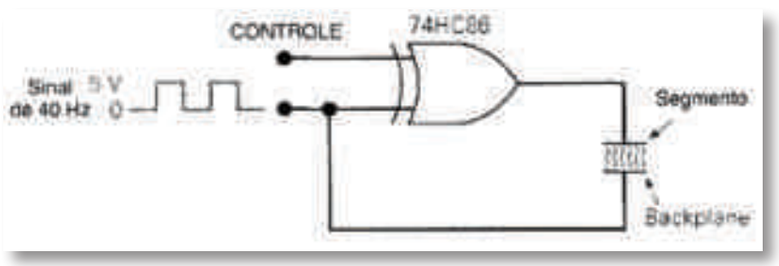


Fig. 10: Método para ativar um segmento de LCD. Quando o CONTROLO está em BAIXO, o segmento está desligado. Quando CONTROLO está em ALTO, o segmento está ligado.

Uma onda quadrada de 40 Hz é aplicada ao backplane e também na entrada de uma XOR CMOS 74HC86. A outra entrada da XOR é uma entrada de controlo, que controla se o segmento esta ligado ou desligado.

Quando a entrada de controlo estiver em BAIXO, a saída da XOR será exatamente a mesma onda quadrada de 40 Hz, de modo que os sinais aplicados ao segmento e ao backplane

serão iguais. Como não existe diferença de tensão, o segmento estará desligado. Quando a entrada de controlo estiver em ALTO, a saída da XOR será o inverso da onda quadrada de 40 Hz, de modo que o sinal aplicado ao segmento estará fora de fase com o sinal aplicado ao backplane. Como resultado, a tensão do segmento estará alternadamente em +5 V e em — 5 V em relação ao backplane. Esta tensão AC ligará o segmento.

A mesma ideia pode ser estendida para um display LCD de 7 segmentos completo conforme mostrado na Fig.11.

Nesta figura, o descodificador/driver CMOS BCD para 7 segmentos 74HC4511 fornece os sinais de controlo para cada uma das sete XORs para os sete segmentos. O 74HC4511 tem saídas ativas em ALTO, já que um nível ALTO é necessário para ligar um segmento. O descodificador/ driver e as portas XOR da Fig.11 estão disponíveis num único chip. Tal dispositivo é o chip CMOS 74HC4543. Recebe o código BCD de entrada e fornece as saídas necessárias para acionar diretamente os segmentos LCD.

De um modo geral, dispositivos CMOS são usados para acionar LCDs por duas razões:

(1) Precisão de muito menos potência do que os TTL e são mais adequados para aplicações controladas por baterias onde os LCDs são usados;

(2) O estado BAIXO TTL não é exatamente 0 V e pode ser até 0,4 V. Isto produziria um componente de tensão DC entre o segmento e o backplane que encurtaria consideravelmente a vida de um LCD.

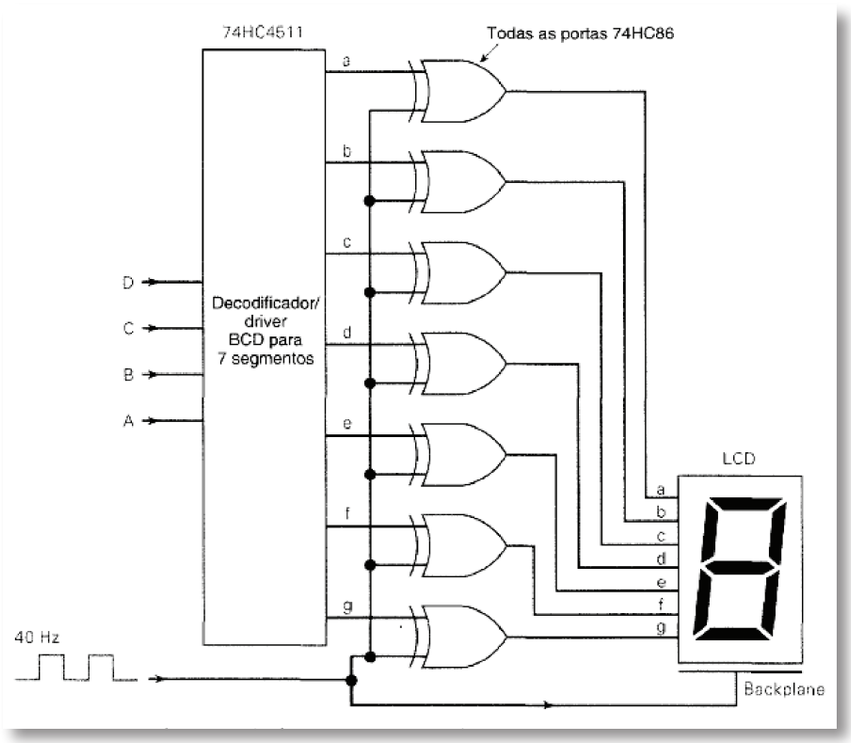


Fig. 11: Método para ativar um LCD de 7 segmentos.

Tipos de LCDs

Cristais líquidos estão disponíveis como *displays* numéricos decimais de vários dígitos. Eles são oferecidos em muitos tamanhos e com diversos caracteres especiais, tais como dois pontos (:) para *displays* de relógios; indicadores +/— para voltímetros digitais; vírgula decimal para calculadoras e indicadores do estado da bateria, tendo em vista que muitos dispositivos com LCD são alimentados por bateria. Estes *displays* devem ser ativos por um chip descodificador/driver tal como o 74HC4543 ou o 74C945.

Um display LCD mais complicado, mas prontamente disponível, é o módulo LCD alfanumérico. Ele é disponibilizado por diversas companhias em vários formatos, tais como o de uma linha com 16 caracteres ou o de 4 linhas de 40 caracteres.

A interface com estes módulos foi padronizada, de modo que um módulo LCD de um fabricante útiliza os mesmos sinais e formatos de dados. O módulo inclui alguns chips VLSI que tornam o dispositivo simples de usar. Oito linhas de dados são utilizadas para enviar o código ASCII do que você deseja mostrar no display. Estas linhas também recebem códigos de controlo especiais para o registador de comandos do LCD.

Três outras entradas (Register Select, Read/Write e Enable) são usadas para controlar a localização, a direção e a temporização da transferência de dados. Conforme os caracteres vão sendo enviados para o módulo, são armazenados na sua própria memória e apresenta-os no display.

Outros módulos LCDs permitem ao útilizador criar um display gráfico, controlando os pontos individuais da tela chamados pixels. Grandes painéis LCD podem ser atualizados a uma taxa elevada, produzindo filmes de alta qualidade. Nestes *displays*, as linhas de controlo são organizadas numa malha de linhas e colunas. Na intersecção de cada linha e coluna está um pixel que atua como uma «janela», que pode ser eletronicamente aberta e fechada para controlar a quantidade de luz que é transmitida através da célula. A tensão entre a linha e a coluna determina o brilho de cada pixel. Num computador laptop, um número binário para cada pixel é armazenado na memória de «vídeo». Estes números são convertidos para tensões que são aplicadas ao display.

Cada pixel num display colorido é formado na verdade por três subpixels. Estes subpixels controlam a luz que passa através dos filtros vermelho, verde e azul para produzir a cor de cada pixel. Numa tela LCD de 640 por 480 existiriam 640 X 3 conexões para colunas

e 480 conexões para linhas, para um total de 2400 conexões para o LCD. Obviamente, o circuito de execução para tal dispositivo é um circuito VLSI muito complicado que é fabricado como uma parte do display.

Codificadores

A maioria dos descodificadores aceita um código de entrada e produz um nível ALTO (ou BAIXO) numa e somente uma linha de saída. Por outras palavras, podemos dizer que um descodificador identifica, reconhece ou testa um código específico. O oposto deste processo de descodificação é chamado codificação e é realizado por um circuito lógico denominado codificador**.** Um codificador tem um certo número de linhas de entrada, onde somente uma delas é ativada por vez, e produz um código de saída de N bits, dependendo da entrada que está ativada. A Fig.12 é o diagrama geral para um codificador com *M* entradas e *N* saídas. Neste caso as entradas são ativas em ALTO, o que significa que estão normalmente em BAIXO.

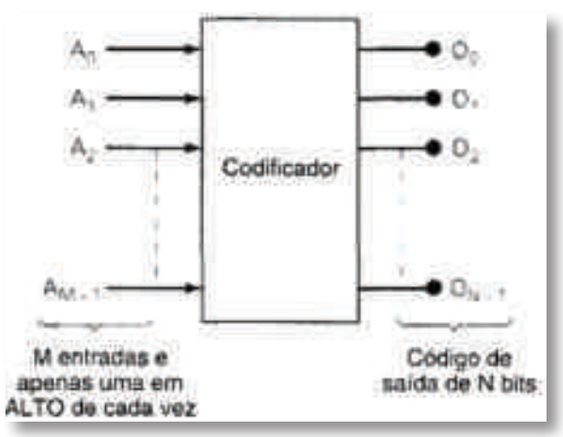


Fig. 12: Diagrama geral do codificador

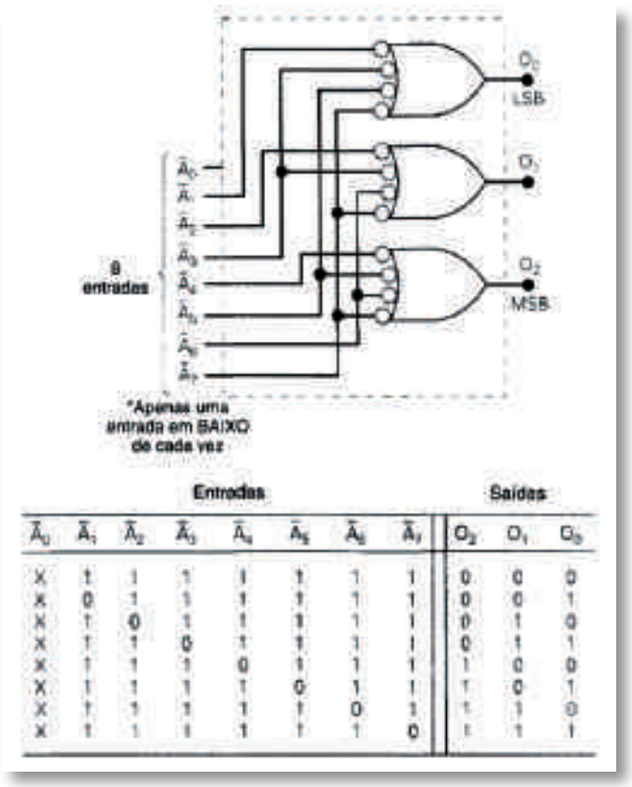
Já vimos que um descodificador binário para octal (descodificador de 3 linhas para 8 linhas) aceita um código de entrada de três bits e ativa uma de entre oito linhas de saída correspondente a este código. Um codificador octal para binário (codificador de 8 linhas para 3 linhas) realiza a função oposta: ele aceita oito linhas de entrada e produz um código de saída de três bits correspondente à entrada ativada. A Fig.13 mostra o circuito lógico e a tabela de verdade para um codificador octal para binário com entradas ativas em BAIXO.

Seguindo a lógica, podemos verificar que um nível BAIXO em qualquer uma das entradas de cada vez, produzirá o código binário de saída correspondente para aquela entrada. Por exemplo, um nível BAIXO em $\overline{A}_3$ (enquanto todas as outras entradas estiverem em

ALTO) produzirá $O_2 = 0$, $O_1 = 1$ e $O_0 = 1$, que é o código binário para 3. Note que $\overline{A}_0$ não está ligado nas portas lógicas, pois as saídas do codificador normalmente estarão em 000 quando nenhuma das entradas de $\overline{A}_1$ até $\overline{A}_9$ estiver em BAIXO.

Exemplo 1:

Determine as saídas do codificador na Fig.13 quando $\overline{A}_3$ e $\overline{A}_5$, estiverem simultaneamente em BAIXO.



$\overline{A}_0$	$\overline{A}_1$	$\overline{A}_2$	$\overline{A}_3$	$\overline{A}_4$	$\overline{A}_5$	$\overline{A}_6$	$\overline{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

Fig. 13: Circuito lógico para um codificador octal para binário (8 para 3 linhas). Para operação correta, apenas uma entrada deve estar ativa de cada vez.

Codificadores de Prioridade

Este último exemplo identifica um inconveniente do circuito codificador simples da Fig.13 quando mais do que uma entrada é ativada ao mesmo tempo. Uma versão modificada deste circuito, é chamado de codificador de prioridade, inclui a lógica necessária para garantir que, quando duas ou mais entradas estiverem ativadas, o código de saída corresponderá à entrada com número mais alto. Por exemplo, tanto $\overline{A}_3$ quanto $\overline{A}_5$, quando estiverem em BAIXO, o código de saída será 101 (5). Analogamente, quando $\overline{A}_6$,

$\overline{A}_2$ e $\overline{A}_0$ estiverem todas em BAIXO, o código de saída será 110 (6). O 74148, 74LS148 e o 74HC148 são codificadores de prioridade octal para binário.

Codificador de Prioridade Decimal para BCD 74147

A Fig.14 mostra o símbolo lógico e a tabela de verdade para o 74147 (74LS147, 74HC147), que funciona como um codificador de prioridade decimal para BCD. Possui nove entradas ativas em BAIXO representando os dígitos decimais de 1 até 9, e produz o código BCD invertido correspondente a entrada ativada de número mais elevado.

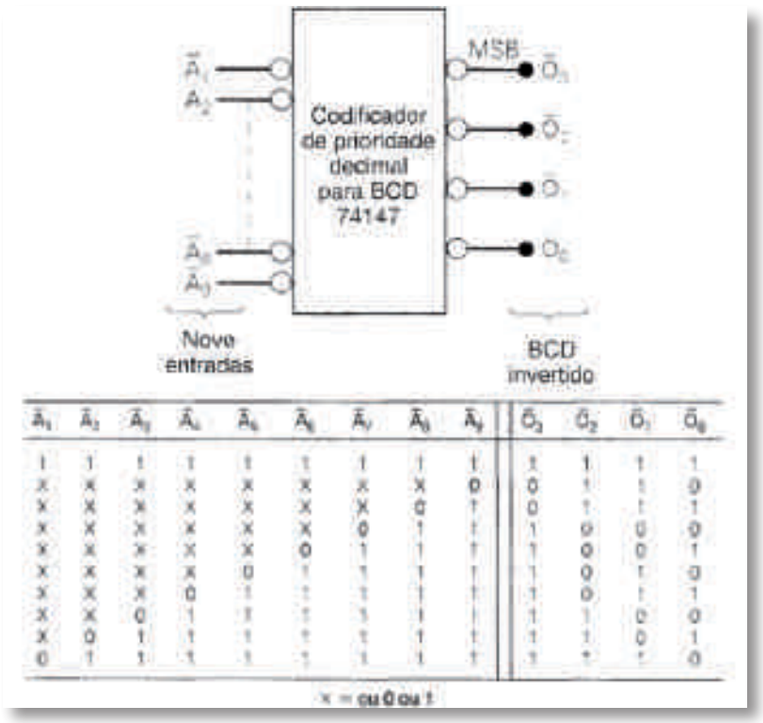


Fig. 14: Codificador de prioridade decimal para BCD 74147

Vamos examinar a tabela de verdade para ver como este CI funciona. A primeira linha na tabela mostra todas as entradas nos seus estados inativos em ALTO. Para esta condição, as saídas são 1111, que é o inverso de 0000, o código BCD para 0. A segunda linha na tabela indica um nível BAIXO em $\overline{A}_9$, que, independentemente dos estados das outras entradas, produzirá um código de saída 0110, que é o inverso de 1001, o código BCD para 9. A terceira linha mostra que um nível BAIXO em $\overline{A}_8$, desde que $\overline{A}_9$ esteja em ALTO, irá produzir um código de saída 0111, o inverso de 1000, o código BCD para 8. Da mesma maneira, as linhas restantes na tabela mostram que um nível BAIXO em qualquer entrada,

desde que todas as outras de mais alta ordem estejam em ALTO, produzirá o inverso do código BCD daquela entrada.

As saídas do 74147 normalmente estarão em ALTO quando nenhuma das entradas estiver a ser ativa. Isto corresponde à condição de entrada decimal 0. Não existe entrada $\overline{A}$ *0,* tendo em vista que o codificador assume o estado de entrada decimal 0 quando todas as outras entradas estão em ALTO. As saídas BCD invertidas do 74147 podem ser convertidas para BCD normal, colocando um INVERSOR para cada uma.

Exercício 2:

Determine os estados das saídas na Fig.14 quando $\overline{A}_7$, $\overline{A}_5$ e $\overline{A}_3$ estiverem em BAIXO e todas as outras entradas estiverem em ALTO.

Multiplexers

Um sistema moderno de som estéreo pode ter um interruptor que selecione a música de uma das três fontes: cassete, compact disc (CD), ou um rádio. O Interruptor seleciona um dos sinais eletrónicos de uma destas três fontes e envia-o para o amplificador de potência e para os altifalantes (colunas). De um modo simplificado, isto é o que um multiplexador (MUX) faz: seleciona um entre vários sinais de entrada e envia-o para a saída.

Um multiplexer digital ou seletor de dados é um circuito lógico que aceita diversos dados digitais de entrada e seleciona um deles, num determinado instante, para a saída. A seleção do sinal de entrada desejado, para a saída é controlado pelas entradas de seleção (frequentemente chamadas de entradas de endereço). A Fig.15 mostra o diagrama funcional de um multiplexador digital geral. As entradas e saídas são desenhadas com setas mais largas em vez de linhas; isto indica que elas podem, na verdade, representar mais do que uma linha de sinal.

O multiplexer atua como um interruptor digital controlado de varias posições, onde o código digital aplicado nas entradas de seleção controla qual das entradas de dados será comutada para a saída. Por exemplo, a saída Z será igual à entrada de dados I_0 para um determinado código de seleção; Z será igual a I_1 para outro determinado código de seleção; e assim por diante. Por outras palavras, um multiplexer seleciona 1 entre *N* dados de entrada e transmite o dado selecionado para um único canal de saída. Isto é chamado de multiplexação.

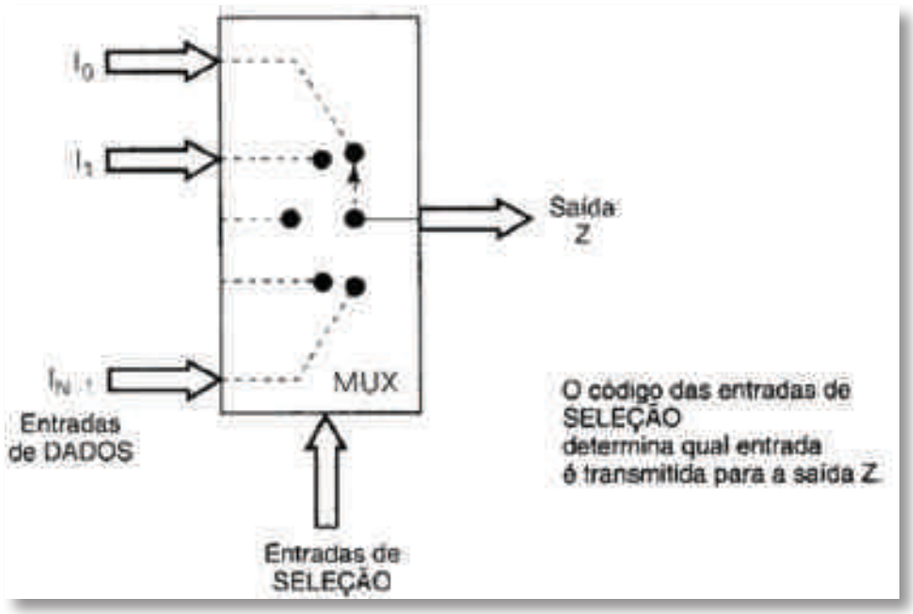


Fig. 15: Diagrama funcional de um Multiplexer (MUX) digital

Multiplexer básico de duas entradas

A Fig.16 mostra o circuito lógico para um multiplexer de duas entradas, com as entradas de dados I_0 e I_1, e uma entrada de seleção *S*. O nível lógico aplicado na entrada *S* determina qual das portas AND é habilitada, de modo que os seus dados passem pela porta OR para a saída Z. Analisando de outra maneira, a expressão booleana para a saída é:

$$Z = I_0\bar{S} + I_1S$$

Com S = 0, esta expressão torna-se em:

$$Z = I_0.1 + I_1.0 \qquad (porta\,2\,habilitada)$$

$$Z = I_0$$

Que indica que Z será idêntica ao sinal de entrada I_0, que pode ser um nível lógico fixo ou um sinal lógico variável no tempo. Com *S* = 1, a expressão fica:

$$Z = I_0.0 + I_1.1 \qquad (porta\,1\,habilitada)$$

$$Z = I_1$$

Provando que a saída Z será idêntica ao sinal de entrada I_1.

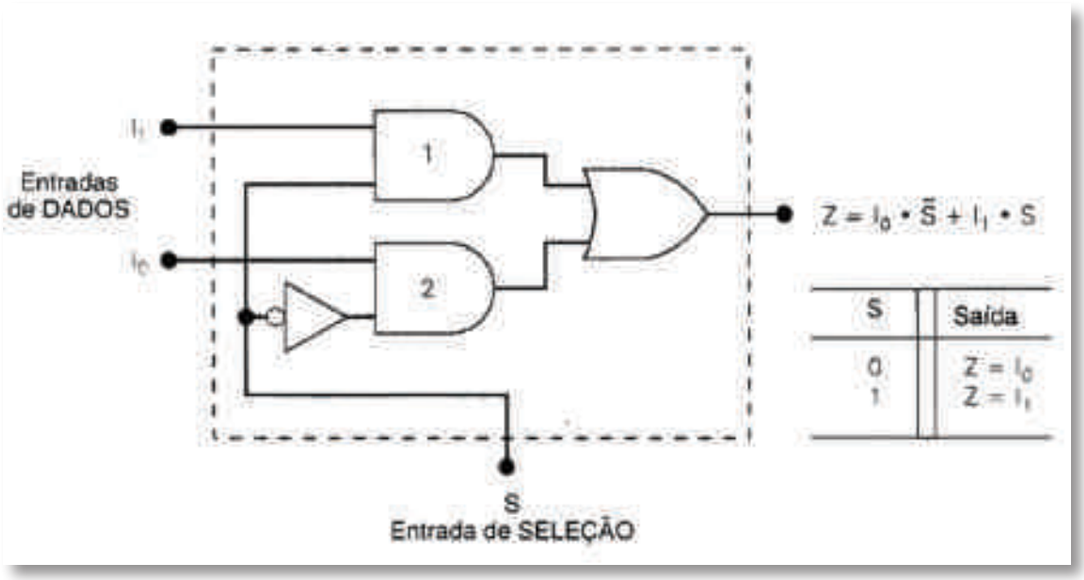


Fig. 16: Multiplexer de duas entradas

Um exemplo de onde um MUX de duas entradas pode ser útilizado é num computador que útilize dois sinais diferentes de CLOCK: um clock de alta velocidade (digamos, 10 MHz) para alguns programas, e um clock lento (digamos, 4,77 MHz) para outros. Utilizando o circuito da Fig. 16, o clock de 10 MHz seria ligado em I_0, e o clock de 4,77 MHz, em I_1. Um sinal da área de controlo lógico do computador acionaria a entrada de seleção, para controlar qual dos sinais de clock apareceria na saída Z, para ser levado a outros circuitos do computador.

Multiplexer de Quatro Entradas

A mesma ideia básica pode ser usada para formar o multiplexer de quatro entradas mostrado na Fig.17. Neste caso, existem quatro entradas, as quais são seletivamente transmitidas para a saída, de acordo com as quatro combinações possíveis das entradas de seleção S_1S_0. Cada entrada de dados é selecionada com uma combinação diferente de níveis das entradas de seleção. I_0 é selecionado com $\bar{S}_1\bar{S}_0$, de modo que I_0 passará pela sua porta AND para a saída Z somente quando $S_1 = 0$ e $S_0 = 0$. A tabela na figura fornece as saídas para os outros três códigos de seleção.

Multiplexers de duas, quatro, oito e dezasseis entradas estão disponíveis nas famílias lógicas TTL e CMOS. Estes CIs básicos podem ser combinados para a multiplexação de um maior número de entradas.

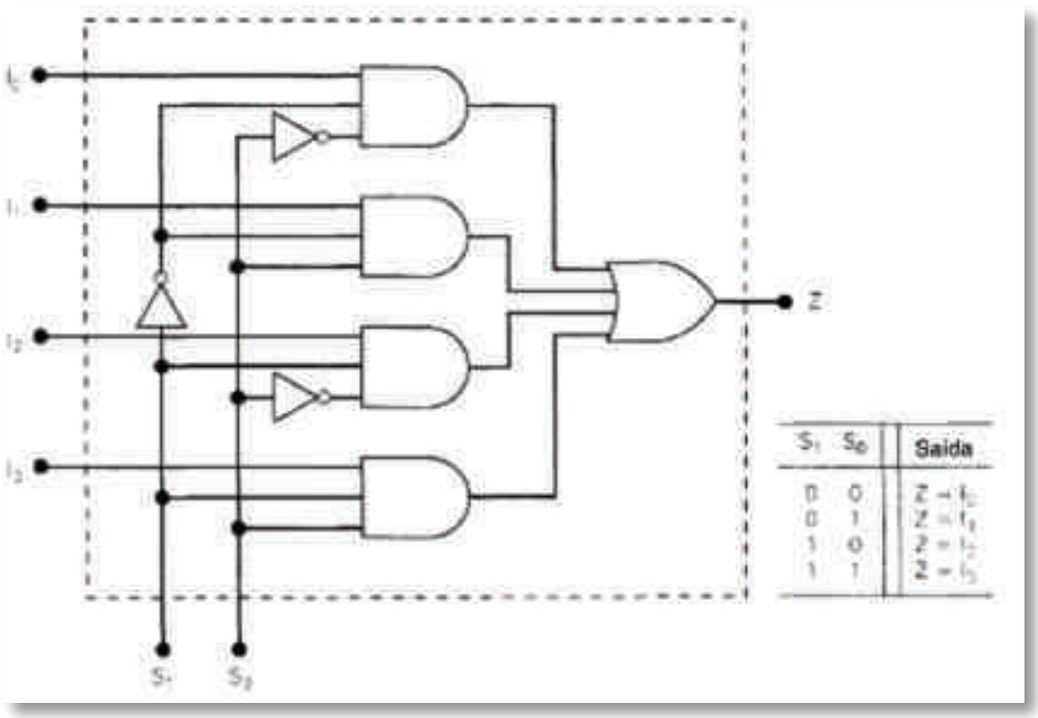
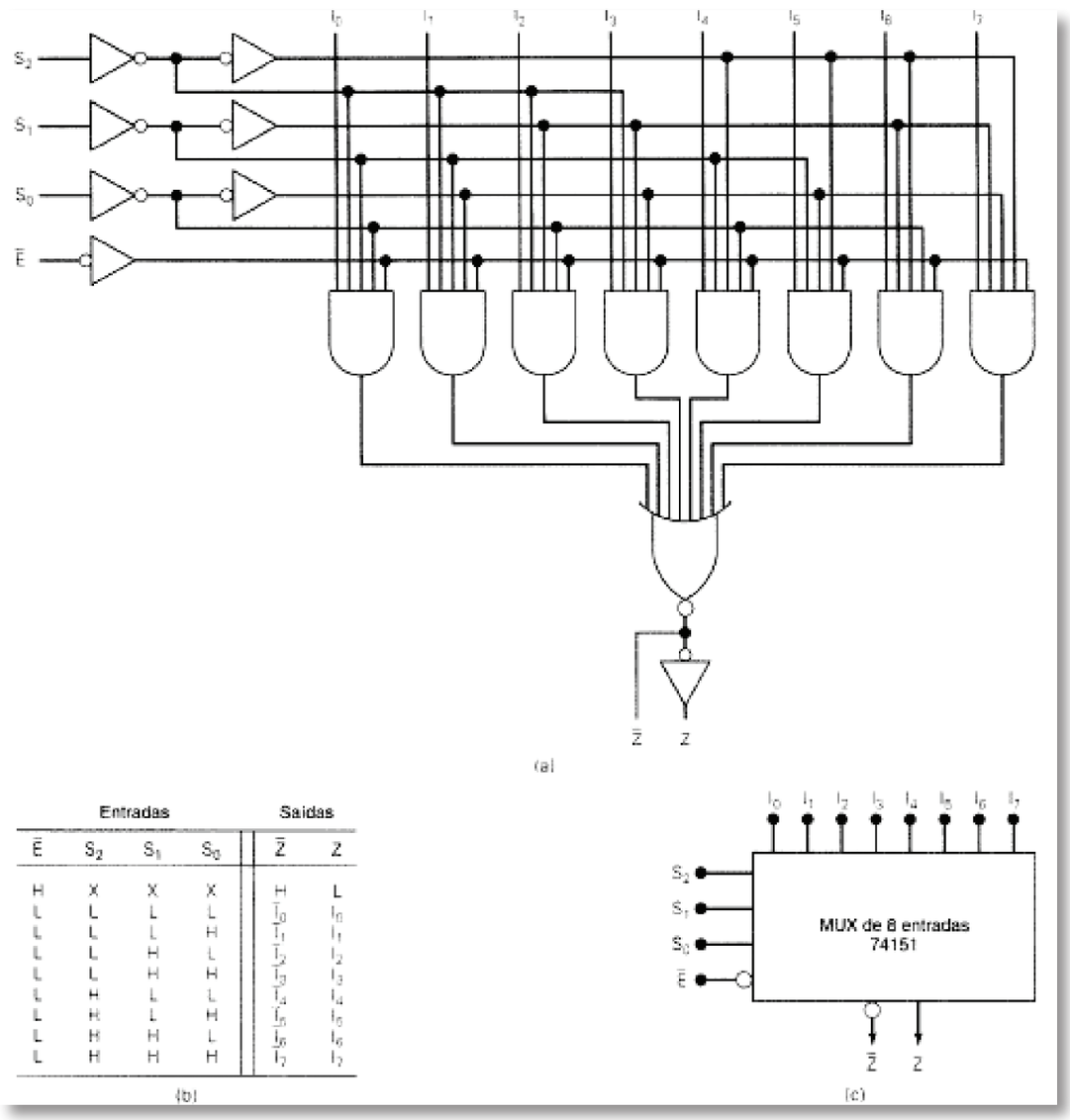


Fig. 17: Multiplexer de quatro entradas

Multiplexer de Oito Entradas

A Fig.18 (a) mostra o diagrama lógico para o multiplexador de oito entradas 74151 (74LS151, 74HC151). Este multiplexer tem uma entrada de habilitação $\overline{E}$ e fornece tanto a saída normal quanto a saída invertida. Quando $\overline{E}$ = 0, as entradas de seleção $S_2S_1S_0$ selecionarão uma das entradas de dados (de I_0 a I_7) para passar para a saída Z. Quando $\overline{E}$ =1, o multiplexer está inibido, de modo que Z = 0, não importando o código de seleção de entrada. Esta operação está resumida na Fig.18 (b), e o símbolo lógico do 74151 e mostrado na Fig.18 (c).



Entradas				Saídas	
$\overline{E}$	S_2	S_1	S_0	$\overline{Z}$	Z
H	X	X	X	H	L
L	L	L	L	$\overline{I}_0$	I_0
L	L	L	H	$\overline{I}_1$	I_1
L	L	H	L	$\overline{I}_2$	I_2
L	L	H	H	$\overline{I}_3$	I_3
L	H	L	L	$\overline{I}_4$	I_4
L	H	L	H	$\overline{I}_5$	I_5
L	H	H	L	$\overline{I}_6$	I_6
L	H	H	H	$\overline{I}_7$	I_7

Fig. 18: (a) Diagrama lógico para o multiplexer 74151; (b) tabela de verdade; (c) símbolo lógico.

Exercício 1:

O circuito na Fig.19 útiliza dois 74HC151s, um INVERSOR e uma porta OR. Descreva a operação do circuito.

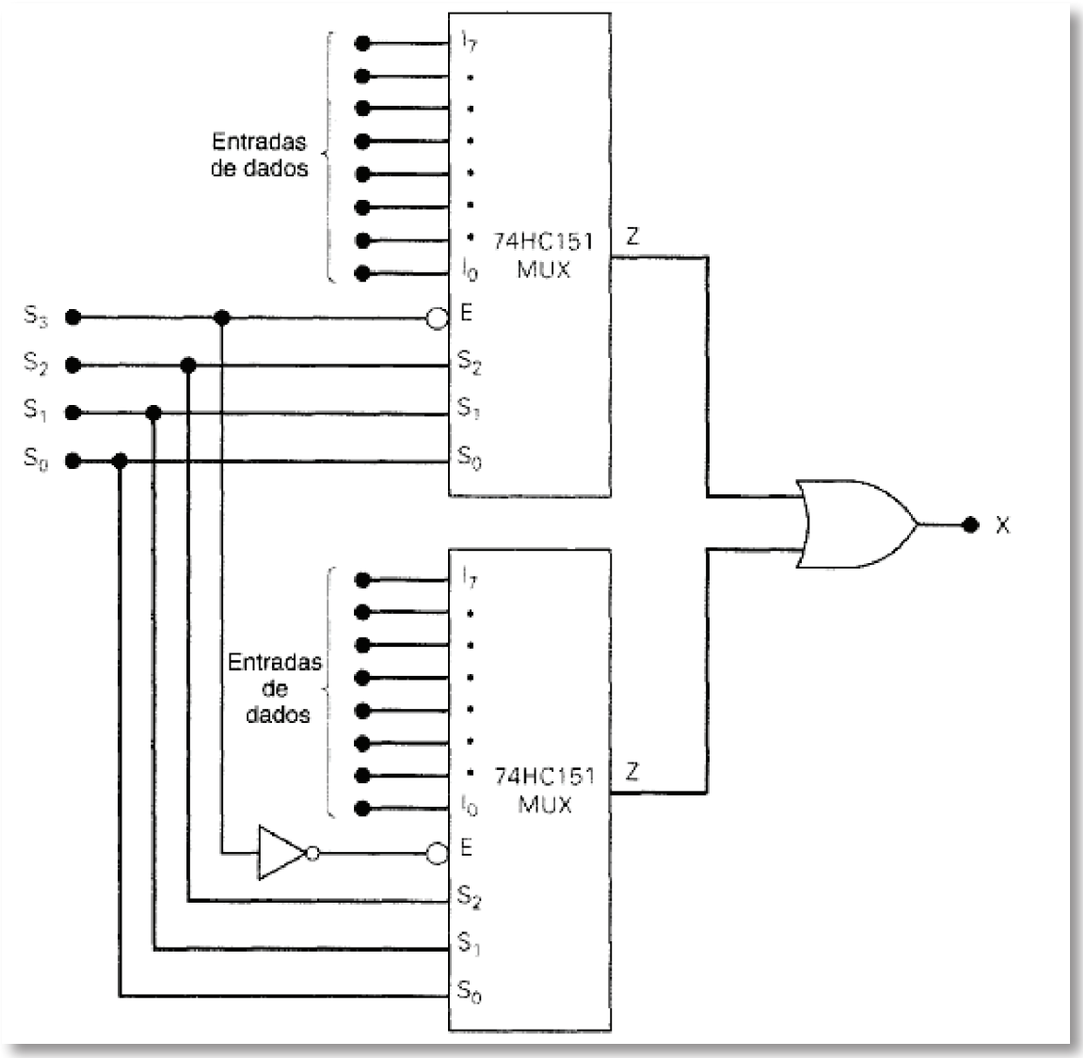


Fig. 19: Exemplo 1: dois 74HC151s combinados para formar um multiplexer de 16 entradas.

Aplicações de Multiplexers

Circuitos multiplexers encontram números e diversas aplicações em sistemas digitais de todos os tipos. Estas aplicações incluem seleção de dados, escolha de dados, sequência de operações, conversões paralelo-série, gerador de formas de onda e de funções lógicas.

Escolha de Dados

Multiplexers podem escolher dados de diversas fontes para um destino. Uma aplicação típica útiliza multiplexadores 74LS157 para selecionar e mostrar o conteúdo de um entre dois contadores BCD onde usa apenas um único conjunto de descodificador/driver e display a LEDs. A organização do circuito é apresentada na Fig. 20.

Cada contador consiste em dois estágios BCD em cascata, e cada um é acionado por seu próprio sinal de clock. Quando a linha SELECIONA_CONTADOR estiver em ALTO, as saídas do contador 1 passarão pelos multiplexers para os descodificadores/drivers para serem mostradas nos *displays* a LEDs. Quando SELECIONA_CONTADOR = 0, as saídas do contador 2 passarão pelos multiplexers para os *displays*. Deste modo, o conteúdo decimal de um ou de outro contador será apresentado sob o controle da entrada SELECIONA_CONTADOR. Uma situação comum onde isto poderia ser usado seria num relógio digital. Os circuitos do relógio digital contêm muitos contadores e registadores que tratam dos segundos, minutos, horas, dias, meses, alarme, e assim por diante. Um esquema de multiplexação, tal como este, permite que diferentes dados sejam apresentados no número limitado de *displays*.

O propósito da técnica de multiplexação, da maneira como é usada aqui, é partilhar os circuitos descodificadores/drivers e *displays* entre os dois contadores, em vez de ter um conjunto de descodificadores/drivers e *displays* para cada contador. Isto traz uma diminuição do número de conexões a serem realizadas, especialmente quando mais estágios BCD forem adicionados para cada contador.

Ainda mais importante, isto representa uma diminuição significativa de consumo, pois descodificadores/drivers e *displays* a LEDs absorvem, relativamente, grandes

quantidades de corrente da fonte *Vcc.* Naturalmente, esta técnica tem a limitação de que apenas o conteúdo de um contador pode ser mostrado de cada vez. Entretanto, em várias aplicações isto não é um inconveniente. Um interruptor mecânico poderia ter sido usado para realizar a função de comutar primeiro um contador e depois o outro para os descodificadores/drivers e *displays*, mas o número de contatos necessários, a complexidade das ligações e as dimensões físicas seriam desvantagens em relação ao método puramente lógico da Fig.20.

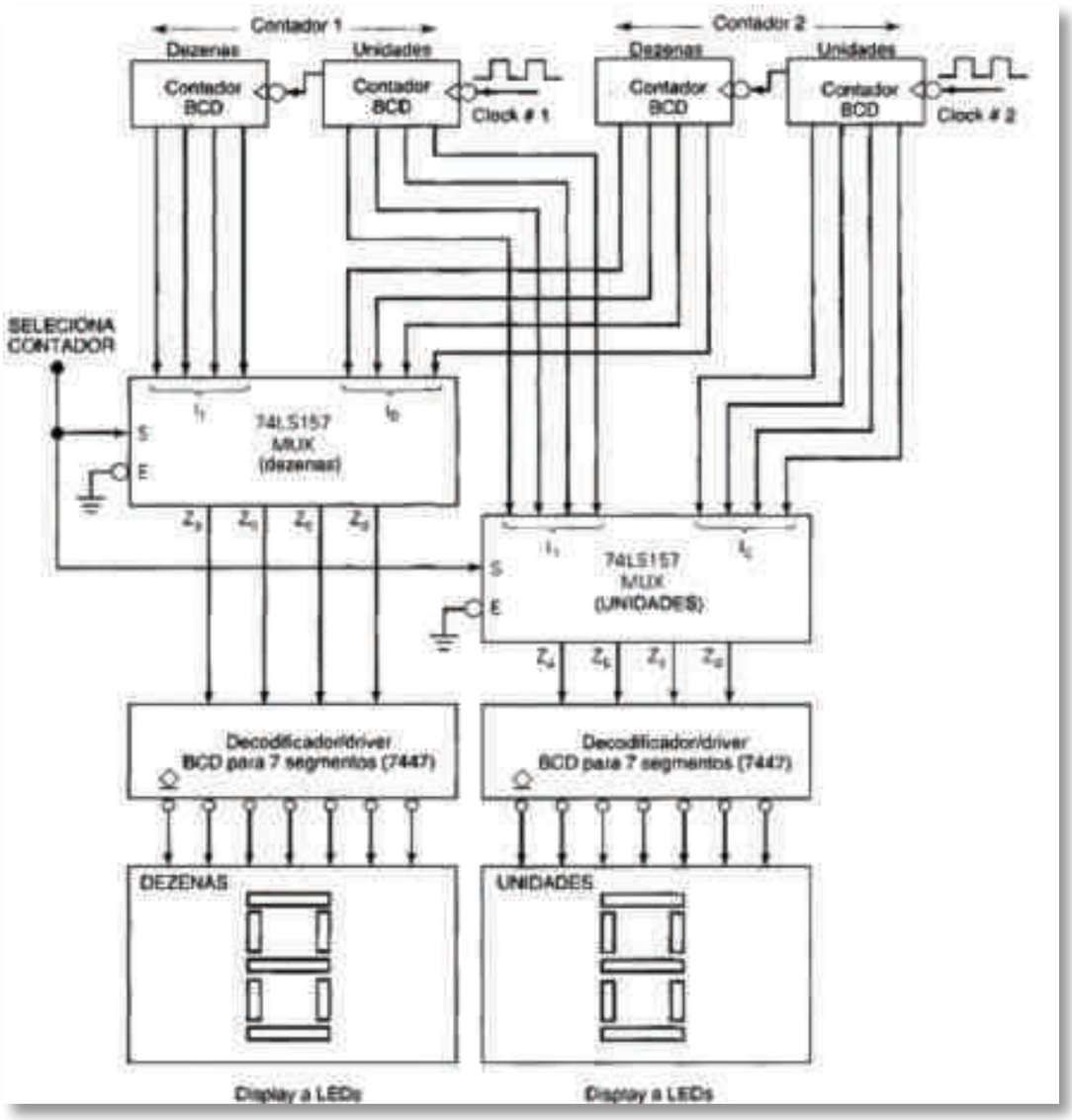


Fig. 20: Sistema para mostrar o valor de dois contadores BCD, um de cada vez.

Conversão Paralelo-Série

Muitos sistemas digitais processam os dados binários na forma paralela (todos os bits simultaneamente) porque é mais rápido. Entretanto, quando os dados devem ser transmitidos por distâncias relativamente longas, o esquema paralelo não é desejável, pois precisa de um grande número de linhas de transmissão. Por esta razão, dados binários ou informações na forma paralela são frequentemente convertidos para o formato série antes de serem transmitidos para um destino remoto. Um método para realizar esta conversão paralelo-série útiliza um multiplexer, conforme ilustrado na Fig.21.

Os dados são apresentados na forma paralela nas saídas do registador X e colocados no multiplexador de oito entradas. Um contador de três bits (modulo 8) é usado para fornecer os bits do código de seleção $S_2S_1S_0$ de modo que faça ciclos desde de 000 até 111, conforme os pulsos de clock são aplicados. Desta maneira, a saída do multiplexer será X_0 durante o primeiro período de clock, X_1 durante o segundo período de clock, e assim por diante. A saída Z tem uma forma de onda que é a representação série do dado paralelo de entrada. As formas de onda da figura são para o caso em que $X_7X_6X_5X_4X_3X_2X_1X_0$ = 10110101. Este processo de conversão gasta oito ciclos de clock no total.

Note que X_0 (o LSB) é transmitido primeiro e X_7 (MSB) é transmitido por ultimo.

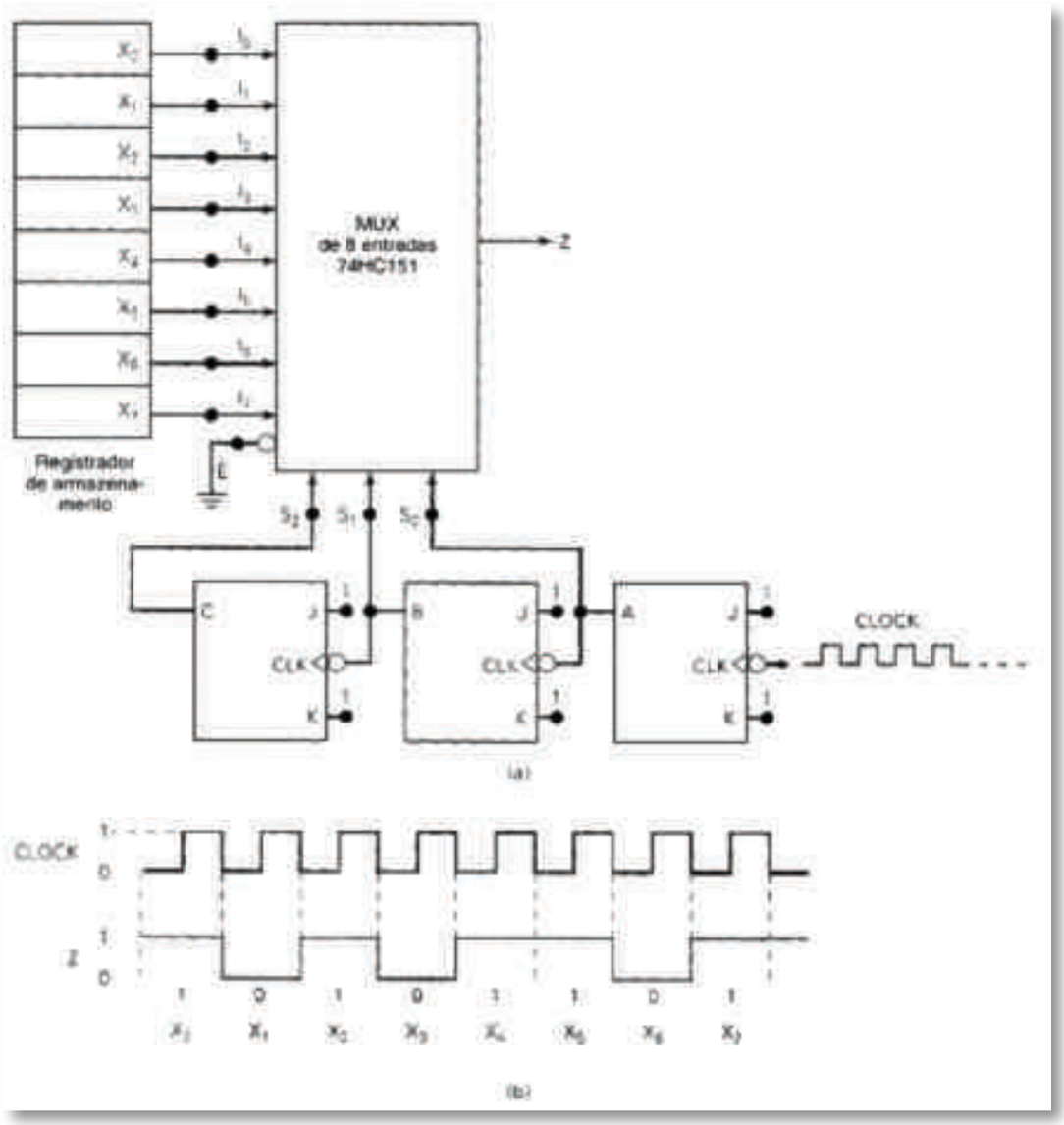


Fig. 21: (a) Conversor paralelo-série; (b) forma de onda para X7X6X5X4X3X2X1X0 = 10110101.

Sequência de operações

O circuito da Fig. 22 utiliza um multiplexer de oito entradas como parte de um sequenciador de controle de sete passos, onde cada passo atua numa porção do processo físico que está a ser controlado. Isto poderia ser, por exemplo, um processo que misturasse dois ingredientes líquidos e cozinhasse a mistura. O circuito também usa um descodificador de 3 para 8 linhas e um contador binário de módulo 8.

A operação é descrita a seguir:

1. Inicialmente é feito um RESET ao contador para o estado 000.

As saídas do contador são levadas para as entradas de seleção do multiplexer e para as entradas do descodificador.

Assim, a saída do descodificador $\overline{O}_0 = 0$ e todas as outras estão em 1, de modo que todas as entradas dos atuadores do processo estão em BAIXO. Todas as saídas dos sensores do processo iniciam em BAIXO. A saída do multiplexador $\overline{Z} = \overline{I}_0 = 1$, já que as entradas S são 000.

2. O pulso de início começa a operação de sequenciamento, fazendo um SET ao flip-flop Q_0, para ALTO, o que provoca com que o contador vá para o estado 001. Isto faz com que a saída $\overline{O}_1$ do descodificador vá para BAIXO, ativando assim o atuador 1, que é o primeiro passo no processo (abrir a válvula de enchimento #1).

3. Algum tempo depois, a saída do SENSOR 1 vai para ALTO, indicando que o primeiro passo foi finalizado (a chave da boia indica que o tanque esta cheio). Este nível ALTO aparece na entrada I_1 do multiplexer. Ele é invertido e alcança a saída $\overline{Z}$ pois o código de seleção do contador é 001.

4. A transição de $\overline{Z}$ para BAIXO é levada ao *CLK* do flip-flop Q_0. Esta transição negativa avança o contador para o estado 010.

5. A saída do descodificador $\overline{O}_2$? agora vai para BAIXO, e ativa o atuador 2, que é o segundo passo do processo (abrir a válvula de enchimento #2). Z agora é igual a $\overline{I}_2$ (o código de seleção é 010). Como a saída do SENSOR 2 ainda esta em BAIXO, Z vai para ALTO.

6. Quando o segundo passo do processo é concluído, a saída do SENSOR 2 vai para ALTO, produzindo um nível BAIXO em $\overline{Z}$ e avançando o contador para 011.

7. Esta mesma ação é repetida para cada um dos outros passos. Quando o sétimo passo é concluído, a saída do SENSOR 7 vai para ALTO, fazendo o contador ir de 111 para 000, onde permanece até que um outro pulso de INICIO recomece a sequência.

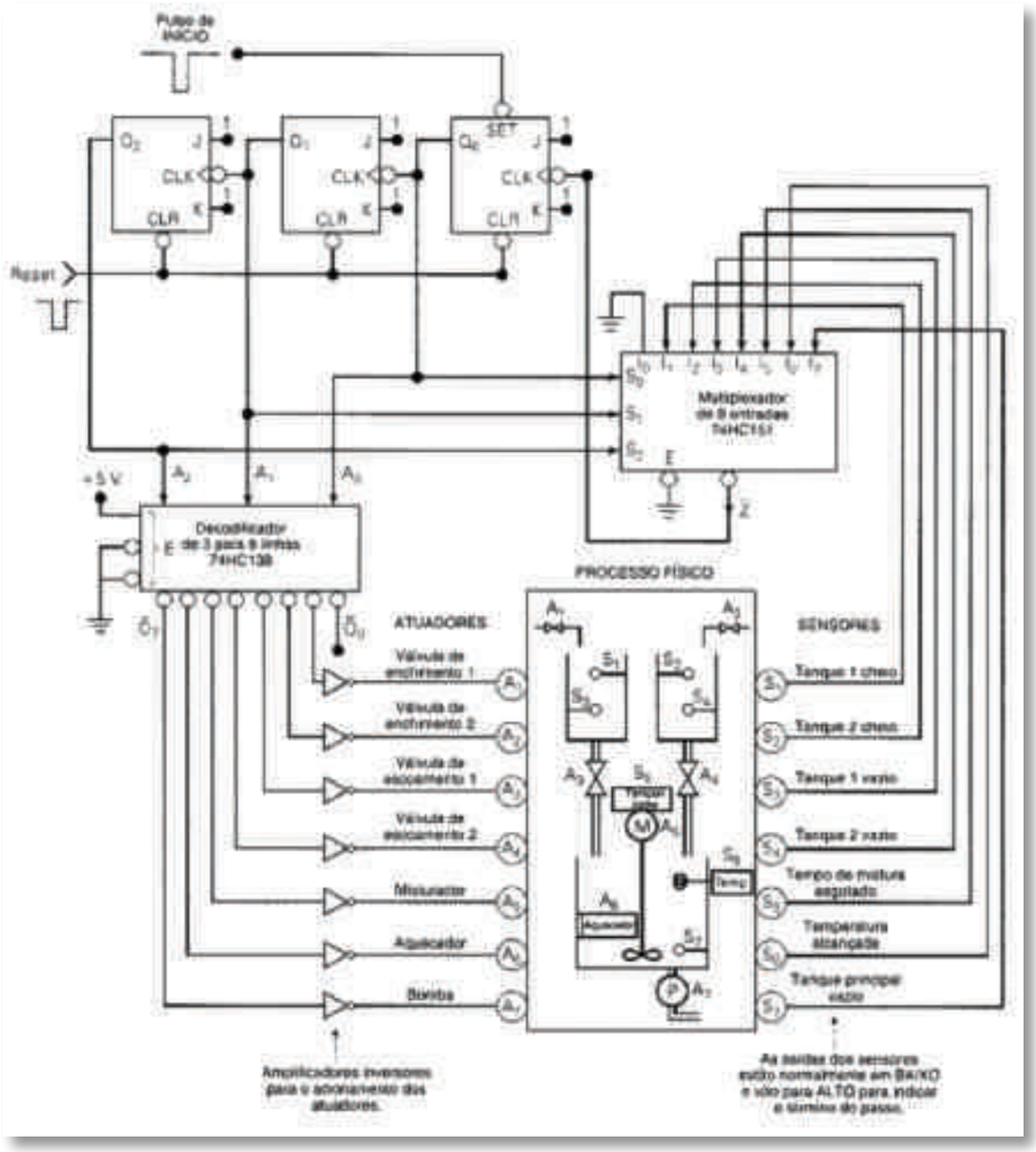


Fig. 22: Sequenciador de controlo de sete passos.

Geração de funções lógicas

Multiplexers podem ser usados para implementar funções lógicas diretamente a partir da tabela de verdade, sem necessidade de simplificação. Quando um multiplexador é usado para este propósito, as entradas de seleção são utilizadas como as variáveis lógicas, e cada entrada de dados é ligada permanentemente em ALTO ou BAIXO, conforme for necessário para satisfazer a tabela de verdade.

A Fig.23 ilustra como um multiplexador de 8 entradas pode ser usado para implementar o circuito lógico de uma determinada tabela de verdade. As variáveis de entrada *A, B e C* são ligadas em S_0, S_1 e S_2, respetivamente, de modo que os níveis destas entradas determinam qual a entrada de dados aparece na saída Z. De acordo com a tabela de verdade, Z deve estar em BAIXO quando *CBA* = 000. Assim, a entrada I_0 do multiplexer deve ser ligada em BAIXO.

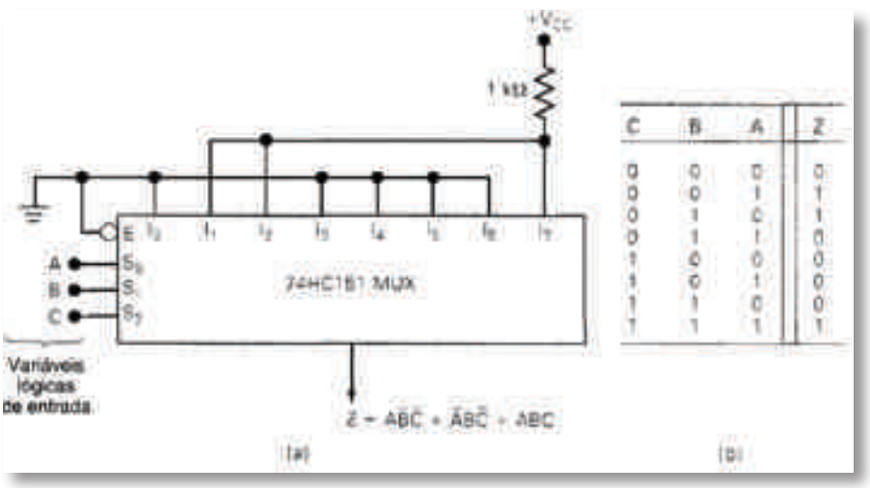
Analogamente, Z deve estar em BAIXO para *CBA* = 011, 100, 101 e 110, logo as entradas I_3, I_4, I_5 e I_6 também devem estar em BAIXO. As outras combinações de *CBA* devem produzir Z = 1, e portanto as entradas I_1, I_2 e I_7 do multiplexer são ligadas permanentemente em ALTO.

É fácil perceber que qualquer tabela de verdade de três variáveis pode ser implementada com este multiplexers de oito entradas. Este método de implementação frequentemente é mais eficiente do que usar portas lógicas separadas. Por exemplo, a expressão da soma de produtos para a tabela de verdade na Fig. 23 é:

$$Z = A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$$

Ela não pode ser simplificada nem algebricamente nem pelo mapa de Karnaugh, e portanto a implementação com portas necessitaria de três inversores e quatro portas NAND, perfazendo um total de dois CIs.

Existe um método ainda mais eficiente para usar multiplexers na implementação de funções lógicas. Este método permite ao projetista usar um multiplexer com três entradas de seleção (por exemplo, um 74HC151) para implementar uma função lógica de quatro variáveis.



C	B	A	Z
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

Fig. 23: Multiplexador útilizado para implementar uma função lógica descrita pela tabela de verdade.

Desmultiplexers

Um multiplexer recebe várias entradas e transmite uma delas para a saída. Um desmultiplexer (DEMUX) realiza a operação inversa: recebe uma única entrada e distribui-a por várias saídas. A Fig.24 mostra o diagrama funcional para um desmultiplexer digital. As setas largas para as entradas e saídas podem representar uma ou mais linhas. O código de seleção de entrada determina para qual saída a entrada de DADOS será transmitida. Por outras palavras, o desmultiplexer recebe uma fonte de dados e seletivamente distribui-a para 1 entre *N* canais de saída, como se fosse um interruptor de várias posições.

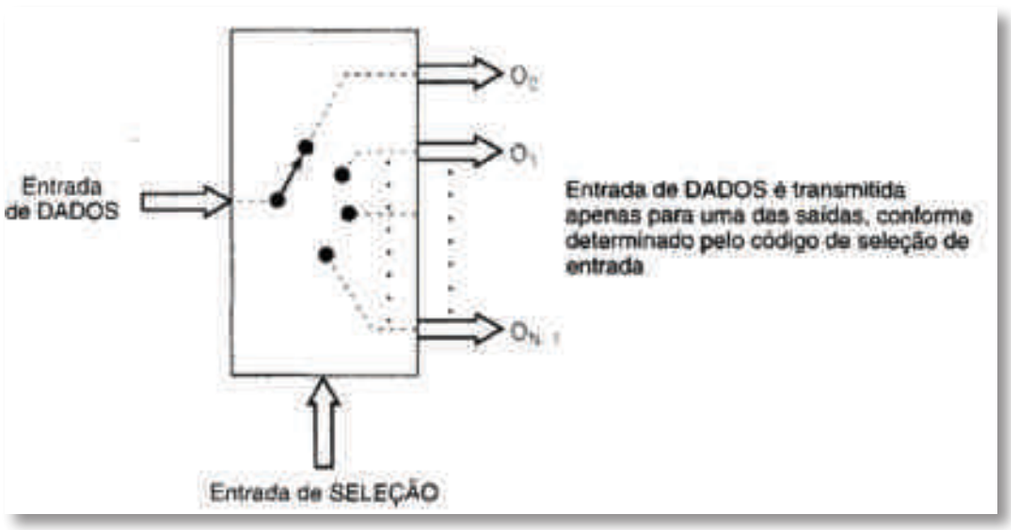


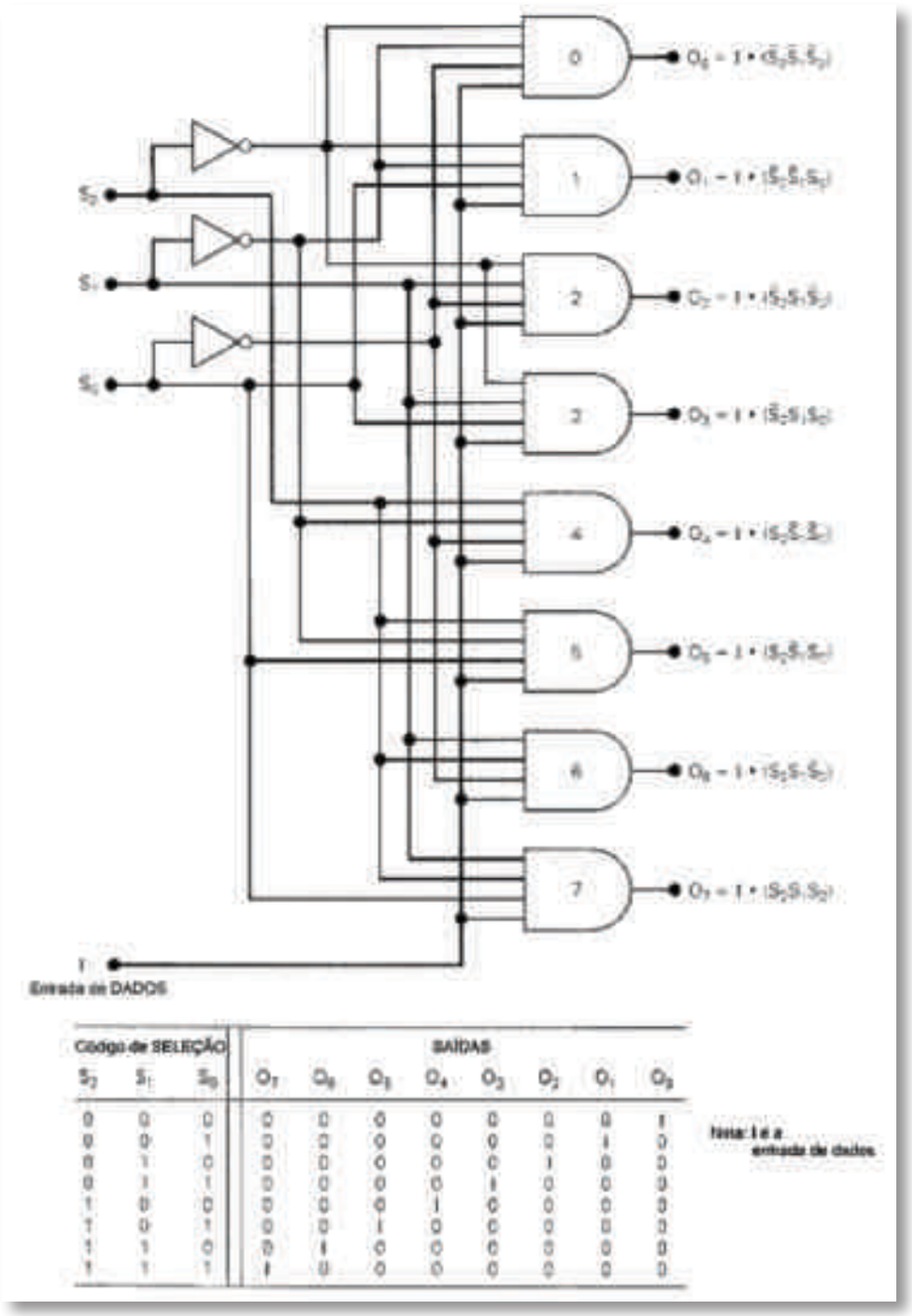
Fig. 24: Desmultiplexer genérico

Desmultiplexer de 1 para 8 Linhas

A Fig. 25 mostra o diagrama lógico para um desmultiplexer que distribui uma linha de entrada para oito linhas de saída. A única linha de entrada de dados I é ligada em todas as portas AND, mas somente uma destas portas será habilitada pelas linhas de seleção de entrada. Por exemplo, com $S_2S_1S_0$= 000, apenas a porta AND 0 estará habilitada, e a entrada de dados I aparecerá na saída O_0. Outros códigos de seleção fazem a entrada I alcançar as outras saídas. A tabela de verdade resume a operação.

O circuito desmultiplexer da Fig.25 é bastante similar ao circuito descodificador 3 para 8 na Fig.2, exceto que uma quarta entrada *(D* foi adicionada em cada porta. Foi ressaltado

anteriormente que muitos descodificadores possuem uma entrada de habilitação, que é uma entrada extra adicionada as portas dos descodificadores. Portanto, este tipo de chip descodificador pode ser usado como um desmultiplexer, com o código binário da entrada (por exemplo, *A, B* e *C* na Fig.2) servindo como entradas de seleção e a entrada de habilitação servindo como a entrada de dados I. Por esta razão, os fabricantes de CIs frequentemente chamam este dispositivo de *descodificador/desmultiplexer*, é ele pode ser usado para ambas as funções.



Código de SELEÇÃO			SAÍDAS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Nota: I é a entrada de dados.

Fig. 25: Desmultiplexer de 1 para 8

Estudámos anteriormente como o 74LS138 é usado como um descodificador 3 para 8. A Fig.26 mostra como pode ser útilizado como um desmultiplexer. A entrada de habilitação $\overline{E}_1$ é usada como a entrada de dados I, enquanto as outras duas entradas de habilitação são mantidas nos seus estados ativos. As entradas $A_2A_1A_0$ são utilizadas como o código de seleção. Para ilustrar a operação, vamos admitir que as entradas de seleção são 000. Com este código de entrada, a única saída que pode ser ativa será $\overline{O}_0$, enquanto todas as outras saídas estão em ALTO. $\overline{O}_0$ vai para BAIXO somente se $\overline{E}_1$ for para BAIXO, e ficará em ALTO se $\overline{E}_1$ for ALTO. Por outras palavras, $\overline{O}_0$ vai seguir o sinal em $\overline{E}_1$ (isto é, a entrada de dados, 1) enquanto todas as outras saídas permanecem em ALTO. Da mesma maneira, um código de seleção diferente aplicado em $A_2A_1A_0$ vai fazer a saída correspondente seguir a entrada de dados, I.

A Fig.26 (b) mostra as formas de onda típicas para o caso em que $A_2A_1A_0 = 000$ selecionem a saída $\overline{O}_0$. Para este caso, o sinal de dados aplicado em $\overline{E}_1$ será transmitido para $\overline{O}_0$, e todas as outras saídas permanecerão nos seus estados inativos em ALTO.

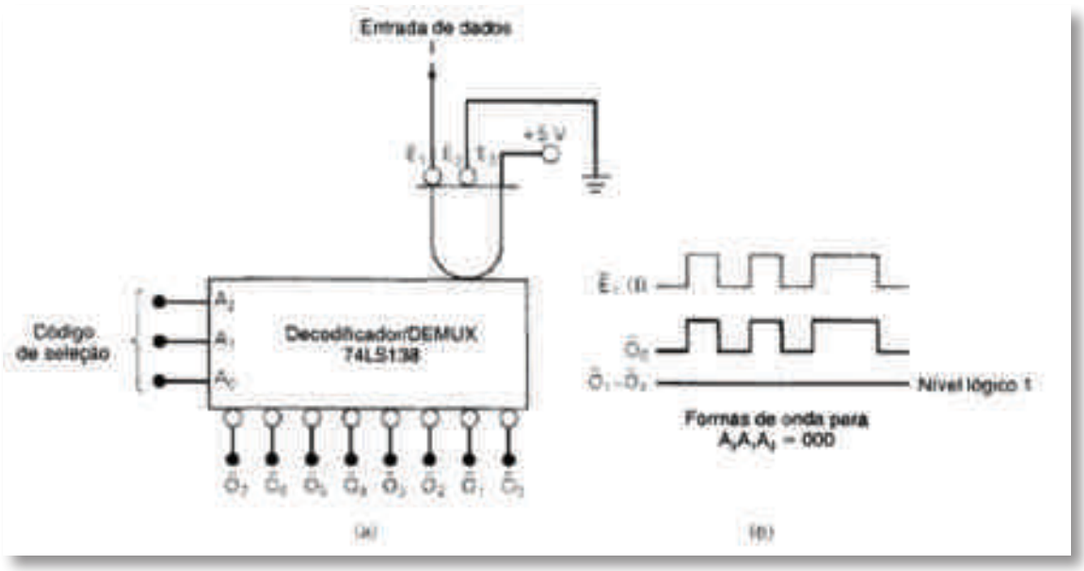


Fig. 26: (a) O descodificador 74LS138 pode funcionar como um desmultiplexer com $\overline{E}_1$ usado como entrada de dados, (b) Formas de onda típicas para o código de seleção de $A_2A_1A_0 = 000$ mostram que $\overline{O}_0$ é idêntico à entrada de dados I em $\overline{E}1$.

Desmultiplexer de Clock

São possíveis muitas aplicações do princípio de desmultiplexação.

A Fig.27 mostra o desmultiplexer 74LS138 a ser usado como um desmultiplexer de *clock.* Sob o controle das linhas de seleção, o sinal de clock é selecionado para o destino desejado. Por exemplo, com $S_2S_1S_0$ = 000, o sinal de clock aplicado a I vai aparecer na saída $\overline{O}_0$. Com $S_2S_1S_0$ = 101, o clock aparecera em $\overline{O}_5$.

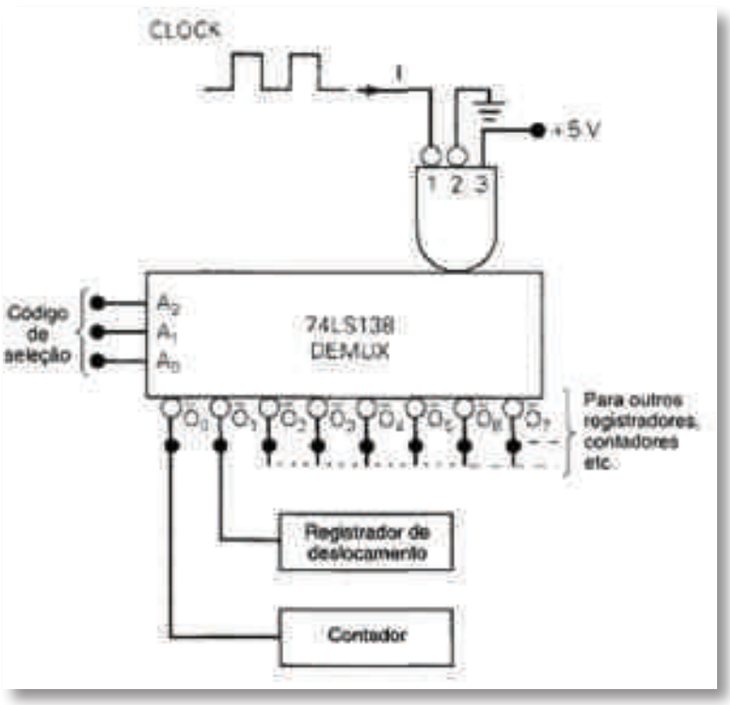


Fig. 27: Um desmultiplexer de clock transmite o sinal de clock para o destino determinado pelo código de seleção de entrada.

Sistema de controlo de segurança

Considere o caso de um sistema de controlo de segurança numa instalação industrial onde o estado aberto ou fechado de várias portas de acesso deve ser controlado.

Cada porta controla o estado de um interruptor e é necessário mostrar o estado de cada interruptor através de LEDs que estão montados num painel remoto de monitorização na sala do pessoal de segurança. Uma maneira de fazer isto, seria levar o sinal de cada porta para um LED no painel. Isto exigiria a instalação de uma grande quantidade de fios por uma distância considerável. Uma solução mais adequada, que reduziria a quantidade de fios para o painel de monitorização, utilizaria uma combinação multiplexer/desmultiplexer.

A Fig.28 mostra um sistema que pode tratar oito portas, mas a ideia básica pode ser expandida para qualquer número.

Exemplo 1:

Examine a Fig.28 cuidadosamente e descreva o seu funcionamento completo.

Solução:

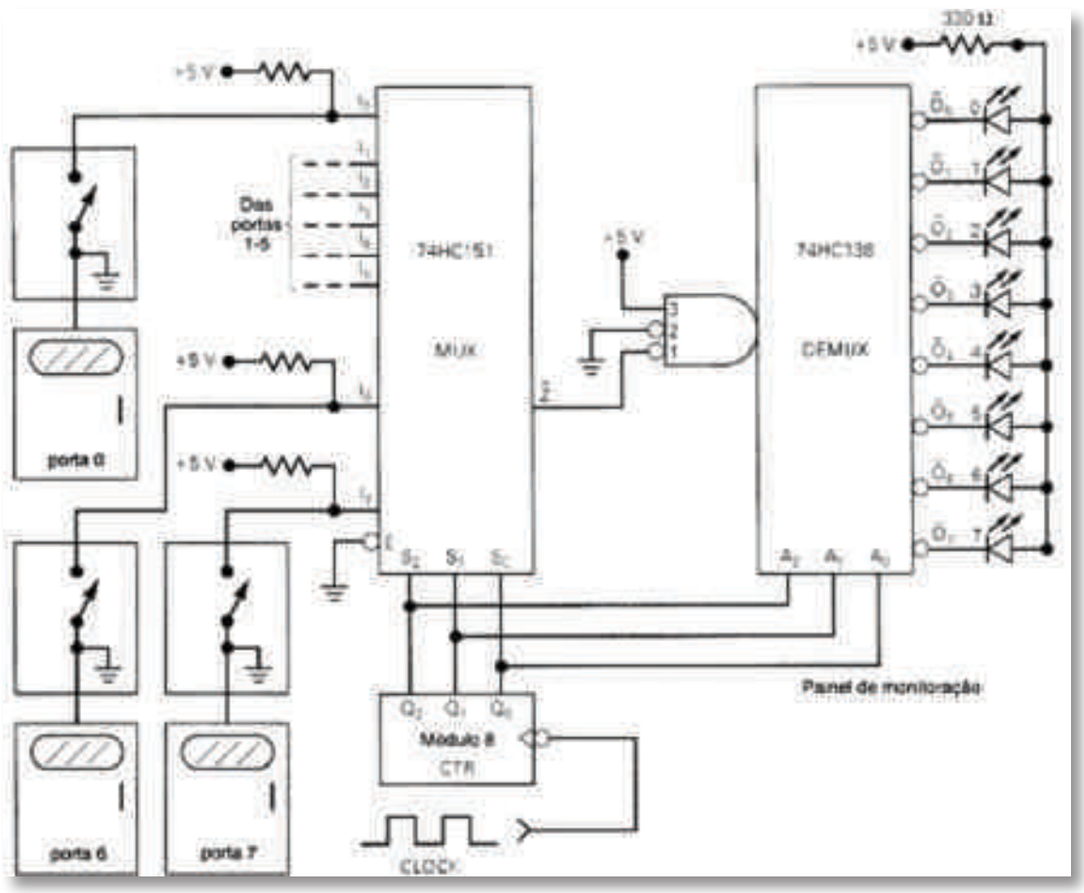


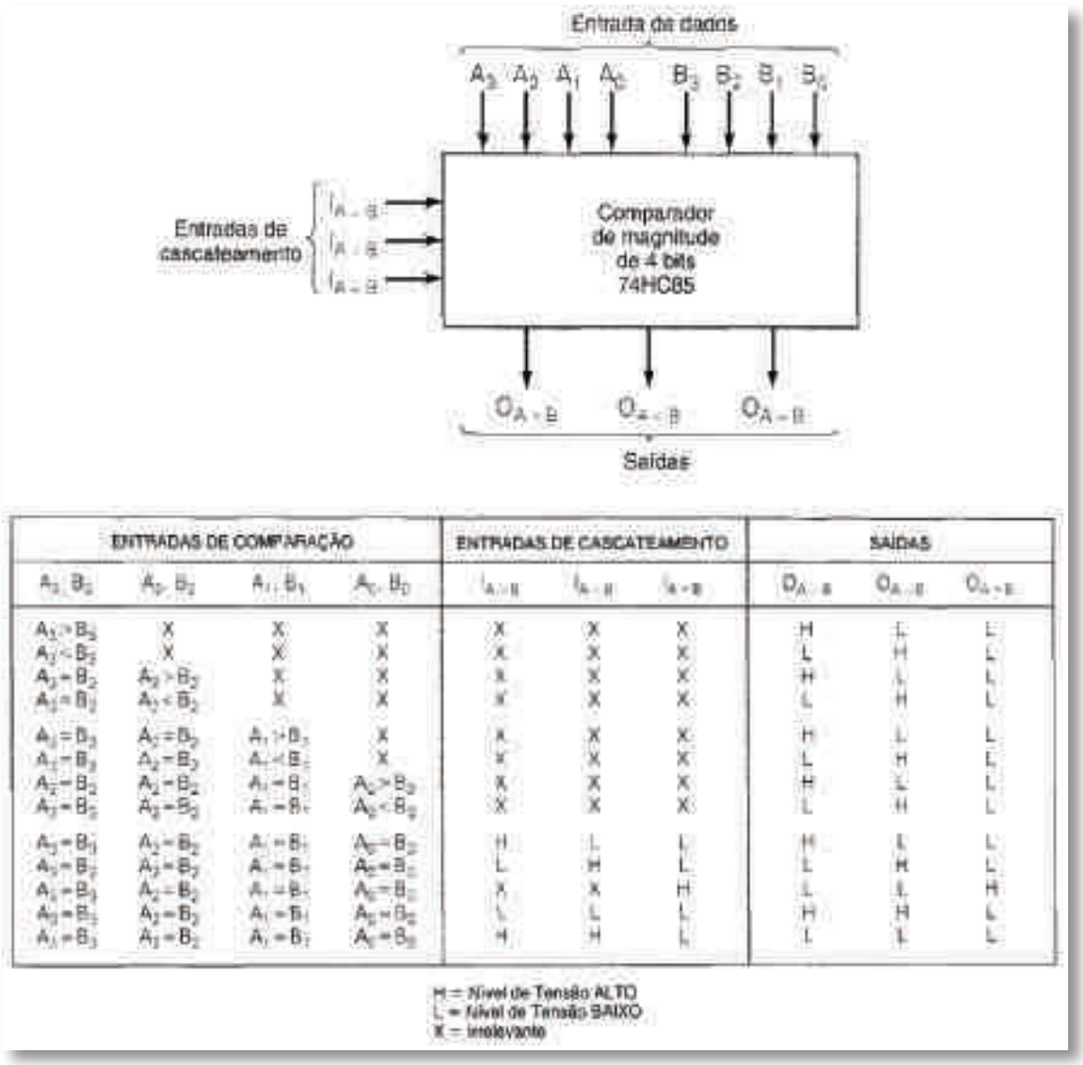
Fig. 28: Sistema de controlo de segurança

Nota:

Repare que existem apenas quatro linhas de sinal que vão dos circuitos que tratam os sensores das portas para o painel remoto de monitorização: a saída $\overline{Z}$ e as três linhas de seleção. Isto economiza quatro linhas quando comparado com a alternativa de ter uma linha por porta. A combinação MUX/DEMUX é usada para transmitir o estado de cada porta para o seu LED, um de cada vez (em série), em vez de todos ao mesmo tempo (em paralelo).

Comparadores

Outro membro útil da categoria MSI de CIs é o comparador. É um circuito lógico combinatório que compara duas quantidades binarias de entrada e gera saídas para indicar qual delas tem a maior importância. A Fig.29 mostra o símbolo lógico e a tabela de verdade para o comparador de quatro bits 74HC85, que também esta disponível como o 7485 e como o 74LS85.



ENTRADAS DE COMPARAÇÃO				ENTRADAS DE CASCATEAMENTO			SAÍDAS		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	X	X	H	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	L	H	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	H	L	L	L	L

H = Nível de Tensão ALTO
L = Nível de Tensão BAIXO
X = Irrelevante

Fig. 29: Símbolo lógico e tabela de verdade para o comparador de quatro bits 74HC85 (7485, 74LS85)

Entrada de dados

O 74HC85 compara dois números binários de quatro bits sem sinal. Um deles é $A_3A_2A_1A_0$, que é denominada de palavra A; O outro é $B_3B_2B_1B_0$, que é chamado de palavra B. O termo palavra é usado no campo dos computadores digitais para designar um grupo de bits que representam algum tipo de informação específica. Aqui, palavra A e palavra B representam quantidades numéricas.

Saídas

O 74HC85 possui três saídas ativas no nível ALTO. A saída $O_{A>B}$ estará em ALTO quando a importância da palavra A for maior do que a importância da palavra B. A saída $O_{A<B}$ estará em ALTO quando a importância da palavra A for menor do que a impotência da palavra B. A saída $O_{A=B}$ estará em ALTO quando a palavra A e a palavra B forem idênticas.

Entradas em cascata

As entradas em cascata fornecem uma maneira de expandir a operação de comparação para mais do que quatro bits, ao fazer uma cascata de dois ou mais comparadores de quatro bits.

Note que as entradas de cascata são identificadas, do mesmo modo que as saídas. Quando uma comparação de quatro bits é feita, como na Fig.30 (a), as entradas de cascata devem ser ligadas como mostrado, para produzirem as saídas corretas.

Quando dois comparadores são colocados em cascata, as saídas do comparador de mais baixa ordem são ligadas nas entradas correspondentes do comparador de mais alta ordem.

Isto é apresentado na Fig.30 (b), onde o comparador da esquerda está a comparar os quatro bits de mais baixa ordem das duas palavras de oito bits: $A_7A_6A_5A_4A_3A_2A_1A_0$ e $B_7B_6B_5B_4B_3B_2B_1B_0$. As saídas são ligadas nas entradas de cascata do comparador da direita, que está a comparar os bits de ordem mais alta. As saídas do comparador de mais alta ordem são as saídas finais que indicam o resultado da comparação de oito bits.

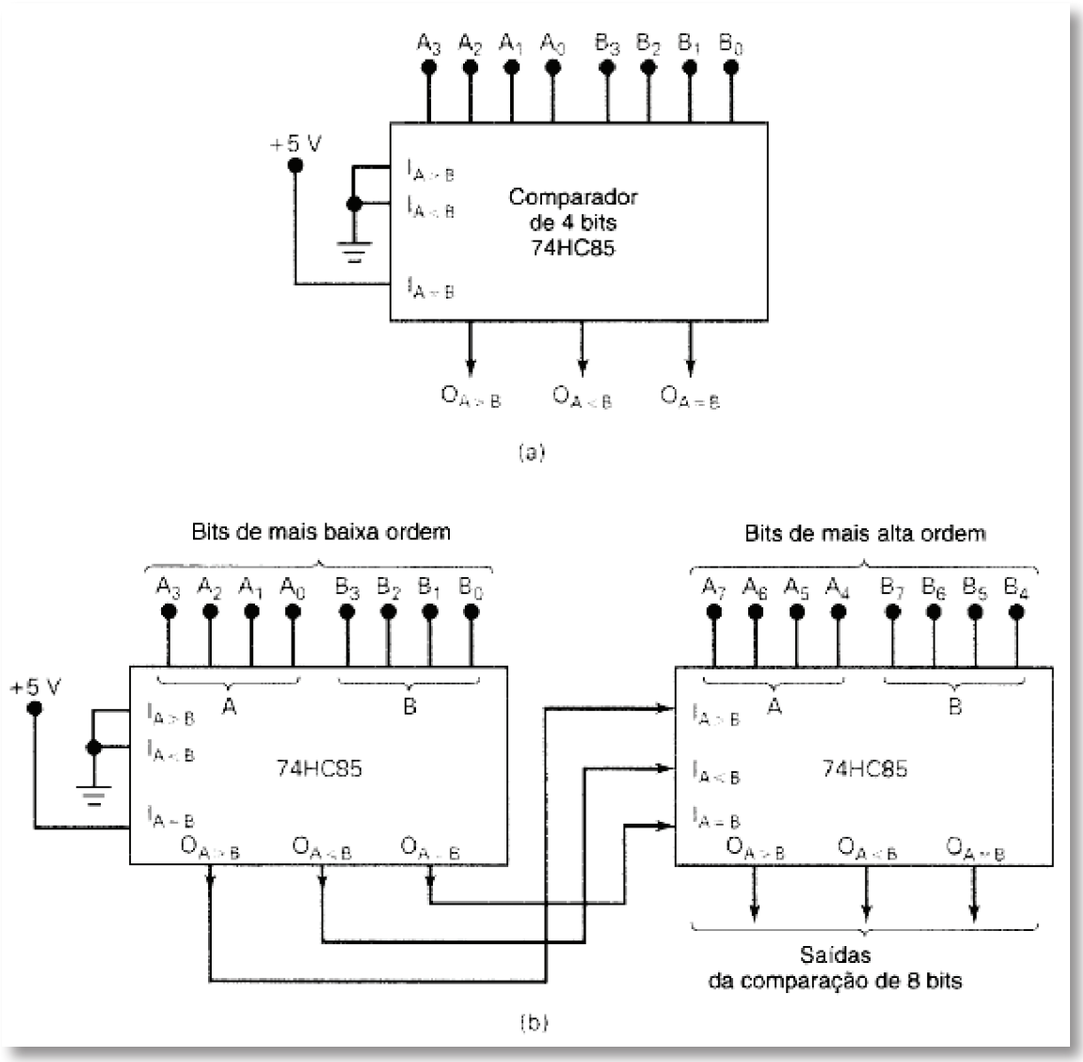


Fig. 30: (a) 74HC85 ligado como um comparador de quatro bits; (b) dois 74HC85s em cascata para realizarem uma comparação de oito bits.

Exercício 1:

Descreva a operação do circuito para comparação de oito bits da Fig.30 (b) para os seguintes casos:

(a) $A_7A_6A_5A_4A_3A_2A_1A_0 = 10101111$; $B_7B_6B_5B_4B_3B_2B_1B_0 = 10110001$

(b) $A_7A_6A_5A_4A_3A_2A_1A_0 = 10101111$; $B_7B_6B_5B_4B_3B_2B_1B_0 = 10101001$

Aplicações:

Estes comparadores também são úteis em aplicações de controlo, onde um número binário que representa uma variável física que seja controlada (por exemplo: posição,

velocidade ou temperatura) é comparado com um valor de referência. As saídas do comparador são usadas para atuar nos circuitos que levam a variável física na direção do valor de referência. O exemplo a seguir ilustrará uma aplicação.

Exercício 2:

Considere um termostato digital, no qual a temperatura ambiente de um quarto é convertida para um número digital e aplicada nas entradas A de um comparador. A temperatura desejada do quarto, informada através de um teclado, é armazenada num registador que está ligado às entradas B. Se A < B, o aquecedor deveria ser ativo para aquecer o quarto. O aquecedor deveria continuar ligado enquanto A = B e desligar-se quando A > B. Conforme o quarto fosse arrefecendo, o aquecedor permaneceria desligado enquanto A = B e ligar-se-ia novamente quando A < B.

Que circuito digital poderia ser usado para controlar um comparador com o aquecedor para realizar esta aplicação de controlo deste termostato?

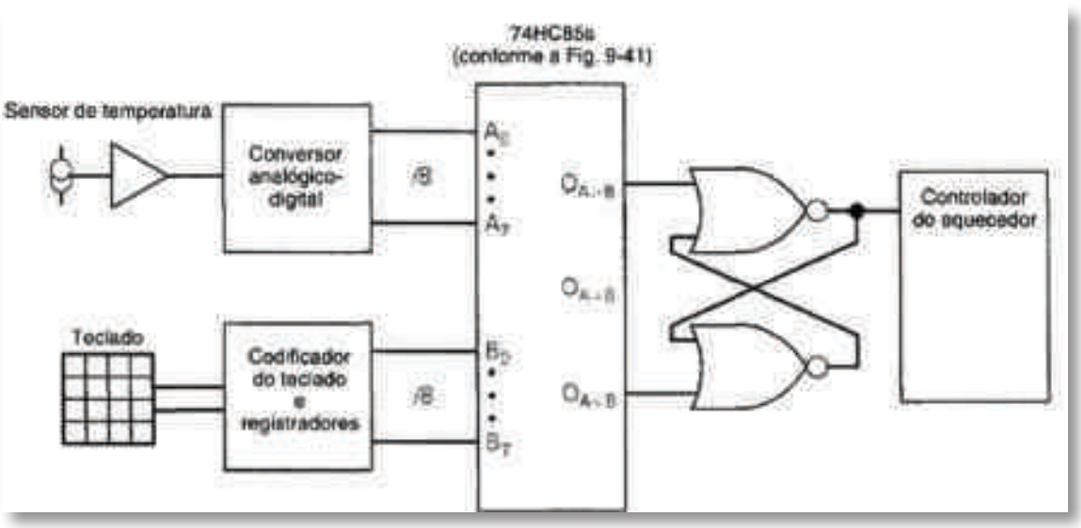


Fig. 31: Comparador usado num termostato digital

Bibliografia

PADILHA, António e outros, *Electrónica Digital*. McGrawHill. (s.d.).

PADILHA, António, *Sistemas Digitais*. McGrawHill. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Analógicos e Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).