

SISTEMAS DIGITAIS

EXERCÍCIOS RESOLVIDOS

Carlos Sêro

Guilherme Arroz

Versão 0.1
12 de Agosto de 2005

Instituto Superior Técnico
Departamento de Engenharia Electrotécnica
e de Computadores
TagusPark
Porto Salvo



Historial

12 de Agosto de 2005	v0.1	Foram acrescentadas mais exercícios resolvidos
22 de Fevereiro de 2005	v0.0	Versão original

Referências

Endereço de e-mail: cas@digitais.ist.utl.pt

Página da cadeira de Sistemas Digitais: <http://sd.tagus.ist.utl.pt>

Versão 0, revisão 1, de 12 de Agosto de 2005

Prefácio

Versão 0.1

Este texto contém alguns exercícios resolvidos, assinalados com um asterisco (*) no fim dos capítulos dos *Sistemas Digitais: Apontamentos das Aulas Teóricas*, aqui designados por *SD:AAT*.

Na lista de agradecimentos incluem-se os alunos do IST:

1. Paulo Gomes, que apontou erros na resolução dos Exercícios 1.10 e 1.20;
e
2. João Loureiro, que apontou um erro na resolução do Exercício 13.6.

Oeiras, 12 de Agosto de 2005

Carlos Sêro

Guilherme Arroz

Índice

1	SISTEMAS DE NUMERAÇÃO	1
2	CÓDIGOS	9
3	ÁLGEBRA DE BOOLE BINÁRIA	15
4	REPRESENTAÇÃO DAS FUNÇÕES	25
5	MÉTODO DE KARNAUGH	39
7	LÓGICA DE POLARIDADE	55
9	CODIFICADORES E DESCODIFICADORES	87
10	MULTIPLEXERS E DEMULTIPLEXERS	103
12	LATCHES	109
13	FLIP-FLOPS	117
14	CONTADORES	127
15	REGISTOS	149
16	CIRCUITOS SEQUENCIAIS SÍNCRONOS	159
19	MÁQUINAS DE ESTADOS	167

Capítulo 1

Sistemas de Numeração

1.1 Escrever os seguintes números em forma polinomial:

- a) $23_{(10)}$; b) $4\,087_{(10)}$; c) $39,28_{(10)}$;
d) $36_{(8)}$; e) $E5,3_{(16)}$; f) $255,6_{(7)}$;
g) $1\,023,003_{(4)}$.

Resolução: a) $23_{(10)} = 2 \times 10^1 + 3 \times 10^0 = 20_{(10)} + 3_{(10)}$.

1.1 a)

b) $4\,087_{(10)} = 4 \times 10^3 + 8 \times 10^1 + 7 \times 10^0 = 4\,000_{(10)} + 80_{(10)} + 7_{(10)}$.

1.1 b)

c) $39,28_{(10)} = 3 \times 10^1 + 9 \times 10^0 + 2 \times 10^{-1} + 8 \times 10^{-2} = 30_{(10)} + 9_{(10)} + 0,2_{(10)} + 0,08_{(10)}$.

1.1 c)

d) $36_{(8)} = 3 \times 8^1 + 6 \times 8^0 = 24_{(10)} + 6_{(10)} = 30_{(10)}$.

1.1 d)

e) $E5,3_{(16)} = 14 \times 16^1 + 5 \times 16^0 + 3 \times 16^{-1} = 224_{(10)} + 5_{(10)} + 3 \times 0,0625_{(10)} = 229,1875_{(10)}$.

1.1 e)

f) $255,6_{(7)} = 2 \times 7^2 + 5 \times 7^1 + 5 \times 7^0 + 6 \times 7^{-1} \simeq 98_{(10)} + 35_{(10)} + 5_{(10)} + 6 \times 0,14286_{(10)} = 138,14286_{(10)}$.

1.1 f)

g) $1\,023,003_{(4)} = 1 \times 4^3 + 2 \times 4^1 + 3 \times 4^0 + 3 \times 4^{-3} = 64_{(10)} + 8_{(10)} + 3_{(10)} + 3 \times 0,015625_{(10)} = 75,046875_{(10)}$.

1.1 g)

1.4 Determinar as bases b e c em:

- a) $5A_{(16)} = 132_{(b)}$;
b) $20_{(10)} = 110_{(c)}$.

Resolução: a) Como sabemos, no sistema hexadecimal temos as seguintes correspondências:

1.4 a)

$$A_{(16)} \leftrightarrow 10_{(10)}$$

$$B_{(16)} \leftrightarrow 11_{(10)}$$

Não esquecer que $10_{(10)}$ e $11_{(10)}$ são *duas sequências de dois dígitos decimais*, enquanto que $A_{(16)}$ e $B_{(16)}$ são *dois dígitos hexadecimais*.

Atendendo à definição de sistema de numeração ponderado temos que

$$5A_{(16)} = 5_{(10)} \times 16^1 + 10_{(10)} \times 16^0 = 80_{(10)} + 10_{(10)} = 90_{(10)}.$$

Por outro lado, e pela mesma definição, temos que

$$132_{(b)} = 1 \times b^2 + 3 \times b + 2$$

na base 10. Logo, podemos estabelecer a seguinte igualdade:

$$b^2 + 3b + 2 = 90_{(10)}$$

de que resulta $b_1 = -11_{(10)}$ e $b_2 = +8_{(10)}$.

Apenas consideramos a solução $b = 8_{(10)}$, embora haja sistemas de numeração com bases que são inteiros negativos.

1.4 b)

(b) Atendendo à definição de sistema de numeração ponderado, temos que

$$110_{(c)} = 1 \times c^2 + 1 \times c + 0 = c^2 + c,$$

de onde resulta que

$$c^2 + c = 20_{(10)}$$

e $c_1 = -5_{(10)}$ e $c_2 = 4_{(10)}$.

Tal como na alínea anterior, apenas consideramos a solução $c = 4_{(10)}$, embora haja sistemas de numeração com bases inteiras negativas.

1.10 O resultado da leitura do valor de uma tensão eléctrica é de 25,76 V. Representar em binário esse valor.

1.10

Resolução: Começemos por converter a parte inteira do número dado: $25_{(10)} \llcorner \llcorner 11001_{(2)}$.

Passemos agora à parte fraccionária. Pelo método das multiplicações sucessivas obtemos:

$$\begin{aligned} 0,76 \times 2 &= 1,52 \rightarrow d_{-1} = 1 \\ 0,52 \times 2 &= 1,04 \rightarrow d_{-2} = 1 \\ 0,04 \times 2 &= 0,08 \rightarrow d_{-3} = 0 \\ 0,08 \times 2 &= 0,16 \rightarrow d_{-4} = 0 \\ 0,16 \times 2 &= 0,32 \rightarrow d_{-5} = 0 \\ 0,32 \times 2 &= 0,64 \rightarrow d_{-6} = 0 \\ 0,64 \times 2 &= 1,28 \rightarrow d_{-7} = 1. \end{aligned}$$

Devemos notar que o número dado possui uma precisão de 1 parte em 100, ou seja, $1/100$. Por outro lado, este número resultou de uma leitura num voltímetro, logo existe um significado físico associado à dízima obtida (não se conseguiu, no processo de leitura, obter uma precisão superior).

Por conseguinte, na conversão do número para a base 2 não devemos “inventar” precisão. Ou seja, devemos assegurar-nos que a parte fraccionária do número binário a obter deve conter 6 bits e não mais (com 6 bits obtemos uma precisão

de 1 parte em $2^6 = 64$, ou seja, uma precisão de $1/64$, *inferior* à precisão dada de $1/100$; com 7 bits já obtínhamos uma precisão de 1 parte em $2^7 = 128$, ou seja, uma precisão de $1/128$, *superior* à precisão dada de $1/100$).

Só nos falta decidir o valor do bit com peso 2^{-6} . Se truncássemos a parte fraccionária para ficar com 6 bits, obtínhamos $0,76_{(10)} = 0,110000_{(2)}$. Porém, este resultado está incorrecto porque não levámos em consideração o bit da parte fraccionária com peso 2^{-7} , que é 1 (é esta a razão porque acima parámos no bit com peso 2^{-7}).

Claramente, precisamos de arredondar o bit com peso 2^{-6} , somando-lhe uma unidade (há casos em que a adição de uma unidade no bit menos significativo faz com que a parte fraccionária e, por vezes, também a parte inteira venha alterada; não é, contudo, o caso aqui, já que apenas o bit menos significativo passa de 0 para 1).

Obtemos, então, o seguinte resultado final:

$$25,76 \text{ V} = 11001,110001_{(2)} \text{ V}.$$

1.13 A primeira expedição a Marte provou a existência de civilizações inteligentes no planeta vermelho porque descobriu, gravada numa rocha, a equação

$$5x^2 - 50x + 125 = 0,$$

bem como as respectivas soluções, $x_1 = 5$ e $x_2 = 8$. O valor $x_1 = 5$ pareceu razoável aos elementos da expedição, mas a outra solução indicava claramente que os marcianos não utilizavam, como nós, o sistema decimal de contagem (talvez porque não possuísem 10 dedos nas mãos). Quantos dedos acha que os marcianos tinham nas mãos? Justifique.

Resolução: Obviamente, a equação do 2º grau e as respectivas soluções não estão escritas na base 10, porque senão, depois de substituirmos x por 5 e por 8 no primeiro membro da equação, devíamos obter $0 = 0$ nos dois casos; mas tal só acontece para $x_1 = 5$ (daí que a expedição, que está habituada ao sistema decimal, não tenha estranhado essa solução, e apenas ficasse intrigada com a solução $x_2 = 8$).

1.13

Trata-se, por conseguinte, de determinar a base b em que a equação e as soluções estão escritas. Acontece que uma das soluções é $x_2 = 8$. Logo, temos de ter $b \geq 9$. Nessa base temos, então:

$$\begin{aligned} 5_{(b)} &= 5_{(10)} \\ 8_{(b)} &= 8_{(10)} \\ 50_{(b)} &= (5b + 0)_{(10)} \\ 125_{(b)} &= (b^2 + 2b + 5)_{(10)} \end{aligned}$$

Logo, a equação dada, escrita em decimal, será:

$$5x^2 - 5bx + b^2 + 2b + 5 = 0.$$

Temos, então, que resolver esta equação *em decimal*, para o que vamos nela substituir x por 5 e, depois, x por 8.

1.

$$\begin{aligned} 5 \times 5^2 - 5b \times 5 + b^2 + 2b + 5 &= 0 \\ 125 - 25b + b^2 + 2b + 5 &= 0 \\ b^2 - 23b + 130 &= 0 \end{aligned}$$

de onde se conclui que $b_1 = 13$ e $b_2 = 10$.

2.

$$\begin{aligned} 5 \times 8^2 - 5b \times 8 + b^2 + 2b + 5 &= 0 \\ 320 - 40b + b^2 + 2b + 5 &= 0 \\ b^2 - 38b + 325 &= 0 \end{aligned}$$

de onde se conclui que $b_1 = 25$ e $b_2 = 13$.

Só $b = 13$ pode constituir solução do problema. Logo, os marcianos possuíam, muito provavelmente, 13 dedos (um número ímpar de dedos não devia dar muito jeito, mas enfim ...).

1.14 Como sabe do exercício anterior, a primeira expedição a Marte provou a existência de antigas civilizações inteligentes no planeta vermelho. Uma das descobertas mais importantes consistiu em perceber que os marcianos usavam um sistema de numeração com 13 símbolos, incluindo os símbolos 0 a 9, tal como nós usamos na Terra, e ainda os símbolos, \textcircled{C} , \triangleleft e \check{L} . Por outro lado, conseguiu-se provar que os marcianos conheciam as operações aritméticas de adição e de subtração. Tendo a expedição terrestre encontrado o seguinte fragmento de uma operação de adição gravada numa rocha,

$$\begin{array}{r} \check{L} \triangleleft 9 \ 3 \ 5 \\ + \quad 9 \ \check{L} \ 4 \ \textcircled{C} \\ \hline \textcircled{C} \ 9 \ 6 \ 4 \ 2 \ 3 \end{array}$$

decidiu enviar esse fragmento para a Terra para ser decifrado (os espaços em branco correspondem a símbolos que não se conseguiram ler). Refaça a adição preenchendo os fragmentos da operação que não puderam ser recuperados pela expedição terrestre, e diga quais os valores que descobriu para os símbolos \textcircled{C} , \triangleleft e \check{L} .

1.14

Resolução: Neste sistema de base 13 como este, conhecemos todos os símbolos. Em particular, sabemos que $0_{(13)} = 0_{(10)}$, $1_{(13)} = 1_{(10)}$, ..., $9_{(13)} = 9_{(10)}$. Porém, não sabemos (por enquanto) o significado dos símbolos \textcircled{C} , \triangleleft e \check{L} . Apenas sabemos que um deles corresponde ao $10_{(10)}$, outro ao $11_{(10)}$ e o terceiro ao $12_{(10)}$.

Começamos por analisar a coluna mais à direita na adição, $5 + \textcircled{C} = 3$. Dado que $3_{(13)} < 5_{(13)}$ e que $3_{(13)} < \textcircled{C}_{(13)}$, esta adição produz um transporte, gerando um resultado real igual a $13_{(13)}$. Logo, $5_{(13)} + \textcircled{C}_{(13)} = 13_{(13)}$. Mas como $13_{(13)} = 16_{(10)}$, segue-se que $5_{(10)} + \textcircled{C}_{(13)} = 16_{(10)}$ e

$$\textcircled{C}_{(13)} = 11_{(10)}.$$

Passemos à coluna seguinte, $? + 4 = 2$. Como existiu um transporte da coluna anterior, temos $1_{(13)} + ?_{(13)} + 4_{(13)} = 12_{(13)}$ (também esta coluna gera um transporte, porque $2_{(13)} < 4_{(13)}$). Como $12_{(13)} = 15_{(10)}$, segue-se que $? = 10_{(10)}$ (o que não adianta muito saber, mas enfim ...).

Passemos à terceira coluna, $3 + ? = 4$. Como existiu um transporte para ela, temos, realmente, $1_{(13)} + 3_{(13)} + ?_{(13)} = 4_{(13)}$, e $?_{(13)} = 0$, não se gerando agora transporte.

Na quarta coluna temos $9 + \check{L} = 6$. Como $6 < 9$ e $6 < \check{L}$, então temos, na realidade, $9_{(13)} + \check{L}_{(13)} = 16_{(13)}$ (é gerado um transporte), e como $16_{(13)} = 19_{(10)}$, então

$$\check{L}_{(13)} = 10_{(10)}.$$

Na quinta coluna temos $\triangleleft + 9 = 9$. Com o transporte que veio da quarta coluna temos, na realidade, $1_{(13)} + \triangleleft_{(13)} + 9_{(13)} = 19_{(13)}$ (mais um transporte que é gerado), e como $19_{(13)} = 22_{(10)}$, então

$$\triangleleft_{(13)} = 12_{(10)}.$$

Na última coluna temos $\check{L}_{(13)} + ?_{(13)} + 1_{(13)} = \odot_{(13)}$. Por outro lado, já determinámos que $\odot_{(13)} = 11_{(10)}$ e que $\check{L}_{(13)} = 10_{(10)}$. Segue-se que existia um 0 nessa coluna.

1.20 Representar os números decimais $+5$, -5 , $+54$ e -54 em notação de complemento para 2 com:

- a) 4 bits; b) 5 bits; c) 6 bits; d) 7 bits;
e) 10 bits; f) 15 bits.

Resolução: a) Temos $+5_{(10)} \langle \rangle 0101_{(C2)}$ e $-5_{(10)} \langle \rangle 1011_{(C2)}$ (notemos que 1011 foi formado por complementação para 2 de 0101).

1.20 a)

$+54_{(10)}$ e $-54_{(10)}$ não se conseguem representar com 4 bits nesta notação. Com efeito, na notação de complemento para 2 o intervalo de representação é $[-2^{n-1}, +(2^{n-1} - 1)]$, o que, para 4 bits, dá $[-8, +7]$.

b) Temos $+5_{(10)} \langle \rangle 00101_{(C2)}$ e $-5_{(10)} \langle \rangle 11011_{(C2)}$ (mais uma vez, 11011 foi formado por complementação para 2 de 00101).

1.20 b)

Comparemos a representação destes números com 5 bits e a representação com 4 bits da alínea anterior, para constatar que as representações são idênticas, excepto pelo bit de sinal que, com 5 bits, ficou duplicado.

Mais uma vez, $+54_{(10)}$ e $-54_{(10)}$ não se conseguem representar com 5 bits nesta notação porque o intervalo de representação em complemento para 2 com 5 bits é igual a $[-16, +15]$.

c) Se reproduzirmos os resultados da alínea anterior duplicando, mais uma vez, o bit de sinal dos números, obtemos o resultado correcto com 6 bits: $+5_{(10)} \langle \rangle \langle \rangle 000101_{(C2)}$ e $-5_{(10)} \langle \rangle \langle \rangle 111011_{(C2)}$.

1.20 c)

Também agora não conseguimos representar $+54_{(10)}$ e $-54_{(10)}$ com 6 bits nesta notação porque o intervalo de representação em complemento para 2 com 6 bits é $[-32, +31]$.

1.20 d)

d) Se reproduzirmos os resultados da alínea anterior duplicando, mais uma vez, o bit de sinal dos números, obtemos o resultado correcto com 7 bits: $+5_{(10)} \langle \rangle 0000101_{(C2)}$ e $-5_{(10)} \langle \rangle 1111011_{(C2)}$.

Agora já conseguimos representar $+54_{(10)}$ e $-54_{(10)}$ com 7 bits, porque o intervalo de representação em complemento para 2 com 7 bits é igual a $[-64, +63]$. Temos

$$\begin{aligned} +54_{(10)} \langle \rangle 0110110_{(C2)} \\ -54_{(10)} \langle \rangle 1001010_{(C2)} \end{aligned}$$

Notemos, mais uma vez, que 1001010 foi formado por complementação para 2 de 0110110.

1.20 e)

e) Vamos fazer como nas alíneas anteriores. Usamos as soluções para os números com um número de bits inferior a 10, e reproduzimos o bit de sinal dos números até atingir os 10 bits. Obtemos:

$$\begin{aligned} +5_{(10)} \langle \rangle 0000000101_{(C2)} \\ -5_{(10)} \langle \rangle 1111111011_{(C2)} \\ +54_{(10)} \langle \rangle 0000110110_{(C2)} \\ -54_{(10)} \langle \rangle 1111001010_{(C2)} \end{aligned}$$

1.20 f)

f) Já vimos na alínea anterior *um algoritmo muito simples para formar um número em notação de complemento para 2 com n bits, uma vez conhecida a sua representação na mesma notação com $k < n$ bits: usamos a representação do número com k bits e reproduzimos o bit de sinal até atingir os n bits.*

Para 15 bits temos, então:

$$\begin{aligned} +5_{(10)} \langle \rangle 0000000000000101_{(C2)} \\ -5_{(10)} \langle \rangle 1111111111111011_{(C2)} \\ +54_{(10)} \langle \rangle 000000000110110_{(C2)} \\ -54_{(10)} \langle \rangle 111111111001010_{(C2)} \end{aligned}$$

1.23 Provar que a subtracção

$$110010_{(C2)} - 110110_{(C2)}$$

de dois números representados em notação de complemento para 2 pode ser substituída pela soma

$$110010_{(C2)} + 001010_{(C2)}$$

do aditivo com o complemento para 2 do subtrativo.

1.23

Resolução: Vamos fazer a subtracção directamente e concluir que obtemos o mesmo resultado que na soma sugerida. Obtemos:

$$\begin{array}{rcccccccl} & 1 & 1 & 0 & 0 & 1 & 0 & (C2) & \langle \rangle & -14_{(10)} \\ - & 1 & 1 & 0 & 1 & 1 & 0 & (C2) & \langle \rangle & - -10_{(10)} \\ \hline \times & 1 & 1 & 1 & 1 & 0 & 0 & (C2) & \langle \rangle & -4_{(10)} \\ & \underbrace{}_1 & \underbrace{}_1 & \underbrace{}_1 & \underbrace{}_1 & \underbrace{}_0 & \underbrace{}_0 & & & \end{array}$$

e

$$\begin{array}{rcccccccl}
 & 1 & 1 & 0 & 0 & 1 & 0 & (C2) & <> & -14_{(10)} \\
 + & 0 & 0 & 1 & 0 & 1 & 0 & (C2) & <> & + +10_{(10)} \\
 \hline
 & 1 & 1 & 1 & 1 & 0 & 0 & (C2) & <> & -4_{(10)} \\
 & \underbrace{} & \underbrace{} & \underbrace{} & \underbrace{} & \underbrace{} & & & & \\
 & 0 & 0 & 0 & 1 & 0 & & & &
 \end{array}$$

Notemos como a subtração, tal como a soma, não leva em linha de conta um eventual transporte para além do bit mais significativo.

Capítulo 2

Códigos

2.1 Utilizar o CBN para codificar a seguinte informação decimal:

- a) $N = 31$; b) $N = 1\,674$; c) $N = 52\,674$.

Resolução: a) Para codificar a informação decimal $N = 31$ apenas precisamos de uma palavra do CBN com comprimento mínimo $n = 5$, já que com 5 bits podemos codificar $2^5 = 32$ palavras distintas. Fica:

2.1 a)

$$N_{(10)} = 31_{(10)} \langle \rangle 1\,1111_{(\text{CBN})},$$

já que $11111_{(\text{CBN})} = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31_{(10)}$.

b) Seguindo o mesmo raciocínio da resolução do exercício anterior, precisamos agora de uma palavra do CBN com comprimento mínimo $n = 11$ (porque $2^{11} = 2\,048$). Fica, então:

2.1 b)

$$N_{(10)} = 1\,674_{(10)} \langle \rangle 110\,1000\,1010_{(\text{CBN})}.$$

2.1 c)

c) Agora precisamos de uma palavra com comprimento mínimo $n = 16$ (porque $2^{16} = 64\,k = 65\,536$). Fica, então:

$$N_{(10)} = 52\,674_{(10)} \langle \rangle 1100\,1101\,1100\,0010_{(\text{CBN})}.$$

2.2 Construir um CBR com palavras de comprimento 5.

Resolução: Para construir um CBR com palavras de comprimento 5 utilizaremos o algoritmo sugerido no texto. O resultado final encontra-se indicado na Figura 2.1.

2.2

2.3 Construir um código reflectido de valência 3 e com palavras de comprimento 3. Determinar as adjacências entre palavras do código (*Nota.* Por valência de um código entende-se o número de símbolos por ele utilizados. Assim, um código binário é um código de valência 2, e um código ternário tem valência 3).

Valência de um código

0	0	0	0	0
0	0	0	0	<u>1</u>
0	0	0	1	<u>1</u>
0	0	0	<u>1</u>	0
0	0	1	1	0
0	0	1	1	<u>1</u>
0	0	1	0	<u>1</u>
0	0	<u>1</u>	0	0
0	1	<u>1</u>	0	0
0	1	1	0	<u>1</u>
0	1	1	1	<u>1</u>
0	1	1	<u>1</u>	0
0	1	0	<u>1</u>	0
0	1	0	1	<u>1</u>
0	1	0	0	<u>1</u>
0	<u>1</u>	0	0	0
1	<u>1</u>	0	0	0
1	1	0	0	<u>1</u>
1	1	0	1	<u>1</u>
1	1	0	<u>1</u>	0
1	1	1	<u>1</u>	0
1	1	1	1	<u>1</u>
1	1	1	0	<u>1</u>
1	1	<u>1</u>	0	0
1	0	<u>1</u>	0	0
1	0	1	0	<u>1</u>
1	0	1	1	<u>1</u>
1	0	1	<u>1</u>	0
1	0	0	<u>1</u>	0
1	0	0	1	<u>1</u>
1	0	0	0	<u>1</u>
1	0	0	0	0

Figura 2.1: CBR com palavras de comprimento 5

2.3*Código natural*

Resolução: Se considerarmos o conjunto ordenado de dígitos de um sistema de numeração de base b como um alfabeto, estamos a formar implicitamente um **código natural** com valência b . Por exemplo, o CBN é um código natural formado a partir do sistema de numeração binário.

Ora, como vimos no Capítulo 1, num sistema de numeração posicional de base b os dígitos de cada coluna sucedem-se pela *ordem natural*, repetindo-se um número de vezes igual ao peso da coluna. Por exemplo, na coluna com peso b^0 os dígitos repetem-se uma vez (porque $b^0 = 1$), na coluna com peso b^1 os dígitos repetem-se b vezes (porque $b^1 = b$), etc. (ver a Tabela 2.1 de *SD:AAT*). No entanto, sempre que na coluna de peso b^k se transita de $b - 1$ para 0, na coluna de peso b^{k+1} transita-se para o dígito seguinte. Ou seja, dois números consecutivos de um sistema de numeração e as palavras do código natural que correspondem a esses números diferem em mais do que um símbolo.

Para formar o CBR a partir do CBN basta substituir em cada coluna a transição de 1 para 0 pela transição de 1 para 1, prosseguindo-se em seguida pela ordem inversa até se chegar a 0, e recomeçando depois pela repetição do 0, prosseguindo-se pela ordem natural, etc. Foi este, aliás, o algoritmo que nos permitiu gerar o CBR com palavras de comprimento 5 do Exercício 2.2.

De forma semelhante, a partir de um código natural de valência b podemos formar um código reflectido com a mesma valência substituindo em cada coluna a transição de $b - 1$ para 0 pela transição de $b - 1$ para $b - 1$, prosseguindo-

-se em seguida pela ordem inversa até se chegar a 0, e recomeçando depois pela repetição do 0, prosseguindo-se pela ordem natural, etc.. Obtém-se, deste modo, a sequência

$$0, 1, 2, \dots, b-2, b-1, b-1, b-2, \dots, 2, 1, 0, 0, 1, 2, \dots$$

Evidentemente, os códigos reflectidos não são ponderados.

$$\begin{array}{c}
 J \left\{ \begin{array}{l} A \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix} \\ B \left\{ \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ C \left\{ \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 1 \\ 0 & 2 & 2 \end{bmatrix} \end{array} \right. \begin{array}{l} \leftarrow (3,3) \\ \leftarrow (3,3) \\ \leftarrow (9,3) \end{array} \\
 \\
 K \left\{ \begin{array}{l} D \left\{ \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 0 \end{bmatrix} \\ E \left\{ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \\ F \left\{ \begin{bmatrix} 1 & 0 & 2 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array} \right. \begin{array}{l} \leftarrow (3,3) \\ \leftarrow (3,3) \\ \leftarrow (9,3) \end{array} \\
 \\
 L \left\{ \begin{array}{l} G \left\{ \begin{bmatrix} 2 & 0 & 0 \\ 2 & 0 & 1 \\ 2 & 0 & 2 \end{bmatrix} \\ H \left\{ \begin{bmatrix} 2 & 1 & 2 \\ 2 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\ I \left\{ \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix} \end{array} \right. \begin{array}{l} \leftarrow (3,3) \\ \leftarrow (3,3) \\ \leftarrow (3,3) \end{array}
 \end{array}$$

Figura 2.2: Código reflectido com valência 3 e palavras de comprimento 3

Para construir o código pedido limitamo-nos a aplicar este algoritmo. Usaremos, como exemplo, os símbolos 0, 1 e 2 (Figura 2.2). Tratando-se de um código de valência $b = 3$ e com palavras de comprimento $n = 3$, ele deverá possuir $b^n = 3^3 = 27$ palavras distintas.

Notemos que, se agruparmos sucessivas linhas em matrizes de ordem (b^i, n) , com $i = 1, 2, \dots, n-1$, para cada par de matrizes sucessivas e de igual ordem são **adjacentes** (isto é, só diferem num símbolo) as linhas simétricas.

Linhas adjacentes

Por exemplo, são adjacentes as linhas 001 e 011 das matrizes A e B da Figura 2.2, tal como são adjacentes, nas mesmas matrizes, as linhas 000 e 010. De igual forma, são adjacentes a 4^a linha da matriz J (a linha 012) e a 6^a linha da matriz K (a linha 112).

Naturalmente, são igualmente adjacentes quaisquer duas palavras consecutivas do código, incluindo a primeira e a última (o símbolo 2 é adjacente ao símbolo 0).

2.4 Construir um código reflectido de valência 4 e com palavras de comprimento 2.

2.4

Resolução: Para construir um código reflectido de valência 4 e com palavras de comprimento 2 vamos utilizar o mesmo processo de construção do código do exercício anterior, porém com 4 símbolos no alfabeto, a , b , c e d (Figura 2.3).

a	a
a	b
a	c
a	d
b	d
b	c
b	b
b	a
c	a
c	b
c	c
c	d
d	d
d	c
d	b
d	a

Figura 2.3: Código reflectido com valência 4 e palavras de comprimento 2

Como vimos no exercício anterior, um código reflectido de valência b que utiliza palavras de comprimento n possui b^n palavras. Neste caso temos $b^n = 4^2 = 16$ palavras.

2.5 Codificar os dígitos decimais $0, 1, \dots, 9$ utilizando códigos binários ponderados com os pesos indicados:

- a) pesos 6 3 2 -1; b) pesos 7 3 2 -1;
 c) pesos 7 3 1 -2; d) pesos 5 4 -2 -1;
 e) pesos 8 7 -4 -2.

2.5

Resolução: As codificações pretendidas encontram-se indicadas na Tabela 2.1.

Notemos que, em geral, é possível encontrar mais do que uma codificação para cada dígito. Quando isso acontece, procuramos codificações que confirmam aos códigos determinadas propriedades. Por exemplo, os códigos 632-1, 731-2 e 54-2-1 distinguem os dígitos iguais ou superiores a 5 dos que são inferiores a 5. Por outro lado, os códigos 731-2 e 87-4-2 são **auto-complementares**, isto é, em que cada dígito decimal, D , e o seu complemento para 9 (obtido pelo resultado da subtração $9 - D$) têm representações binárias complementares, com os zeros trocados por uns.

*Códigos
autocomplementares*

Tabela 2.1: Codificação dos dígitos 0 a 9 utilizando os códigos ponderados do Exercício 2.5

Dígito	Código				
	6 3 2 -1	7 3 2 -1	7 3 1 -2	5 4 -2 -1	8 7 -4 -2
0	0000	0000	0000	0000	0000
1	0011	0011	0010	0111	0111
2	0010	0010	0111	0110	1011
3	0100	0100	0100	0101	0110
4	0111	0111	0110	0100	1010
5	1001	0110	1001	1000	0101
6	1000	1001	1011	1111	1001
7	1011	1000	1000	1110	0100
8	1010	1011	1101	1101	1000
9	1100	1010	1111	1100	1111

Capítulo 3

Álgebra de Boole Binária

3.1 Mostre que a função Equivalência é comutativa e associativa.

Resolução: Uma vez que estamos a trabalhar no contexto da álgebra de Boole binária, é exequível fazer a demonstração pedida por **indução completa**, isto é, mostrando todos os casos possíveis. É o que fazemos na tabela de verdade da função Equivalência (Tabela 3.1).

3.1

Indução completa

Tabela 3.1: Tabela de verdade onde se mostra que a função Equivalência é comutativa e associativa

A	B	$A \odot B$	$B \odot A$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

A	B	C	$A \odot B$	$(A \odot B) \odot C$	$B \odot C$	$A \odot (B \odot C)$
0	0	0	1	0	1	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	0	0	1	0
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Como podemos constatar, $A \odot B = B \odot A$ (pelo que a função é comutativa) e $(A \odot B) \odot C = A \odot (B \odot C)$ (e a função é associativa).

Naturalmente, podemos também demonstrar algebricamente a comutatividade e a associatividade da função Equivalência, atendendo à sua definição. É o que

faremos de seguida.

$$\begin{aligned} A \odot B &\stackrel{\text{def}}{=} AB + \overline{A}\overline{B} \\ B \odot A &\stackrel{\text{def}}{=} BA + \overline{B}\overline{A} \\ &= A \odot B, \end{aligned}$$

o que demonstra a comutatividade.

Por outro lado,

$$\begin{aligned} A \odot (B \odot C) &\stackrel{\text{def}}{=} A(B \odot C) + \overline{A}(\overline{B \odot C}) \\ &\stackrel{\text{def}}{=} A(B \odot C) + \overline{A}(B \oplus C) \\ &\stackrel{\text{def}}{=} A(BC + \overline{B}\overline{C}) + \overline{A}(B\overline{C} + \overline{B}C) \\ &= ABC + A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C \end{aligned}$$

e

$$\begin{aligned} (A \odot B) \odot C &\stackrel{\text{def}}{=} (A \odot B)C + (\overline{A \odot B})\overline{C} \\ &\stackrel{\text{def}}{=} (A \odot B)C + (\overline{A \oplus B})\overline{C} \\ &\stackrel{\text{def}}{=} (AB + \overline{A}\overline{B})C + (\overline{A\overline{B} + \overline{A}B})\overline{C} \\ &= ABC + \overline{A}\overline{B}C + A\overline{B}\overline{C} + \overline{A}B\overline{C} \\ &= A \odot (B \odot C), \end{aligned}$$

o que demonstra a associatividade.

3.2 Prove a seguinte lei de De Morgan: $\overline{x + y} = \overline{x}\overline{y}$.

3.2

Resolução: Da mesma forma que no exercício anterior, podemos demonstrar a lei de De Morgan por indução completa (Tabela 3.2) ou algebricamente.

Tabela 3.2: Tabela de verdade onde se demonstra a segunda lei de De Morgan

A	B	A + B	$\overline{A + B}$	\overline{A}	\overline{B}	$\overline{A}\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Para a demonstração algébrica vamos ter de nos socorrer dos axiomas e de outros teoremas da álgebra de Boole binária.

Para tanto, consideremos

$$\begin{aligned} xy(\overline{x} + \overline{y}) &= xy\overline{x} + xy\overline{y} && [\text{A2a}] \\ &= 0 + 0 && [\text{A4a, T2a}] \\ &= 0. && [\text{A3b}] \end{aligned}$$

Por outro lado, consideremos

$$x y + (\bar{x} + \bar{y}) = (\bar{x} + \bar{y}) + x y \quad [\text{A1b}]$$

$$= [(\bar{x} + \bar{y}) + x] \cdot [(\bar{x} + \bar{y}) + y] \quad [\text{A2b}]$$

$$= (1 + \bar{y}) \cdot (1 + \bar{x}) \quad [\text{T3a, A4b}]$$

$$= 1 \cdot 1 \quad [\text{T2b}]$$

$$= 1. \quad [\text{A3a}]$$

Sendo assim, $\bar{x} + \bar{y}$ e $x y$ satisfazem os axiomas A4a e A4b, pelo que $\bar{x} + \bar{y}$ deverá ser o (único) complemento de $x y$, e vice-versa. É então possível escrever

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

e, por dualidade,

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

3.3 Escreva tabelas de verdade adequadas para as seguintes funções booleanas simples:

- a) $f(A, B, C) = A(\bar{B} + \bar{C})(B + C)$;
- b) $f(A, B, C, D) = A(B + \bar{C}(\bar{B} + D))$;
- c) $f(A, B, C) = \bar{A}\bar{C} + \bar{B}\bar{C}$.

Resolução: a) $f(A, B, C) = A(\bar{B} + \bar{C})(B + C) \rightarrow vd.$ a Tabela 3.3.

3.3 a)

Tabela 3.3: Tabela de verdade da função $f(A, B, C) = A(\bar{B} + \bar{C})(B + C)$. A enumeração das linhas pelos equivalentes decimais dos números correspondentes às quantidades booleanas gerais $(A, B, C) = (0, 0, 0) \dots (1, 1, 1)$ não faz parte da tabela

Linha #	A	B	C	A	$\bar{B} + \bar{C}$	$B + C$	$f = A(\bar{B} + \bar{C})(B + C)$
0	0	0	0	0	1	0	0
1	0	0	1	0	1	1	0
2	0	1	0	0	1	1	0
3	0	1	1	0	0	1	0
4	1	0	0	1	1	0	0
5	1	0	1	1	1	1	1
6	1	1	0	1	1	1	1
7	1	1	1	1	0	1	0

Notemos, na coluna $\bar{B} + \bar{C}$, que é suficiente que $B = 0$ ou que $C = 0$ para que $\bar{B} + \bar{C} = 1$. Por outro lado, na coluna $B + C$ é suficiente que $B = 1$ ou que $C = 1$ para que $B + C = 1$. Finalmente, na coluna da função basta que um dos três factores, A , $\bar{B} + \bar{C}$ ou $B + C$ tenha o valor 0 para que $f = 0$.

b) $f(A, B, C, D) = A(B + \bar{C}(\bar{B} + D)) \rightarrow vd.$ a Tabela 3.4.

3.3 b)

c) $f(A, B, C) = \bar{A}\bar{C} + \bar{B}\bar{C} \rightarrow vd.$ a Tabela 3.5.

3.3 c)

Tabela 3.4: Tabela de verdade da função $f(A, B, C, D) = A(B + \overline{C}(\overline{B} + D))$

A	B	C	D	$\overline{B} + D$	$\overline{C}(\overline{B} + D)$	$B + \overline{C}(\overline{B} + D)$	$f = A(B + \overline{C}(\overline{B} + D))$
0	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	0
0	1	1	0	0	0	1	0
0	1	1	1	1	0	1	0
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	0	1	1
1	1	0	1	1	1	1	1
1	1	1	0	0	0	1	1
1	1	1	1	1	0	1	1

Tabela 3.5: Tabela de verdade da função $f(A, B, C) = \overline{A}\overline{C} + \overline{B}\overline{C}$. A enumeração das linhas pelos equivalentes decimais dos números correspondentes às quantidades booleanas gerais $(A, B, C) = (0, 0, 0) \dots (1, 1, 1)$ não faz parte da tabela

Linha #	A	B	C	AC	$\overline{A}\overline{C}$	\overline{B}	\overline{C}	$\overline{B}\overline{C}$	$f = \overline{A}\overline{C} + \overline{B}\overline{C}$
0	0	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	0	0	1
2	0	1	0	0	1	0	1	0	1
3	0	1	1	0	1	0	0	0	1
4	1	0	0	0	1	1	1	1	1
5	1	0	1	1	0	1	0	0	0
6	1	1	0	0	1	0	1	0	1
7	1	1	1	1	0	0	0	0	0

3.4 Prove, examinando todos os casos possíveis (demonstração por **indução completa**), que os seguintes teoremas são válidos (os pares de teoremas são duais):

- a) $\overline{\overline{A}} = A$;
 b) $A + 0 = A$ $A \cdot 1 = A$;
 c) $A + 1 = 1$ $A \cdot 0 = 0$;
 d) $A + A = A$ $A A = A$;
 e) $A + \overline{A} = 1$ $A \overline{A} = 0$.

Resolução: Como se pretendem provar as igualdades por indução completa, teremos que as provar para todas as quantidades booleanas gerais possíveis (que, no caso, se resumem a $A = 0$ e $A = 1$). A forma mais simples de o fazer é recorrer a tabelas de verdade. Por outro lado, para as igualdades duais basta provar uma delas.

3.4

→ *vd.* a Tabela 3.6.

Tabela 3.6: Várias tabelas de verdade que mostram, por indução completa, as igualdades do Exercício 3.4

A	\overline{A}	$\overline{\overline{A}}$	$A + 0$	$A + 1$	$A + A$	$A + \overline{A}$
0	1	0	0	1	0	1
1	0	1	1	1	1	1

3.5 Prove, por indução completa, que $AB + \overline{A}C + BC = AB + \overline{A}C$ (**teorema do consenso**).

Resolução: Tal como no exercício anterior, vamos recorrer a tabelas de verdade para provar, por indução completa, o teorema do consenso. → *vd.* a Tabela 3.7.

3.5

Tabela 3.7: Teorema do consenso provado por indução completa

A	B	C	AB	$\overline{A}C$	BC	$AB + \overline{A}C + BC$	$AB + \overline{A}C$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	1	0	1	1	1

3.6 Através de manipulações algébricas, e utilizando os axiomas e os teoremas da álgebra de Boole binária que conhece, verifique as seguintes igualdades:

- a) $(A + \overline{B} + AB)(A + \overline{B})\overline{A}B = 0$;
 b) $\overline{A}B(\overline{D} + D\overline{C}) + (A + D\overline{A}C)B = B$;
 c) $[(\overline{B} + C)A] + (\overline{C}\overline{D}) = CD$.

Resolução: a) Temos:

3.6 a)

$$\begin{aligned}
(A + \overline{B} + AB)(A + \overline{B})\overline{A}B &= 0 \\
[A(1 + B) + \overline{B}](A + \overline{B})\overline{A}B &= 0 \\
(A + \overline{B})(A + \overline{B})\overline{A}B &= 0 \\
(A + \overline{B})\overline{A}B &= 0 \\
A\overline{A}B + \overline{A}\overline{B}B &= 0 \\
0 &= 0
\end{aligned}$$

3.6 b)

b) Temos:

$$\begin{aligned}
\overline{A}B(\overline{D} + D\overline{C}) + (A + D\overline{A}C)B &= B \\
\overline{A}B(\overline{D} + \overline{C}) + (A + CD)B &= B \\
\overline{A}B\overline{D} + \overline{A}B\overline{C} + AB + BCD &= B \\
B(\overline{A}\overline{D} + \overline{A}\overline{C} + A + CD) &= B \\
B(A + \overline{C} + \overline{D} + CD) &= B \\
B(A + \overline{C}\overline{D} + CD) &= B \\
B(A + 1) &= B \\
B \cdot 1 &= B \\
B &= B
\end{aligned}$$

3.6 c)

c) Temos:

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

3.9 Simplifique algebricamente

- a) $ABCD + ABC\overline{D} + \overline{A}BC\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D}$;
b) $\overline{X} + XY\overline{Z} + \overline{Y}$;
c) $XY + WXY\overline{Z} + \overline{X}Y$;
d) $\overline{X}\overline{Y}Z + YZ + XZ$.

3.9 a)

Resolução: a) Temos:

$$\begin{aligned}
& ABCD + ABC\overline{D} + \overline{A}BC\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} \\
&= ABC(D + \overline{D}) + \overline{A}B\overline{D}(C + \overline{C}) + \overline{A}\overline{B}\overline{D}(C + \overline{C}) \\
&= ABC \cdot 1 + \overline{A}B\overline{D} \cdot 1 + \overline{A}\overline{B}\overline{D} \cdot 1 \\
&= ABC + \overline{A}B\overline{D} + \overline{A}\overline{B}\overline{D} \\
&= ABC + \overline{A}\overline{D}(B + \overline{B}) \\
&= ABC + \overline{A}\overline{D} \cdot 1 \\
&= ABC + \overline{A}\overline{D}
\end{aligned}$$

b) Temos:

3.9 b)

$$\begin{aligned}
&\overline{X} + XY\overline{Z} + \overline{Y} \\
&= (\overline{X} + XY\overline{Z}) + \overline{Y} \\
&= (\overline{X} + Y\overline{Z}) + \overline{Y} \\
&= \overline{X} + (Y\overline{Z} + \overline{Y}) \\
&= \overline{X} + \overline{Z} + \overline{Y}
\end{aligned}$$

c) Temos:

3.9 c)

$$\begin{aligned}
&XY + WXY\overline{Z} + \overline{X}Y \\
&= (XY + XYW\overline{Z}) + \overline{X}Y \\
&= XY(1 + W\overline{Z}) + \overline{X}Y \\
&= XY \cdot 1 + \overline{X}Y \\
&= XY + \overline{X}Y \\
&= Y(X + \overline{X}) \\
&= Y \cdot 1 \\
&= Y
\end{aligned}$$

3.13 Um técnico de laboratório químico possui quatro produtos químicos, A , B , C e D , que devem ser guardados em dois depósitos. Por conveniência, é necessário mover um ou mais produtos de um depósito para o outro de tempos a tempos. A natureza dos produtos é tal que é perigoso guardar B e C juntos, a não ser que A esteja no mesmo depósito. Também é perigoso guardar C e D juntos se A não estiver no depósito. Escreva uma expressão para uma função, Z , tal que $Z = 1$ sempre que exista uma combinação perigosa em qualquer dos depósitos.

Resolução: Há duas situações perigosas.

3.13

Situação 1: produto B + produto C + ausência de produto A : $(BC\overline{A})$.

Situação 2: produto C + produto D + ausência de produto A : $(CD\overline{A})$.

Logo $Z(A, B, C, D) = BC\overline{A} + CD\overline{A} = C\overline{A}(B + D)$.

3.14 Existem três interruptores de parede, a , b e c . $A = 1$ representa a condição

“interruptor a ligado”, e $A = 0$ representa a condição “interruptor a desligado”. De modo similar, as variáveis B e C estão associadas às posições dos interruptores b e c , respectivamente. Escreva uma expressão booleana para uma função Z , de modo que a alteração do estado de um interruptor, independentemente dos outros, vá provocar a mudança do valor da função.

3.14

Resolução: Suponhamos que $Z = 0$ quando os três interruptores estão desligados (naturalmente, podíamos ter feito a suposição oposta). O enunciado afirma que o valor de Z se altera sempre que um interruptor mude de estado (de desligado para ligado, ou de ligado para desligado).

Vamos, então, dispor as combinações possíveis dos estados dos interruptores de modo que, de uma combinação para a outra, apenas mude o estado de um interruptor. No fundo, vamos escrever as palavras de um CBR — código binário reflectido — como mostra a Tabela 3.8.

Tabela 3.8: Tabela de verdade da função Z que, ao contrário do habitual, ordena as linhas segundo as palavras de um CBR e não do CBN

Linha #	A	B	C	Z	Eixos
1	0	0	0	0	3
2	0	0	1	1	2
3	0	1	1	0	3
4	0	1	0	1	1
5	1	1	0	0	3
6	1	1	1	1	2
7	1	0	1	0	3
8	1	0	0	1	

Arbitrou-se que
 $Z = 0$ quando
 $A = B = C = 0$

Linhas adjacentes

Consideremos, por exemplo, a linha 1, quando $A = B = C = 0$, e notemos que existem 3 linhas que apenas diferem dela numa variável (diz-se que existem 3 **linhas adjacentes** à linha 1). Assim, a linha 2 difere da linha 1 na variável C , a linha 4 difere da linha 1 na variável B e, finalmente, a linha 8 difere da linha 1 na variável A . Para qualquer uma destas 3 linhas o valor de Z deve mudar: como na linha 1 se tem $A = B = C = 0$ e se arbitrou que, nestas condições, devia vir $Z = 0$, segue-se que nas linhas 2, 4 e 8 se deve ter $Z = 1$.

Naturalmente, o raciocínio anteriormente feito em relação à linha 1 pode ser estendido a qualquer outra (cada linha de um CBR com palavras de comprimento n possui n linhas adjacentes que, por definição, apenas diferem dela numa variável).

Notemos que a disposição pouco habitual da tabela de verdade da Tabela 3.8, que ordena as linhas segundo um CBR em vez do habitual CBN, tem a vantagem de fazer salientar de forma fácil as adjacências entre as linhas. Com efeito, dada uma linha i , qualquer, a linha que é simétrica de i em relação ao eixo 1 difere desta na variável A , a linha que é simétrica de i em relação ao eixo 2 mais próximo difere desta na variável B , e a linha que é simétrica de i em relação ao eixo 3 mais próximo difere desta na variável C .

Podemos, finalmente, estabelecer agora a expressão da função Z :

$$\begin{aligned}
 Z(A, B, C) &= \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C} \\
 &= \overline{A}\overline{B}C + (\overline{A}B\overline{C} + A\overline{B}\overline{C}) + ABC \\
 &= \overline{A}\overline{B}C + (\overline{A}B + A\overline{B})\overline{C} + ABC \\
 &= \overline{A}\overline{B}C + (A \oplus B)\overline{C} + ABC \\
 &= (\overline{A}\overline{B} + AB)C + (A \oplus B)\overline{C} \\
 &= (\overline{A \oplus B})C + (A \oplus B)\overline{C} \\
 &= (A \oplus B) \oplus C \\
 &= A \oplus B \oplus C
 \end{aligned}$$

Capítulo 4

Representação das Funções

4.3 Desenhar as tabelas de verdade das seguintes funções booleanas simples:

- a) $F_1(A, B, C) = \overline{A}BC + \overline{A}B\overline{C} + AC$;
- b) $F_2(A, B, C) = A(B + \overline{C})(\overline{B} + C)$;
- c) $F_3(A, B, C, D) = A[\overline{B} + \overline{C}(\overline{B} + D)]$;
- d) $F_4(A, B, C) = \overline{A\overline{C} + BC}$;
- e) $F_5(A, B, C) = A(\overline{B\overline{C} + BC})$.

e identificar, para cada uma delas, as correspondentes formas canónicas.

Resolução:

- a) $F_1(A, B, C) = \overline{A}BC + \overline{A}B\overline{C} + AC \rightarrow \text{vd. a Tabela 4.1.}$

4.3 a)

Tabela 4.1: Tabela de verdade da função booleana simples $F_1(A, B, C) = \overline{A}BC + \overline{A}B\overline{C} + AC$. A enumeração das linhas pelos equivalentes decimais dos números correspondentes às quantidades booleanas gerais $(A, B, C) = (0, 0, 0) \dots (1, 1, 1)$ não faz parte da tabela

Linha #	A	B	C	$\overline{A}BC$	$\overline{A}B\overline{C}$	AC	$F_1 = \overline{A}BC + \overline{A}B\overline{C} + AC$
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	1	0	0	1	0	1
3	0	1	1	1	0	0	1
4	1	0	0	0	0	0	0
5	1	0	1	0	0	1	1
6	1	1	0	0	0	0	0
7	1	1	1	0	0	1	1

Temos que $F_1(A, B, C) = \sum m(2, 3, 5, 7) = \prod M(0, 1, 4, 6)$.

Note-se que podemos obter a tabela de verdade de uma maneira mais expedita, se atendermos a que: (i) a expressão da função vem dada por uma soma lógica

de produtos lógicos (mais simplesmente, uma soma de produtos); e (ii) a soma vem a 1 desde que pelo menos uma parcela da soma lógica venha a 1. Então, $F_1 = \overline{A}BC + \overline{A}B\overline{C} + AC$ vem a 1 nas seguintes condições:

1. $\overline{A}BC = 1$, para o que basta que $A = 0$ e $B = 1$ e $C = 1$ (linha 3); ou
2. $\overline{A}B\overline{C} = 1$, para o que basta que $A = 0$ e $B = 1$ e $C = 0$ (linha 2); ou
3. $AC = 1$, para o que basta que $A = 1$ e $C = 1$ (linhas 5 e 7).

Ou seja, $F_1 = 1$ nas linhas 2, 3, 5 e 7, pelo que $F_1 = \sum m(2, 3, 5, 7)$, como tínhamos anteriormente verificado.

4.3 b)

b) $F_2(A, B, C) = A(B + \overline{C})(\overline{B} + C) \rightarrow vd.$ a Tabela 4.2.

Tabela 4.2: Tabela de verdade da função $F_2 = A(B + \overline{C})(\overline{B} + C)$. A enumeração das linhas pelos equivalentes decimais dos números correspondentes às quantidades booleanas gerais $(A, B, C) = (0, 0, 0) \dots (1, 1, 1)$ não faz parte da tabela

Linha #	A	B	C	$B + \overline{C}$	$\overline{B} + C$	$F_2 = A(B + \overline{C})(\overline{B} + C)$
0	0	0	0	1	1	0
1	0	0	1	0	1	0
2	0	1	0	1	0	0
3	0	1	1	1	1	0
4	1	0	0	1	1	1
5	1	0	1	0	1	0
6	1	1	0	1	0	0
7	1	1	1	1	1	1

Temos que $F_2(A, B, C) = \sum m(4, 7) = \prod M(0 - 3, 5, 6)$.

Para obtermos a tabela de verdade de forma expedita, vamos agora atender a que a expressão dada para a função, $F_2 = A(B + \overline{C})(\overline{B} + C)$, vem num produto lógico de somas lógicas (mais simplesmente, num produto de somas), em que o primeiro produto vem simplificado, porque reduzido ao literal A . Ora, para que o produto dê 0 basta que pelo menos um dos factores do produto venha a 0.

Então, F_2 vem a 0 nas seguintes condições:

1. $A = 0$ (linhas 0 a 3); ou
2. $B + \overline{C} = 0$, para o que deve ser $B = 0$ e $C = 1$ (linhas 1 e 5); ou
3. $\overline{B} + C = 0$, para o que deve ser $B = 1$ e $C = 0$ (linhas 2 e 6).

Ou seja, $F_2 = 0$ nas linhas 0 a 3, 5 e 6, pelo que $F_2 = \prod M(0 - 3, 5, 6)$, como tínhamos anteriormente constatado.

4.3 c)

c) $F_3(A, B, C, D) = A[\overline{B} + \overline{C}(\overline{B} + D)] \rightarrow vd.$ a Tabela 4.3.

Temos que $F_3(A, B, C, D) = \sum m(8 - 11, 13) = \prod M(0 - 7, 12, 14, 15)$.

Tabela 4.3: Tabela de verdade da função booleana simples $F_3 = A[\overline{B} + \overline{C}(\overline{B} + D)]$

A	B	C	D	$\overline{B} + D$	$\overline{C}(\overline{B} + D)$	$\overline{B} + \overline{C}(\overline{B} + D)$	$F_3 = A[\overline{B} + \overline{C}(\overline{B} + D)]$
0	0	0	0	1	1	1	0
0	0	0	1	1	1	1	0
0	0	1	0	1	0	1	0
0	0	1	1	1	0	1	0
0	1	0	0	0	0	0	0
0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	0	1	1
1	0	1	1	1	0	1	1
1	1	0	0	0	0	0	0
1	1	0	1	1	1	1	1
1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0

De forma expedita, vamos transformar a expressão de F_3 numa soma de produtos. Fica:

$$\begin{aligned} F_3 &= A[\overline{B} + \overline{C}(\overline{B} + D)] \\ &= A\overline{B} + A\overline{B}\overline{C} + A\overline{C}D. \end{aligned}$$

Ou seja, F_3 vem a 1 nas seguintes condições:

1. $A = 1$ e $B = 0$ (linhas 8 a 11); ou
2. $A = 1$ e $B = 0$ e $C = 0$ (linhas 8 e 9); ou
3. $A = 1$ e $C = 0$ e $D = 1$ (linhas 9 e 13).

Segue-se que $F_3 = 1$ nas linhas 8 a 11, e 13, pelo que $F_3 = \sum m(8 - 11, 13)$, como tínhamos anteriormente verificado.

d) $F_4(A, B, C) = \overline{A}\overline{C} + BC \rightarrow$ *vd.* a Tabela 4.4.

4.3 d)

Temos que $F_4(A, B, C) = \sum m(0 - 4, 6, 7) = M_5$.

Olhando para o resultado obtido, constatamos que $F_4(A, B, C) = \overline{A}\overline{C} + BC$ vale 1 por toda a parte, excepto para a quantidade booleana geral $(A, B, C) = (1, 0, 1)$, em que vale 0 (ou seja, apenas contém o maxtermo M_5). Segue-se que podemos deduzir a seguinte expressão alternativa para a função:

$$F_4(A, B, C) = M_5 = \overline{A\overline{B}C} = \overline{A} + B + \overline{C}.$$

Tabela 4.4: Tabela de verdade da função booleana simples $F_4(A, B, C) = \overline{A}C + BC$

A	B	C	AC	$\overline{A}C$	BC	$F_4(A, B, C) = \overline{A}C + BC$
0	0	0	0	1	0	1
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	0	1	1	1
1	0	0	0	1	0	1
1	0	1	1	0	0	0
1	1	0	0	1	0	1
1	1	1	1	0	1	1

Notemos que a tabela de verdade correspondente à expressão booleana $A\overline{B}C$ tem zeros por toda a parte excepto para a quantidade booleana geral $(A, B, C) = (1, 0, 1)$, em que tem um 1.

Vamos demonstrar algebricamente que as duas expressões anteriores para F_4 são equivalentes. Temos:

$$\begin{aligned}
 F_4(A, B, C) &= \overline{A}C + BC \\
 &= \overline{A} + \overline{C} + BC \quad (\text{uma das leis de De Morgan}) \\
 &= \overline{A} + B + \overline{C} \quad (\text{teorema da redundância}) \\
 &= \overline{A\overline{B}C}. \quad (\text{a mesma lei de De Morgan})
 \end{aligned}$$

4.3 e)

e) $F_5(A, B, C) = \overline{A(\overline{B}\overline{C} + BC)} \rightarrow \text{vd. a Tabela 4.5.}$

Tabela 4.5: Tabela de verdade da função booleana simples $F_5(A, B, C) = \overline{A(\overline{B}\overline{C} + BC)}$

A	B	C	BC	$\overline{B}\overline{C}$	$A(\overline{B}\overline{C} + BC)$	$F_5(A, B, C) = \overline{A(\overline{B}\overline{C} + BC)}$
0	0	0	0	1	0	1
0	0	1	0	0	0	1
0	1	0	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	1	0	0	0	1
1	1	0	0	0	0	1
1	1	1	1	0	1	0

Temos que $F_5(A, B, C) = \sum m(0-3, 5, 6) = \prod M(4, 7)$.

4.5 Considere as seguintes funções booleanas simples:

- $f(A, B, C) = (A \oplus B)C + \overline{A}(B \oplus C)$;
- $f(A, B, C, D) = A + \overline{A}BC + C\overline{D} \oplus (C\overline{D} + \overline{C}D) + \overline{C}D \oplus (\overline{C}D + C\overline{D})$.

Escreva-as na forma normal conjuntiva (produto de somas) mais simples que conseguir.

Resolução: a) Como $A \oplus B = (A + B)(\overline{A} + \overline{B})$, temos que

4.5 a)

$$\begin{aligned} f(A, B, C) &= (A \oplus B)C + \overline{A}(B \oplus C) \\ &= (A + B)(\overline{A} + \overline{B})C + \overline{A}(B \oplus C) \end{aligned}$$

Vamos agora utilizar o axioma da distributividade da soma lógica em relação ao produto lógico, que afirma que $X + YZ = (X + Y)(X + Z)$ para obter

$$\begin{aligned} f(A, B, C) &= (A \oplus B)C + \overline{A}(B \oplus C) \\ &= (A + B)(\overline{A} + \overline{B})C + \overline{A}(B \oplus C) \\ &= [(A + B) + \overline{A}(B \oplus C)][(\overline{A} + \overline{B}) + \overline{A}(B \oplus C)][C + \overline{A}(B \oplus C)] \end{aligned}$$

Vamos agora analisar cada um dos 3 factores do produto em separado, começando pelo factor $(A + B) + \overline{A}(B \oplus C)$. Como $X + \overline{X}Y = X + Y$ (é um dos teoremas que demos sem demonstração — convém que o aluno se convença que ele está correcto, demonstrando-o), segue-se que

$$\begin{aligned} A + B + \overline{A}(B \oplus C) &= A + B + (B \oplus C) \\ &= A + B + (B\overline{C} + \overline{B}C) \\ &= A + (B + B\overline{C}) + (B + \overline{B}C) \\ &= A + B(1 + \overline{C}) + (B + C) \\ &= A + B + C \end{aligned}$$

Passemos agora ao segundo factor, $(\overline{A} + \overline{B}) + \overline{A}(B \oplus C)$. Fica

$$\begin{aligned} \overline{A} + \overline{B} + \overline{A}(B \oplus C) &= \overline{A}[1 + (B \oplus C)] + \overline{B} \\ &= \overline{A} + \overline{B}. \end{aligned}$$

Quanto ao terceiro factor, $C + \overline{A}(B \oplus C)$, atendemos mais uma vez a que $X + \overline{X}Y = X + Y$ para escrever

$$\begin{aligned} C + \overline{A}(B \oplus C) &= (\overline{A} + C)[C + (B \oplus C)] \\ &= (\overline{A} + C)[C + (B\overline{C} + \overline{B}C)] \\ &= (\overline{A} + C)[(C + B\overline{C}) + (C + \overline{B}C)] \\ &= (\overline{A} + C)[(C + B) + C(1 + \overline{B})] \\ &= (\overline{A} + C)[(C + B) + C] \\ &= (\overline{A} + C)(B + C), \end{aligned}$$

Fica, então,

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

Lembremo-nos agora que $(X + Y)X = X + XY = X(1 + Y) = X$ para obter $(A + B + C)(B + C) = B + C$. Fica, então,

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

Esta expressão já é razoavelmente simples. Mas vamos tentar torná-la ainda mais simples, já agora mantendo a forma normal conjuntiva (produto de somas). Para isso, vamos expandir os produtos, transformando a expressão numa soma de produtos, e em seguida voltamos a convertê-la num produto de somas. Ora

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

e

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

Então,

$$\begin{aligned} f(A, B, C) &= (\overline{A} + \overline{B})(\overline{A} + C)(B + C) \\ &= (\overline{A} + \overline{B}C)(\overline{A}B + C) \\ &= \overline{A}B + \overline{A}C + \overline{B}C \\ &= \overline{A}B + \overline{B}B + \overline{A}C + \overline{B}C \\ &= (\overline{A} + \overline{B})B + (\overline{A} + \overline{B})C \\ &= (\overline{A} + \overline{B})(B + C). \end{aligned}$$

4.5 b)

b) Este problema é muito simples de resolver se pensarmos um pouco e tentarmos ser económicos. Com efeito:

$$A + \overline{A}BC = A + BC.$$

pelo teorema da redundância. Enfim, embora pequena, trata-se de uma simplificação. Continuemos ...

$$C\overline{D} + \overline{C}D = C \oplus D,$$

pelo que fica

$$\begin{aligned} A + \overline{A}BC + C\overline{D} \oplus (C\overline{D} + \overline{C}D) + \overline{C}D \oplus (\overline{C}D + C\overline{D}) &= \\ &= A + BC + C\overline{D} \oplus (C \oplus D) + \overline{C}D \oplus (C \oplus D) \end{aligned}$$

Agora convém “lembrarmo-nos” que o Ou-exclusivo é associativo e comutativo, pelo que

$$\begin{aligned} A + BC + C\overline{D} \oplus (C \oplus D) + \overline{C}D \oplus (C \oplus D) &= \\ &= A + BC + (C\overline{D} \oplus C) \oplus D + (\overline{C}D \oplus D) \oplus C \end{aligned}$$

Agora atendemos a que, por definição de OU-exclusivo,

$$(XY) \oplus X = \overline{X}\overline{Y} \cdot X + XY \cdot \overline{X} = (\overline{X} + \overline{Y}) \cdot X + 0 = X\overline{Y},$$

pelo que

$$\begin{aligned} C\overline{D} \oplus C &= CD \\ (C\overline{D} \oplus C) \oplus D &= CD \oplus D = \overline{C}D \end{aligned}$$

e

$$\begin{aligned} \overline{C}D \oplus D &= CD \\ (\overline{C}D \oplus D) \oplus C &= CD \oplus C = C\overline{D}. \end{aligned}$$

Segue-se que

$$A + BC + C\overline{D} \oplus (C \oplus D) + \overline{C}D \oplus (C \oplus D) = A + BC + \overline{C}D + C\overline{D}.$$

Vamos agora transformar $\overline{C}D + C\overline{D}$ num produto de somas, atendendo à distributividade da soma lógica em relação ao produto lógico:

$$\begin{aligned}\overline{C}D + C\overline{D} &= (\overline{C} + C)(\overline{C} + \overline{D})(C + D)(D + \overline{D}) \\ &= (\overline{C} + \overline{D})(C + D).\end{aligned}$$

Fica, portanto, a seguinte expressão simplificada para a função

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

Evidentemente, não podemos garantir que esta expressão para f é mínima. Apenas podemos garantir uma expressão “suficientemente simples” para a função. Este é, aliás, o principal inconveniente da simplificação algébrica.

4.8 Represente por uma soma de mintermos e por um produto de maxtermos a função

$$f(A, B, C) = (A + B)C + (A \oplus C)AB.$$

Resolução: Podíamos escrever a tabela de verdade da função e dela deduzir as expressões em forma canónica disjuntiva e conjuntiva. Em alternativa, vamos obter essas expressões por via algébrica. Fazemos, para obter a soma de mintermos:

4.8

$$\begin{aligned}f(A, B, C) &= (A + B)C + (A \oplus C)AB \\ &= AC + BC + (A\overline{C} + \overline{A}C)AB \\ &= AC + BC + A\overline{C}AB + \overline{A}CAB \\ &= AC + BC + AB\overline{C} \\ &= A(B + \overline{B})C + (A + \overline{A})BC + AB\overline{C} \\ &= ABC + A\overline{B}C + ABC + \overline{A}BC + AB\overline{C} \\ &= ABC + A\overline{B}C + \overline{A}BC + AB\overline{C} \\ &= m_7 + m_5 + m_3 + m_6 \\ &= \sum m(3, 5 - 7),\end{aligned}$$

se admitirmos que A é a variável booleana simples com maior peso e C a de menor peso.

Para obter o produto de maxtermos fazemos:

$$\begin{aligned}f(A, B, C) &= (A + B)C + (A \oplus C)AB \\ &= (A + B)C + (A + C)(\overline{A} + \overline{C})AB\end{aligned}$$

Vamos agora usar a distributividade da soma lógica em relação ao produto lógico (aquela propriedade que os alunos geralmente “esquecem”) que diz que

$XY + Z = (X + Z)(Y + Z)$, para obtermos:

$$\begin{aligned} f(A, B, C) &= (A + B)C + (A \oplus C)AB \\ &= (A + B)C + (A + C)(\overline{A} + \overline{C})AB \\ &= [(A + B) + (A + C)(\overline{A} + \overline{C})AB][C + (A + C)(\overline{A} + \overline{C})AB] \end{aligned}$$

Vamos agora ver como podemos simplificar $(A + C)(\overline{A} + \overline{C})AB$. Temos

$$\begin{aligned} (A + C)(\overline{A} + \overline{C})AB &= A(\overline{A} + \overline{C})AB + C(\overline{A} + \overline{C})AB \\ &= AB(\overline{A} + \overline{C}) + ABC(\overline{A} + \overline{C}) \\ &= AB\overline{A} + AB\overline{C} + ABC\overline{A} + ABC\overline{C} \\ &= AB\overline{C}, \end{aligned}$$

pelo que fica

$$\begin{aligned} f(A, B, C) &= [(A + B) + (A + C)(\overline{A} + \overline{C})AB][C + (A + C)(\overline{A} + \overline{C})AB] \\ &= [(A + B) + AB\overline{C}](C + AB\overline{C}) \\ &= (A + B)(C + AB\overline{C}) \\ &= (A + B)(C + A)(C + B)(C + \overline{C}) \\ &= (A + B)(C + A)(C + B). \end{aligned}$$

Vamos agora expandir $(A + B)(A + C)(B + C)$ de modo a incluir todas as variáveis (para termos maxtermos). Fica

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

Logo,

$$\begin{aligned} f(A, B, C) &= (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + B + C) \\ &= M_0 M_1 M_2 M_4 \\ &= \prod M(0, 1, 2, 4), \end{aligned}$$

ainda admitindo que A é a variável com maior peso e C a de menor peso.

Naturalmente, este era o resultado que esperávamos depois de obter a primeira forma canónica, já que os índices dos maxtermos formam o conjunto complementar dos índices dos mintermos, e vice-versa.

4.9 Dada a função

$$f(A, B, C, D) = (A + B)\overline{C} + A(C \oplus D) + AB\overline{C}D,$$

obtenha:

- a tabela de verdade;
- a expressão em soma de mintermos;
- a expressão em produto de maxtermos;
- a expressão em soma de mintermos da função $\overline{f}(A, B, C, D)$.

Tabela 4.6: Tabela de verdade da função do Exercício 4.9

Linha #	A	B	C	D	f(A, B, C, D)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

4.9 a)*Resolução:* a) A tabela de verdade de f encontra-se na Tabela 4.6.**4.9 b)**

b) Da tabela de verdade deduz-se a primeira forma canónica (forma canónica disjuntiva) da função:

$$\begin{aligned}
f(A, B, C, D) &= \sum m(4, 5, 8 - 10, 12 - 14) \\
&= \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + \\
&\quad + A\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D},
\end{aligned}$$

admitindo que A é a variável com maior peso e D a de menor peso.

c) Da tabela de verdade deduz-se também a segunda forma canónica (forma canónica conjuntiva) da função:

4.9 c)

$$\begin{aligned}
f(A, B, C, D) &= \prod M(0 - 3, 6, 7, 11, 15) \\
&= (A + B + C + D) \cdot (A + B + C + \overline{D}) \cdot (A + B + \overline{C} + D) \cdot \\
&\quad \cdot (A + B + \overline{C} + \overline{D}) \cdot (A + \overline{B} + \overline{C} + D) \cdot (A + \overline{B} + \overline{C} + \overline{D}) \cdot \\
&\quad \cdot (\overline{A} + B + \overline{C} + \overline{D}) \cdot (\overline{A} + \overline{B} + \overline{C} + \overline{D}),
\end{aligned}$$

ainda admitindo que A é a variável com maior peso e D a de menor peso.d) A função \overline{f} tem “1”s onde f tem “0”s, e vice-versa. Logo, a sua primeira forma canónica (forma canónica conjuntiva) é:**4.9 d)**

$$\begin{aligned}
\overline{f}(A, B, C, D) &= \sum m(0 - 3, 6, 7, 11, 15) \\
&= \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \\
&\quad + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}C\overline{D} + ABCD,
\end{aligned}$$

mais uma vez admitindo que A é a variável com maior peso e D a de menor peso.

4.10 Utilizar o conjunto completo {AND,OR,NOT} para representar algebricamente (em somas de produtos) as seguintes funções booleanas simples:

- a) $f_1 = \overline{(a \oplus b \oplus c)} \bar{a}$;
 b) $f_2 = \overline{(a \odot b)} \odot c$.

4.10 a)

Resolução: a) Como $x \oplus y \stackrel{\text{def}}{=} x\bar{y} + \bar{x}y$, segue-se que:

$$\begin{aligned} a \oplus b \oplus c &= (a \oplus b) \oplus c \\ &= (a \oplus b) \bar{c} + \overline{(a \oplus b)} c \\ &= (a\bar{b} + \bar{a}b) \bar{c} + (a\bar{b} + \bar{a}b) c \\ &= a\bar{b}\bar{c} + \bar{a}b\bar{c} + abc + \bar{a}\bar{b}c, \end{aligned}$$

pelo que

$$\begin{aligned} f_1 &= (a \oplus b \oplus c) \bar{a} \\ &= (a\bar{b}\bar{c} + \bar{a}b\bar{c} + abc + \bar{a}\bar{b}c) \bar{a} \\ &= \bar{a}b\bar{c} + \bar{a}\bar{b}c \end{aligned}$$

4.10 b)

b) Como sabemos, $\overline{x \odot y} = x \oplus y$, pelo que $\overline{(a \odot b) \odot c} = (a \odot b) \oplus c$. Segue-se que:

$$\begin{aligned} f_2 &= \overline{(a \odot b) \odot c} \\ &= (a \odot b) \oplus c \\ &\stackrel{\text{def}}{=} (a \odot b) \bar{c} + \overline{(a \odot b)} c \\ &= (a \odot b) \bar{c} + (a \oplus b) c \\ &\stackrel{\text{def}}{=} (a\bar{b} + \bar{a}b) \bar{c} + (a\bar{b} + \bar{a}b) c \\ &= a\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c + \bar{a}bc \end{aligned}$$

4.11 Representar as seguintes funções booleanas simples em primeira forma canónica:

- a) $f_1 = \overline{(a \oplus b \oplus c)} \bar{a}$;
 b) $f_2 = \overline{(a \odot b)} \odot c$.

4.11 a)

Resolução: a) No exercício anterior obteve-se a primeira forma canónica da função:

$$\begin{aligned} f_1 &= \bar{a}b\bar{c} + \bar{a}\bar{b}c \\ &= m_2 + m_1 \\ &= \sum m(1, 2), \end{aligned}$$

se admitirmos que a é a variável booleana simples com maior peso e c a de menor peso.

4.11 b)

b) A primeira forma canónica da função já foi obtida no exercício anterior:

$$\begin{aligned}
 f_2 &= a b \bar{c} + \bar{a} \bar{b} \bar{c} + a \bar{b} c + \bar{a} b c \\
 &= m_6 + m_0 + m_5 + m_3 \\
 &= \sum m(0, 3, 5, 6),
 \end{aligned}$$

ainda se admitirmos que a é a variável booleana simples com maior peso e c a de menor peso.

4.12 Representar as seguintes funções booleanas simples em segunda forma canónica:

- a) $f_1 = (a \oplus b \oplus c) \bar{a}$;
 b) $f_2 = (a \odot b) \odot c$.

Resolução: a) Como $f_1 = \sum m(1, 2)$, conclui-se que $f_1 = \prod M(0, 3 - 7)$ ou

4.12 a)

$$f_1 = (a + b + c) (a + \bar{b} + \bar{c}) (\bar{a} + b + c) (\bar{a} + b + \bar{c}) (\bar{a} + \bar{b} + c) (\bar{a} + \bar{b} + \bar{c}),$$

se admitirmos que a é a variável booleana simples com maior peso e c a de menor peso.

b) Como $f_2 = \sum m(0, 3, 5, 6)$, conclui-se que $f_2 = \prod M(1, 2, 4, 7)$ ou

4.12 b)

$$f_2 = (a + b + \bar{c}) (a + \bar{b} + c) (\bar{a} + b + c) (\bar{a} + \bar{b} + \bar{c}),$$

se admitirmos que a é a variável booleana simples com maior peso e c a de menor peso.

4.14 Considere o logigrama da Figura 4.3 (de *SD:AAT*). Redesenhe-o da forma mais simples que conseguir.

Resolução: Para simplificar o logigrama temos de simplificar a função, o que faremos algebricamente.

4.14

Do logigrama tira-se que $F = (A D + B) \oplus (B \bar{C} + B \bar{A} \bar{D})$, pelo que:

$$\begin{aligned}
 F &= (A D + B) \oplus (B \bar{C} + B \bar{A} \bar{D}) \\
 &\stackrel{\text{def}}{=} (A D + B) (\overline{B \bar{C} + B \bar{A} \bar{D}}) + (\overline{A D + B}) (B \bar{C} + B \bar{A} \bar{D}).
 \end{aligned}$$

Mas

$$\begin{aligned}
 (\overline{A D + B}) (B \bar{C} + B \bar{A} \bar{D}) &= (\bar{A} + \bar{D}) \bar{B} (B \bar{C} + B \bar{A} \bar{D}) \\
 &= 0,
 \end{aligned}$$

pelo que

$$\begin{aligned}
 F &= (A D + B) (\overline{B \bar{C} + B \bar{A} \bar{D}}) \\
 &= (A D + B) (\bar{B} + C) (\bar{B} + A + D).
 \end{aligned}$$

Esta expressão de F necessita de 5 portas lógicas com um total de 12 entradas (admitindo que se dispõe de \bar{B} à entrada): um AND de 3 entradas para formar o produto final, um AND de 2 entradas para formar $A D$, um OR de 2 entradas

para formar $AD + B$, outro OR de 2 entradas para formar $\overline{B} + C$ e ainda um OR de 3 entradas para formar $\overline{B} + A + D$.

Vamos obter expressões alternativas para F , por exemplo

$$\begin{aligned} F &= (AD + B)(\overline{B} + C)(\overline{B} + A + D) \\ &= (AD + B)(\overline{B} + AC + CD). \end{aligned}$$

Agora precisamos de um OR de 2 entradas para formar o produto final, de um AND de 2 entradas para formar AD , um OR de 2 entradas para formar $AD + B$, de dois ANDs de 2 entradas para formar AC e CD , e de um OR de 3 entradas para formar $\overline{B} + AC + CD$, num total de 6 portas com 13 entradas. Claramente, esta expressão é mais “dispendiosa” do que a anterior.

Consideremos outra expressão para F :

$$\begin{aligned} F &= (AD + B)(\overline{B} + AC + CD) \\ &= A\overline{B}D + ACD + ABC + BCD. \end{aligned}$$

Esta expressão necessita de um OR de 4 entradas para formar a soma global, e de 4 ANDs de 3 entradas para formar cada uma das parcelas, num total de 5 portas com 16 entradas (a primeira solução continua a ser a mais “económica”).

Finalmente, podemos obter

$$\begin{aligned} F &= A\overline{B}D + ACD + ABC + BCD \\ &= AD(\overline{B} + C) + BC(A + D), \end{aligned}$$

uma solução que precisa de um OR de 2 entradas para formar a soma global, de dois ANDs de 3 entradas para formar $AD(\overline{B} + C)$ e $BC(A + D)$, e de 2 ORs de 2 entradas para formar $\overline{B} + C$ e $A + D$, num total de 5 portas com 12 entradas, ou seja, um logigrama com uma “complexidade” igual à da primeira expressão.

Não vamos tentar obter outras expressões (podíamos obter muitas outras). Ficamo-nos por uma das soluções mais económicas,

$$\begin{aligned} F &= (AD + B)(\overline{B} + C)(\overline{B} + A + D) \\ &= AD(\overline{B} + C) + BC(A + D), \end{aligned}$$

com 5 portas e 12 entradas, e desenhámos o logigrama da segunda solução na Figura 4.1.

4.15 Usando apenas:

- a) NANDs;
- b) NORs;
- c) AOIs,

desenhe o logigrama da seguinte função:

$$f(A, B, C) = (A \oplus C)B + \overline{B}C + AC.$$

(Nota: **AOI** é a sigla de “**And-Or Invert**”. Ou seja, o logigrama deve apresentar um primeiro andar com portas AND e um segundo andar com uma porta NOR).

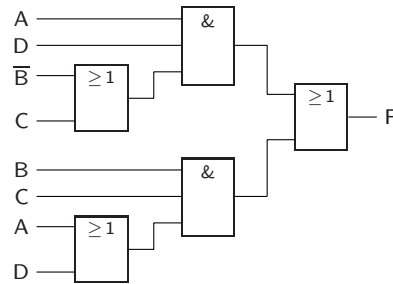


Figura 4.1: Logigrama da expressão $F = AD(\overline{B} + C) + BC(A + D)$

4.15 a)

Resolução: a) Vamos simplificar a expressão da função. Fica:

$$\begin{aligned}
 f(A, B, C) &= (A \oplus C)B + \overline{B}C + AC \\
 &\stackrel{\text{def}}{=} AB\overline{C} + \overline{A}BC + \overline{B}C + AC \\
 &= A(C + B\overline{C}) + (\overline{B} + \overline{A}B)C \\
 &= A(C + B) + (\overline{B} + \overline{A})C \\
 &= AB + AC + \overline{A}C + \overline{B}C \\
 &= AB + C + \overline{B}C \\
 &= AB + C.
 \end{aligned}$$

Vamos agora transformar a expressão simplificada de f por forma a apenas incluir NANDs.

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

a que corresponde o logigrama da Figura 4.2.

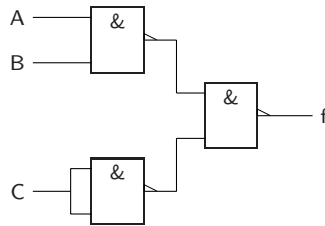


Figura 4.2: Logigrama de f só com NANDs

b) Vamos agora transformar a expressão simplificada de f por forma a apenas incluir NORs.

4.15 b)

$$\begin{aligned}
 f(A, B, C) &= AB + C \\
 &= (A + C)(B + C) \\
 &= \overline{\overline{(A + C)(B + C)}} \\
 &= \overline{\overline{(A + C)} + \overline{(B + C)}},
 \end{aligned}$$

a que corresponde o logigrama da Figura 4.3.

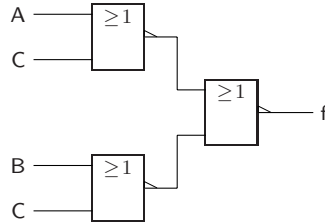


Figura 4.3: Logigrama de f só com NORs

4.15 c)

c) Vamos agora obter um logigrama com um AOI. Como o AOI tem um NOR no fim, geramos a expressão booleana de \bar{f} .

$$\begin{aligned}\bar{f}(A, B, C) &= \overline{AB + C} \\ &= \overline{AB} \cdot \overline{C} \\ &= (\overline{A} + \overline{B}) \cdot \overline{C} \\ &= \overline{A}\overline{C} + \overline{B}\overline{C},\end{aligned}$$

a que corresponde o logigrama para a função f da Figura 4.4.

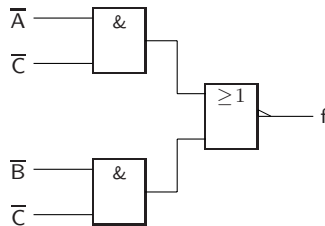


Figura 4.4: Logigrama de f com um AOI

Capítulo 5

Método de Karnaugh

5.1 Nas tabelas de verdade das funções booleanas simples

- a) $f_1(A, B, C) = (A + B)C + \overline{A}(B + C)$; e
b) $f_2(A, B, C, D) = A + \overline{A}BC + C\overline{D} \oplus (C\overline{D} + \overline{C}D)$,

identificar todas as linhas adjacentes às linhas em que as funções têm o valor 1.

Resolução: a) Tal como fizemos no Exercício 3.14, vamos construir as tabelas de verdade pedidas usando um CBR em vez do CBN habitual. Isso permite, como se viu nesse exercício, identificar facilmente as linhas da tabela que são adjacentes a uma dada linha, isto é, que só diferem dela numa variável.

5.1 a)

Tabela 5.1: Tabela de verdade da função $f_1(A, B, C)$ que, ao contrário do habitual, ordena as linhas segundo as palavras de um CBR (e não do CBN)

Linha #	A	B	C	f_1	Eixos
1	0	0	0	0	Z
2	0	0	1	1	Y
3	0	1	1	1	Z
4	0	1	0	1	X
5	1	1	0	0	Z
6	1	1	1	1	Y
7	1	0	1	1	Z
8	1	0	0	0	

Qualquer que seja a linha que consideremos, há sempre 3 linhas que lhe são adjacentes, porque a função possui 3 variáveis (no caso de uma função de n variáveis, existem n linhas adjacentes a uma dada linha).

Consideremos a linha 2, em que a função vale 1. As linhas adjacentes à 2 são a 1 (eixo Z), a 3 (eixo Y) e a 7 (eixo X).

Consideremos agora a linha 3, em que a função vale 1. As linhas adjacentes à 3 são a 2 (eixo Y), a 4 (eixo Z) e a 6 (eixo X).

Consideremos a linha 4, em que a função vale 1. As linhas adjacentes à 4 são a 3 (eixo Z), a 0 (eixo Y) e a 5 (eixo X).

Consideremos agora a linha 6, em que a função vale 1. As linhas adjacentes à 6 são a 7 (eixo Y), a 5 (eixo Z) e a 3 (eixo X).

Finalmente, consideremos a linha 7, em que a função vale 1. As linhas adjacentes à 7 são a 6 (eixo Y), a 8 (eixo Z) e a 2 (eixo X).

5.1 b)

b) A tabela de verdade de $f_2(A, B, C, D)$ está na Tabela 5.2.

Tabela 5.2: Tabela de verdade da função $f_2(A, B, C, D)$ que, ao contrário do habitual, ordena as linhas segundo as palavras de um CBR (e não do CBN)

Linha #	A	B	C	D	Z	Eixos
1	0	0	0	0	0	Z
2	0	0	0	1	1	Y
3	0	0	1	1	0	Z
4	0	0	1	0	0	X
5	0	1	1	0	1	Z
6	0	1	1	1	1	Y
7	0	1	0	1	1	Z
8	0	1	0	0	0	V
9	1	1	0	0	1	Z
10	1	1	0	1	1	Y
11	1	1	1	1	1	Z
12	1	1	1	0	1	X
13	1	0	1	0	1	Z
14	1	0	1	1	1	Y
15	1	0	0	1	1	Z
16	1	0	0	0	1	

As linhas adjacentes às linhas em que $f_2 = 1$ estão na Tabela 5.3.

5.3 Identificar todos os agrupamentos legítimos de dois “1”s no quadro de Karnaugh da função booleana simples $F(A, B, C) = \sum m(1-7)$, e indicar as correspondentes expressões booleanas.

5.3

Resolução: Dado o elevado número de agrupamentos, eles encontram-se repartidos por 3 figuras, como mostra a Figura 5.1.

De notar os seguintes agrupamentos:

$$— m_1 + m_3 = \overline{A}\overline{B}C + \overline{A}BC = \overline{A}(B + \overline{B})C = \overline{A}C;$$

$$— m_4 + m_6 = A\overline{B}\overline{C} + AB\overline{C} = A(B + \overline{B})\overline{C} = A\overline{C};$$

$$— m_5 + m_7 = A\overline{B}C + ABC = A(B + \overline{B})C = AC;$$

$$— m_2 + m_3 = \overline{A}B\overline{C} + \overline{A}BC = \overline{A}B(C + \overline{C}) = \overline{A}B;$$

Tabela 5.3: Linhas da tabela de verdade de $f_2(A, B, C)$ que são adjacentes às linhas em que a função vale 1

Linha #	Eixo V Linha adj	Eixo X Linha adj	Eixo Y Linha adj	Eixo Z Linha adj
2	15	7	3	1
5	12	4	8	6
6	11	3	7	5
7	10	2	6	8
9	8	16	12	10
10	7	15	11	9
11	6	14	10	12
12	5	13	9	11
13	4	12	16	14
14	3	11	15	13
15	2	10	14	16
16	1	9	13	15

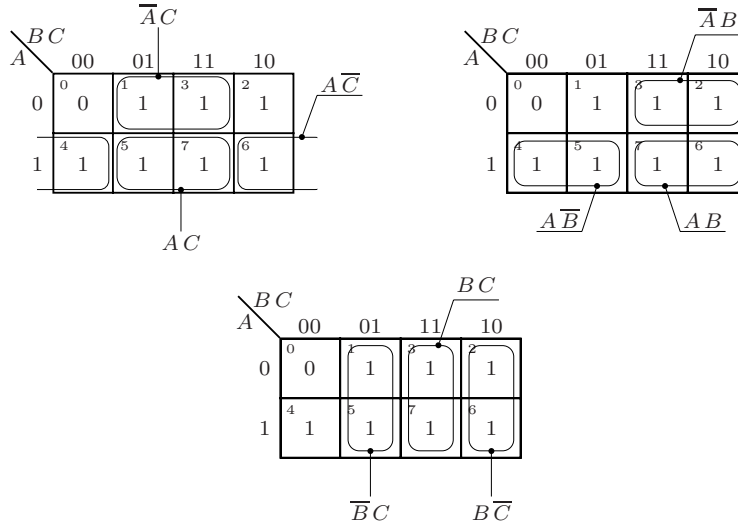


Figura 5.1: Quadro de Karnaugh da função $F(A, B, C) = \sum m(1-7)$ onde se identificam todos os agrupamentos legítimos de dois “1”s

- $m_4 + m_5 = A\overline{B}\overline{C} + A\overline{B}C = A\overline{B}(C + \overline{C}) = A\overline{B}$;
- $m_6 + m_7 = AB\overline{C} + ABC = AB(C + \overline{C}) = AB$;
- $m_1 + m_5 = \overline{A}\overline{B}C + A\overline{B}C = (A + \overline{A})\overline{B}C = \overline{B}C$;
- $m_3 + m_7 = \overline{A}BC + ABC = (A + \overline{A})BC = BC$; e
- $m_2 + m_6 = \overline{A}B\overline{C} + AB\overline{C} = (A + \overline{A})B\overline{C} = B\overline{C}$.

Naturalmente, também podemos fazer agrupamentos legítimos de quatro “1”s, por exemplo:

$$— m_1 + m_3 + m_5 + m_7 = C;$$

$$— m_4 + m_5 + m_6 + m_7 = A; \text{ e}$$

$$— m_2 + m_3 + m_6 + m_7 = B,$$

que, contudo, não são pedidos neste exercício.

5.5 Dada a função $F = \sum m(0 - 2, 4 - 7, 10)$, dizer se os seguintes mintermos e somas de mintermos são ou não implicantes de F :

- a) m_1 ;
- b) m_3 ;
- c) $m_1 + m_2$;
- d) $m_1 + m_3$;
- e) $m_0 + m_1 + m_2$;
- d) $m_4 + m_5 + m_6 + m_7$.

5.5 a)

Resolução: a) Consideremos as tabelas de verdade de m_1 e de F , na Tabela 5.4.

Tabela 5.4: Tabela de verdade de $m_1 = \overline{A}\overline{B}\overline{C}D$ e de $F = \sum m(0 - 2, 4 - 7, 10)$

A	B	C	D	$m_1 = \overline{A}\overline{B}\overline{C}D$	$F = \sum m(0 - 2, 4 - 7, 10)$
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Notemos que m_1 apenas contém um 1, na linha correspondente ao seu índice (linha 1). Este é um facto esperado, já que $m_1 = \overline{A}\overline{B}\overline{C}D$ apenas vale 1 quando $A = B = C = 0$ e $D = 1$.

Por outro lado, notemos que F também contém um 1 na linha 1 (para além de conter outros “1”s, noutras linhas). Também esperávamos este facto, já que F

é a soma de m_1 com outros mintermos, e que basta que m_1 valha 1 para que F também tenha o valor 1 (o que acontece na linha 1).

Finalmente notemos que, em consequência do que foi dito anteriormente, nunca existe, para uma linha i qualquer, a situação em que $m_1 = 1$ e $F = 0$. Ora esta é precisamente a condição de **implicação**. Esta condição pode traduzir-se pelas seguintes condições:

Implicação

- $0 \rightarrow 0$ (o que se lê “0 implica 0”) é verdadeiro;
- $0 \rightarrow 1$ também é verdadeiro;
- $1 \rightarrow 1$ é ainda verdadeiro; mas
- $1 \rightarrow 0$ é falso.

Estas condições resultam da definição de **função implicação**, uma das funções de 2 variáveis que se estudaram na Secção 3.3 de *SD:AAT* (a função f_{11} da Tabela 3.2).

Função implicação

Ou seja, concluímos que $m_1 \rightarrow F$ ou que m_1 é um *implicante* de F .

Aliás, pelo raciocínio anterior podemos inferir que *qualquer mintermo de uma função é implicante da função*.



b) m_3 não é mintermo de F , logo não pode ser implicante da função.

5.5 b)

c) m_1 e m_2 são mintermos de F , logo cada um deles é, individualmente, um implicante de F . Mas será que a sua soma também é implicante de F ? Vamos ver que sim.

5.5 c)

Tabela 5.5: Tabela de verdade de $m_1 + m_2$ e de $F = \sum m(0 - 2, 4 - 7, 10)$

A	B	C	D	m_1	m_2	$m_1 + m_2$	$F = \sum m(0 - 2, 4 - 7, 10)$
0	0	0	0	0	0	0	1
0	0	0	1	1	0	1	1
0	0	1	0	0	1	1	1
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	1
0	1	1	1	0	0	0	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

Para tanto, consideremos as tabelas de verdade de $m_1 + m_2$ e de F , na Tabela 5.5, e notemos como nunca se verifica a condição $m_1 + m_2 = 1$ com $F = 0$. Logo, $m_1 + m_2 \rightarrow F$, e $m_1 + m_2$ é um implicante da função.



Aliás, pelo raciocínio anterior podemos inferir que *qualquer soma de mintermos de uma função é um implicante da função*.

5.5 d)

d) Como m_3 não é mintermo de F , a soma $m_1 + m_3$ não pode ser implicante da função.

5.5 e)

e) Como m_0, m_1 e m_2 são mintermos de F , a sua soma é um implicante de F .

5.5 f)

f) Como m_4, m_5, m_6 e m_7 são mintermos de F , a sua soma é um implicante de F .



Notemos, para terminar, que *dada a expressão de uma função em soma de produtos, cada uma das parcelas da expressão é um implicante da função*. Com efeito, sendo a expressão uma soma dessas parcelas, basta que uma delas venha a 1 para que a função também venha a 1, e a condição de ser a parcela igual a 1 e a função igual a 0 nunca se verifica.

Por exemplo, dada $G = AB + \overline{A}C$, conclui-se que $AB \rightarrow G$ e que $\overline{A}C \rightarrow G$.

5.6 Identificar todos os implicantes primos essenciais das funções booleanas simples que se seguem (admita que A é a variável booleana simples com maior peso):

- a) $F_1(A, B, C, D) = \sum m(0 - 2, 4 - 7, 10)$;
- b) $F_2(A, B, C, D) = ABC + A\overline{B}D + BC + D$,

e identificar os correspondentes quadrados essenciais. Identificar ainda, para cada uma das funções, pelo menos 2 implicantes primos não essenciais.

5.6 a)

Implicante primo
essencial

Implicante primo (não
essencial)

Implicante (não primo)

Resolução: a) Antes de responder ao exercício, vamos recordar o que se entende por **implicante primo essencial**, por **implicante primo (não essencial)**, e por **implicante (não primo)**.

Começamos por reparar na nomenclatura utilizada. Um *implicante primo essencial* é sempre identificado como tal, pelo que um *implicante primo não essencial* é, em geral, designado mais simplesmente por *implicante primo* (apenas). Por razões semelhantes, um *implicante não primo* é geralmente designado de forma mais simples por *implicante* (apenas).



Consideremos então a função $F_1(A, B, C, D) = \sum m(0 - 2, 4 - 7, 10)$ e tomemos a soma de mintermos

$$\overline{A}\overline{C} = m_0 + m_1 + m_4 + m_5,$$

na Figura 5.2.

Como sabemos do exercício anterior, qualquer mintermo de F_1 ou qualquer soma de mintermos de F_1 implica F_1 , ou é um implicante de F_1 ; então $\overline{A}\overline{C}$ é um *implicante de F_1* .

Porém, trata-se de um implicante de F_1 muito especial, na medida em que constitui um “*agrupamento máximo*”, isto é, um *agrupamento legítimo que não pode ser incluído dentro de um outro agrupamento, igualmente legítimo, mas maior* (que só poderia ser, neste caso, formado por oito “1”s). Como $\overline{A}\overline{C}$ é um agrupamento máximo, ele também é um *implicante primo de F_1* .



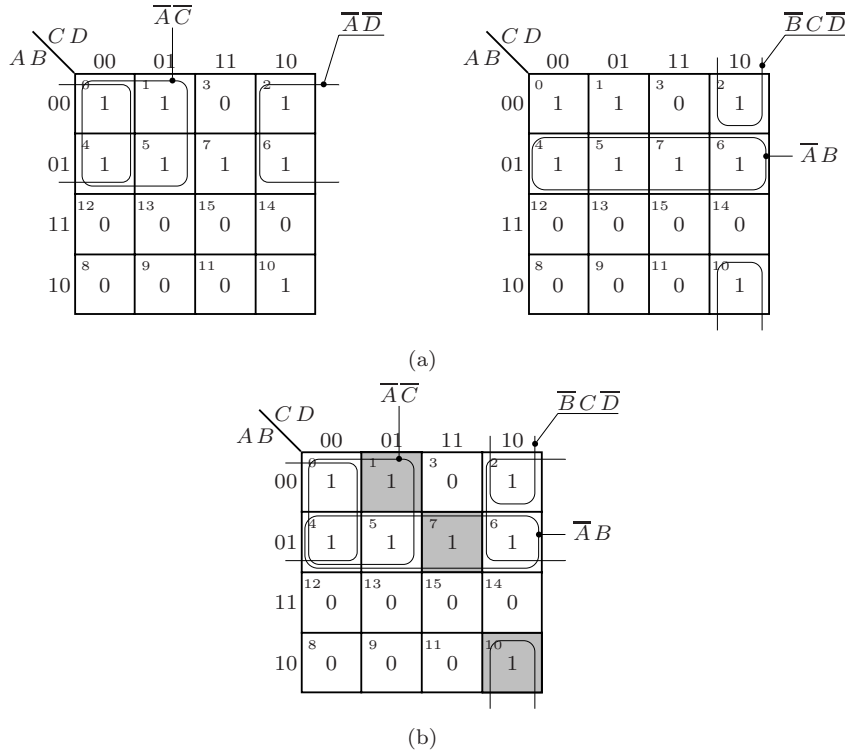


Figura 5.2: (a) Quadro de Karnaugh da função $F_1(A, B, C, D) = \sum m(0 - 2, 4 - 7, 10)$, onde se identificam quatro agrupamentos legítimos de “1”s que constituem implicantes primos da função; (b) os quadrados essenciais e os implicantes primos essenciais

Notemos, por exemplo, que agrupamentos legítimos de dois “1”s incluídos em $\overline{A}\overline{C}$ (por exemplo, $m_0 + m_1$, ou $m_4 + m_5$, ou $m_0 + m_4$ ou, finalmente, $m_1 + m_5$) constituem implicantes de F_1 sem, contudo, serem primos, porque não são “agrupamentos máximos” (estão contidos em $\overline{A}\overline{C}$, que é um agrupamento legítimo mas maior do que qualquer dos agrupamentos de dois “1”s).

Da mesma forma, cada um dos “1”s isolados contidos em $\overline{A}\overline{C}$ (m_0 , ou m_1 , ou m_4 , ou m_5) constituem implicantes de F_1 que também não são primos, porque não são “agrupamentos máximos” (estão contidos em $\overline{A}\overline{C}$, que é um agrupamento legítimo maior do que qualquer dos agrupamentos de “1”s isoladamente).

Seguindo o mesmo raciocínio, podíamos deduzir que F_1 contém três outros implicantes primos,

$$\overline{A}B = m_4 + m_5 + m_6 + m_7,$$

$$\overline{B}C\overline{D} = m_2 + m_{10}$$

e

$$\overline{A}\overline{D} = m_0 + m_2 + m_4 + m_6,$$

igualmente identificados na Figura 5.2.

Quanto a implicantes não primos de F_1 , já vimos vários: por exemplo m_0 , ou $m_4 + m_5$, que constituem agrupamentos legítimos de “1”s mas que não são

máximos.

Veamos agora como identificar os *implicantes primos essenciais* de F_1 . Um implicante primo essencial de uma função é, antes de tudo, um implicante primo da função. Logo, neste exercício, só poderão ser (eventualmente) implicantes primos essenciais de F_1 os quatro, ou apenas alguns dos quatro, implicantes primos que foram determinados anteriormente.



Se um determinado implicante primo de uma função possuir pelo menos um 1 que não faz parte de nenhum outro implicante primo da função, então o primeiro diz-se ser um implicante primo essencial da função, por causa desse ou desses “1”s.

Por exemplo, consideremos, mais uma vez, o implicante primo

$$\overline{A}\overline{C} = m_0 + m_1 + m_4 + m_5$$

de F_1 . O mintermo m_0 deste implicante primo está contido também noutro implicante primo de F_1 , o implicante primo

$$\overline{A}\overline{D} = m_0 + m_2 + m_4 + m_6,$$

pelo que o 1 no quadrado 0 de $\overline{A}\overline{C}$ não torna este implicante primo essencial.

Da mesma forma, o mintermo m_4 de $\overline{A}\overline{C}$ está contido também em dois outros implicantes primos de F_1 , os implicantes primos

$$\overline{A}B = m_4 + m_5 + m_6 + m_7$$

e

$$\overline{A}\overline{D} = m_0 + m_2 + m_4 + m_6,$$

pelo que o 1 no quadrado 4 de $\overline{A}\overline{C}$ também não torna este implicante primo essencial.

Também o mintermo m_5 de $\overline{A}\overline{C}$ está contido noutro implicante primo de F_1 , o implicante primo

$$\overline{A}B = m_4 + m_5 + m_6 + m_7,$$

pelo que o 1 no quadrado 5 de $\overline{A}\overline{C}$ também não torna este implicante primo essencial.

Mas o mintermo m_1 de $\overline{A}\overline{C}$ não está contido em qualquer outro implicante primo de F_1 , pelo que *o 1 no quadrado 1 de $\overline{A}\overline{C}$ torna este implicante primo essencial*. Esse facto vem assinalado, sombreando o quadrado 1 da Figura 5.2 (diz-se que o quadrado sombreado é um **quadrado essencial**).

Quadrado essencial

De forma idêntica se concluiria que o quadrado 7 torna o implicante primo $\overline{A}B$ essencial, e que o quadrado 10 torna o implicante primo $\overline{B}C\overline{D}$ essencial.

5.6 b)

b) Agora é fácil estabelecer, no quadro de Karnaugh de $F_2 = ABC + A\overline{B}D + B\overline{C}D$, todos os seus implicantes primos e identificar os essenciais (Figura 5.3).

Como podemos constatar, a função apenas possui dois implicantes primos, que são essenciais por causa dos mintermos nos quadrados essenciais (a sombreado).

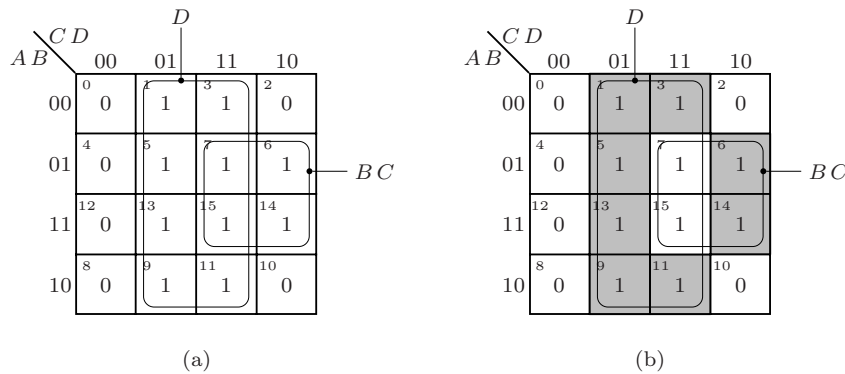


Figura 5.3: (a) Quadro de Karnaugh da função $F_2 = ABC + A\bar{B}D + BC + D$, onde se identificam todos os implicantes primos da função; (b) os quadrados essenciais e os implicantes primos essenciais da função

Não devemos estranhar o resultado tão simples que se obteve, já que a expressão da função pode vir simplificada para

$$\begin{aligned} ABC + A\bar{B}D + BC + D &= (ABC + BC) + (A\bar{B}D + D) \\ &= BC(A + 1) + D(A\bar{B} + 1) \\ &= BC + D, \end{aligned}$$

isto é, a soma dos seus dois implicantes primos essenciais.

5.8 Escrever a ou as formas normais disjuntivas mínimas para a função booleana simples

$$F(A, B, C, D) = \bar{A}\bar{B}\bar{C}\bar{D} + B\bar{C}D + \bar{A}\bar{B}CD + ABCD + \bar{A}BC\bar{D},$$

tendo em conta que nunca surgem as combinações de valores nas entradas correspondentes aos mintermos 1, 4, 7, 10 e 11.

Resolução: A função dada é incompletamente especificada, com indiferenças nas posições 1, 4, 7, 10 e 11 do seu quadro de Karnaugh (Figura 5.4). Por outro lado, pretende-se obter a soma ou somas de produtos (forma ou formas normais disjuntivas) mínimas.

5.8

Notemos que o mintermo m_0 , que apenas pode ser incluído no implicante primo $\bar{A}\bar{C}$, torna este implicante primo essencial ($\bar{A}\bar{C}$ é o nosso primeiro implicante primo essencial, pelo que o designamos por IPE1).

O mintermo m_{15} , por seu turno, não torna o implicante primo BD essencial, porque m_{15} também pode ser incluído no implicante primo CD . Mas m_{13} não pode ser incluído em nenhum outro implicante primo de F , pelo que BD é essencial (designamo-lo por IPE2).

Finalmente, m_6 torna o implicante primo $\bar{A}B$ essencial (designamo-lo por IPE3).

Reparemos que o processo de minimização deve incluir os mintermos (todos os mintermos) da função, mas não necessariamente as indiferenças. Destas, apenas devem ser usadas as que forem estritamente necessárias para, em conjunto



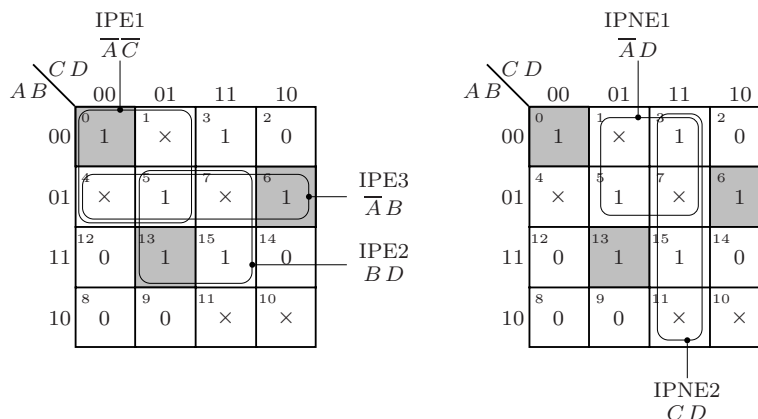


Figura 5.4: (a) Quadro de Karnaugh da função $F = \overline{A}\overline{B}\overline{C}\overline{D} + B\overline{C}D + \overline{A}\overline{B}C\overline{D} + ABC\overline{D} + \overline{A}BC\overline{D}$, onde se identificam todos os implicantes primos essenciais e não essenciais necessários à sua minimização

com os “1”s, formar os maiores agrupamentos que conseguirmos (os implicantes primos da função).

Notemos ainda que, de acordo com o algoritmo de Karnaugh da Secção 5.9 de *SD:AAT*, depois de incluir na soma de produtos mínima *todos* os implicantes primos essenciais, teremos ainda que cobrir os “1”s que sobejem com o *menor número de implicantes primos não essenciais*. Desta etapa do algoritmo pode resultar apenas uma solução mínima, se os “1”s excedentários poderem ser cobertos apenas por um implicante primo não essencial, ou mais do que uma solução mínima no caso contrário.

Neste exercício, a soma lógica dos implicantes primos essenciais não cobre todos os “1”s da função (deixa de fora m_3). Por essa razão, teremos de recorrer a um ou mais implicantes primos não essenciais. Como o 1 correspondente a m_3 pode ser coberto de duas formas diferentes, pelos implicantes primos $\overline{A}\overline{D}$ ou $C\overline{D}$, existem duas somas de produtos mínimas para F .

Reparemos que o implicante primo $\overline{A}\overline{D}$ (que designamos por IPNE1) não é essencial, porque m_3 também pode ser incluído no implicante primo $C\overline{D}$, e porque m_5 , como vimos atrás, também está coberto pelos três implicantes primos essenciais. Outro tanto acontecendo com m_5 . Por sua vez, o implicante primo $C\overline{D}$ também não é essencial, por causa do m_3 também estar incluído em $\overline{A}\overline{D}$, e porque m_{15} também está coberto por pelo implicante primo essencial $B\overline{D}$.

Para cobrir m_3 não precisamos dos dois implicantes primos não essenciais. Basta-nos um deles. Como ambos possuem o mesmo grau de complexidade (o mesmo número de literais), podemos escolher indiferentemente qualquer um deles, o que, de acordo com o passo 4 do algoritmo de Karnaugh, dá origem a duas soluções mínimas para a função, que são:

$$\begin{aligned} F &= \text{IPE1} + \text{IPE2} + \text{IPE3} + \text{IPNE1} \\ &= \overline{A}\overline{C} + B\overline{D} + \overline{A}\overline{B} + \overline{A}\overline{D}, \end{aligned}$$

ou

$$\begin{aligned} F &= \text{IPE1} + \text{IPE2} + \text{IPE3} + \text{IPNE2} \\ &= \overline{A}\overline{C} + B D + \overline{A} B + C D. \end{aligned}$$

Notemos, finalmente, que passam a valer 1 as indiferenças que vêm contidas em implicantes primos (essenciais ou não) que fazem parte de uma certa soma de produtos mínima, enquanto que as outras indiferenças, que ficam de fora no processo de minimização, passam a valer 0. Por exemplo, a indiferença no quadrado 1 passa a valer 1 nas duas expressões mínimas, enquanto que a indiferença no quadrado 10 passa a valer 0.

Ou seja, podemos considerar que a função original, incompletamente especificada, é, na realidade, *um conjunto de funções completamente especificadas (uma por cada valor 0 ou 1 atribuível a cada indiferença)*, e que o processo de minimização selecciona, desse conjunto de funções, um certo número delas, que são mínimas.



5.9 Porque é que o agrupamento formado pela soma de mintermos $m_9 + m_{10} + m_{11} + m_{14}$ num quadro de Karnaugh de 4 variáveis está incorrecto? E num quadro com $n > 4$ variáveis?

Resolução: Porque não forma um agrupamento legítimo.

5.9

Para o agrupamento ser legítimo devemos poder formar dois grupos de dois “1”s (eliminando, em ambos os casos, um literal, mas o mesmo nos dois casos) e depois, com esses dois grupos de dois “1”s, formar um grupo de quatro “1”s (eliminando um segundo literal, diferente do primeiro).

Por exemplo, consideremos o agrupamento $m_0 + m_1 + m_2 + m_3$ num quadro de 4 variáveis, e formemos dois grupos com dois mintermos. Podemos fazê-lo de duas maneiras:

1. Somando m_0 com m_1 para eliminar um literal, e em seguida somar m_2 com m_3 para eliminar *o mesmo* literal. Obtemos (se a função depender das variáveis A, B, C e D , e A for a variável com maior peso)

$$\begin{aligned} m_0 + m_1 &= \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D \\ &= \overline{A}\overline{B}\overline{C}, \end{aligned}$$

eliminando o literal D .

Em seguida somamos

$$\begin{aligned} m_2 + m_3 &= \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD \\ &= \overline{A}\overline{B}C, \end{aligned}$$

voltando a eliminar D .

Em seguida podemos somar $(m_0 + m_1) + (m_2 + m_3)$ para dar

$$\begin{aligned} (m_0 + m_1) + (m_2 + m_3) &= \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C \\ &= \overline{A}\overline{B}, \end{aligned}$$

eliminando agora o literal C .

2. Somando m_0 com m_2 para eliminar um literal diferente do anterior, e em seguida somar m_1 com m_3 para eliminar o mesmo literal. Obtemos, nas mesmas condições que anteriormente,

$$\begin{aligned} m_0 + m_2 &= \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} \\ &= \overline{A}\overline{B}\overline{D}, \end{aligned}$$

eliminando agora o literal C .

Em seguida somamos

$$\begin{aligned} m_1 + m_3 &= \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}CD \\ &= \overline{A}\overline{B}D, \end{aligned}$$

eliminando o mesmo literal que anteriormente, isto é, C .

Em seguida podemos somar $(m_0 + m_2) + (m_1 + m_3)$ para dar

$$\begin{aligned} (m_0 + m_2) + (m_1 + m_3) &= \overline{A}\overline{B}\overline{D} + \overline{A}\overline{B}D \\ &= \overline{A}\overline{B}, \end{aligned}$$

eliminando agora o literal D . E, naturalmente, obtemos para $m_0 + m_1 + m_2 + m_3$ o mesmo resultado que anteriormente, $\overline{A}\overline{B}$.

Comparar, nas Figuras 5.5(a) e (b), as duas formas alternativas de construção do agrupamento de quatro “1”s correspondente a $m_0 + m_1 + m_2 + m_3$, um agrupamento legítimo, à custa de dois agrupamentos de dois “1”s, igualmente legítimos.

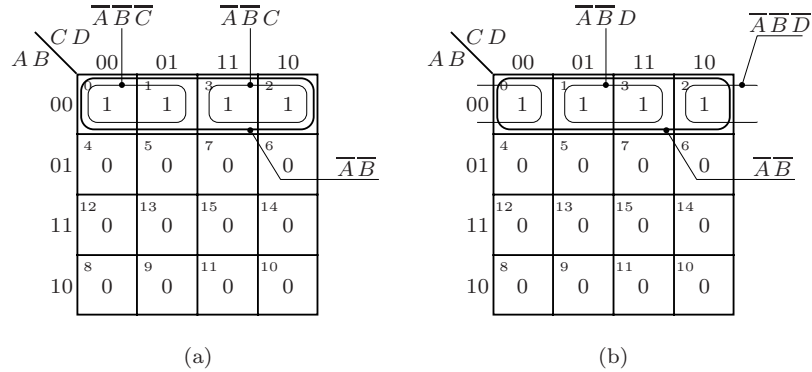


Figura 5.5: (a) e (b) As duas formas que existem para construir o agrupamento legítimo de quatro “1”s correspondente a $m_0 + m_1 + m_2 + m_3$, à custa de dois agrupamentos legítimos de dois “1”s

Na Figura 5.6 apresentam-se as formas alternativas de construção de um outro agrupamento legítimo de quatro “1”s, correspondente a $m_5 + m_7 + m_{13} + m_{15}$, mais uma vez formado à custa de dois agrupamentos de dois “1”s, igualmente legítimos.

Com $m_9 + m_{10} + m_{11} + m_{14}$ o caso é diferente, porque não conseguimos construir o agrupamento de quatro “1”s de forma idêntica à que fizemos anteriormente

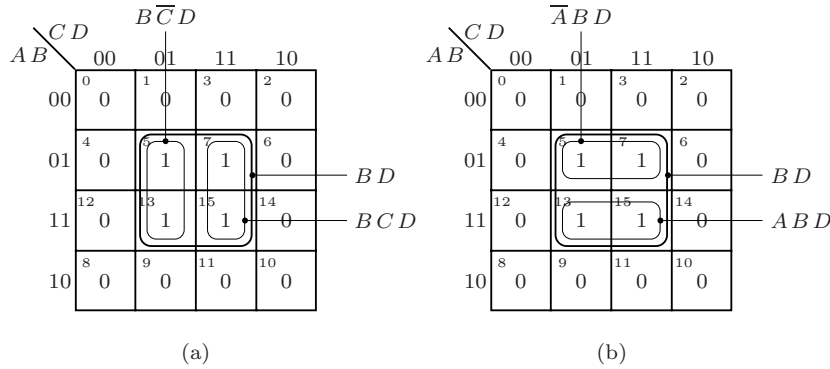


Figura 5.6: Outro agrupamento legítimo de quatro “1”s, correspondente a $m_5 + m_7 + m_{13} + m_{15}$, formado à custa de dois agrupamentos legítimos de dois “1”s

(ou seja, este agrupamento *não é legítimo*). Podemos somar m_9 com m_{11} para eliminar C , mas depois a soma de m_{10} com m_{14} *não elimina o mesmo literal* (elimina B). Ou podemos somar $m_{10} + m_{11}$, mas depois não podemos somar $m_9 + m_{14}$.

Para finalizar, notemos que um agrupamento legítimo de quatro “1”s é construído juntando dois agrupamentos legítimos de dois “1”s que sejam *adjacentes*, como é o caso de $m_0 + m_1$ e $m_2 + m_3$, ou de $m_1 + m_3$ e $m_0 + m_2$. E um agrupamento legítimo de oito “1”s é construído juntando dois agrupamentos legítimos de quatro “1”s que sejam *adjacentes*, um agrupamento legítimo de dezasseis “1”s é construído juntando dois agrupamentos legítimos de oito “1”s que sejam *adjacentes*, etc.

5.18 Minimizar a seguinte função booleana simples:

$$f = \sum m(1, 3, 5, 6, 9, 12, 17, 19, 22, 27, 28, 30) + \sum m_d(4, 11, 14, 20, 21, 25).$$

Resolução: No quadro de Karnaugh da Figura 5.7 admite-se que A é a variável booleana simples com maior peso e E a de menor peso.

5.18

Notemos que $\overline{C}E$ é implicante primo essencial por causa de m_3, m_9, m_{19} e m_{27} , mas não por causa de m_1 ou de m_{17} , já que os “1”s correspondentes a estes dois últimos mintermos podem ser agrupados num outro implicante primo, formado pelos quadrados 1, 5, 17 e 21 (que, iremos constatar, é um implicante primo não essencial que nos vai ser útil).

Notemos ainda que todos os “1”s do implicante primo $C\overline{E}$ o tornam essencial.

Como podemos observar pela Figura 5.7(a), depois de considerados todos os implicantes primos essenciais falta cobrir o mintermo no quadrado 5, o que pode ser feito recorrendo a um de dois implicantes primos não essenciais, representados na Figura 5.7(b), ambos com a mesma complexidade (medida em número de literais).

Notemos que um dos implicantes primos não essenciais, $\overline{B}\overline{D}E$, resulta do agrupamento dos quadrados 1, 5, 17 e 21, tal como mencionámos anteriormente.

		$\overline{C}E$							
		000	001	011	010	110	111	101	100
AB	00	0	1	1	2	1	7	5	1
	01	8	1	11	10	14	15	13	12
11	11	24	25	27	26	30	31	29	28
	10	16	17	19	18	22	23	21	20

(a)

		$\overline{C}E$							
		000	001	011	010	110	111	101	100
AB	00	0	1	3	2	6	7	5	4
	01	8	9	11	10	14	15	13	12
11	11	24	25	27	26	30	31	29	28
	10	16	17	19	18	22	23	21	20

(b)

Figura 5.7: Quadro de Karnaugh da função do Exercício 5.18 com todos os implicantes primos essenciais em (a) e dois implicantes primos não essenciais e de igual complexidade em (b), um dos quais é necessário à soma de produtos mínima da função

Temos, assim, que há duas somas de produtos mínimos para a função F :

$$F = \overline{C}E + C\overline{E} + \overline{B}C\overline{D}$$

$$= \overline{C}E + C\overline{E} + \overline{B}\overline{D}E.$$

5.24 Uma função de 4 variáveis é dada na forma

$$y = (m_1 + m_3 + m_5 + m_9 + m_{10} + m_{11} + m_{12} + m_{14}) (M_8 \cdot M_{10}).$$

O factor $(M_8 \cdot M_{10})$ é necessário para a definição da função, ou não fornece qualquer indicação que não esteja já contida no primeiro factor do produto lógico? Responda referindo-se separadamente aos dois termos máximos que constituem o segundo factor.

5.24

Resolução: A existência do maxtermo M_8 indica que a função vale 0 para a quantidade booleana geral com afixo 8, isto é, $(A, B, C, D) = (1, 0, 0, 0)$, admitindo que A é a variável com maior peso e D a de menor peso. Mas essa informação já estava contida no primeiro factor, $(m_1 + m_3 + m_5 + m_9 + m_{10} + m_{11} + m_{12} + m_{14})$,

na medida em que da lista de mintermos da função não constava m_8 . Logo, esse factor já indicava que a função vale 0 para essa quantidade booleana geral.

A situação é, contudo, diferente no que diz respeito a M_{10} . Com efeito, o primeiro factor menciona a existência de m_{10} , o que significa que a função vale 1 para a quantidade booleana geral com afixo 10, isto é, para $(A, B, C, D) = (1, 0, 1, 0)$. Mas o segundo factor menciona a existência do maxtermo M_{10} , pelo que se pode concluir que, para a mesma quantidade booleana geral, a função vale 0. Claramente, ela não pode valer 1 e 0 para as mesmas combinações de valores nas entradas.

Mas, se notarmos, o segundo factor está a multiplicar logicamente o primeiro factor. Então, para a quantidade booleana geral com afixo 10 o primeiro factor vale 1 e o segundo factor vale 0, pelo que o valor da função que resulta deste produto lógico é 0.

Capítulo 7

Lógica de Polaridade

7.1 Em lógica de polaridade não existem portas NAND e NOR. Existem apenas portas AND, OR, NOT e conversores de polaridade (por vezes também portas XOR, embora estas possam sempre ser compostas por portas dos tipos anteriores). Em contrapartida, a inclusão de indicadores de polaridade permite obter todas as variantes de portas de que necessitamos. Diga que integrados TTL (em lógica positiva) utilizaria para implementar as seguintes portas com 2 entradas, e desenhe os símbolos IEC para cada uma delas:

- a) porta OR com as entradas e a saída activas a L;
- b) porta AND com as entradas e a saída activas a L;
- c) porta OR com as entradas activas a H e a saída activa a L;
- d) porta AND com as entradas activas a H e a saída activa a L;
- e) porta OR com as entradas activas a L e a saída activa a H;
- f) porta AND com as entradas activas a L e a saída activa a H.

Resolução: a) Tratando-se de um OR com 2 entradas, ele deverá ter a tabela de verdade genérica de um OR, com a saída activa desde que pelo menos uma das entradas esteja activa (Tabela 7.1).

7.1 a)

Tabela 7.1: Tabela de verdade genérica para uma porta OR com 2 entradas

A	B	A + B
I	I	I
I	A	A
A	I	A
A	A	A

Sabendo os níveis de actividade das entradas e da saída da porta, podemos deduzir imediatamente a tabela de verdade física do OR (Figura 7.1).

Com a tabela de verdade física do OR podemos agora obter a tabela de verdade em lógica positiva, para podermos determinar a porta equivalente em TTL. Essa tabela é a que se apresenta na Figura 7.2.

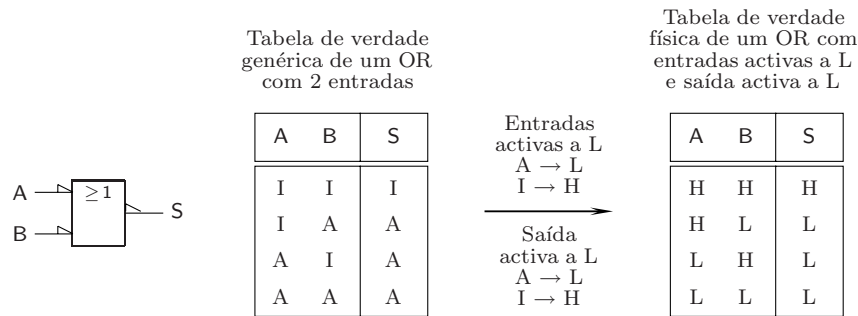


Figura 7.1: Símbolo IEC de um OR com 2 entradas activas a L e a saída activa a L, e correspondentes tabelas de verdade genérica e física

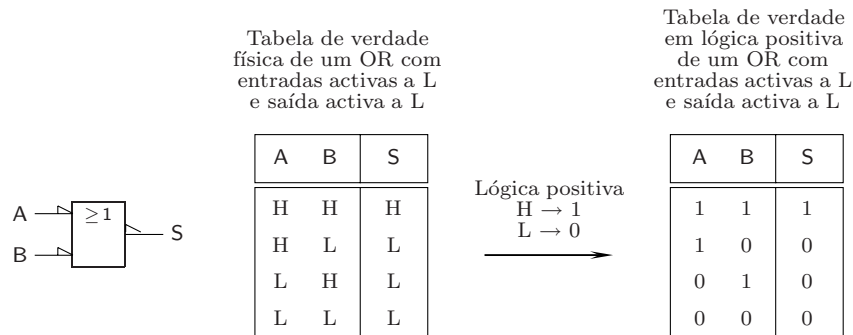


Figura 7.2: Símbolo IEC de um OR com 2 entradas activas a L e a saída activa a L, e correspondentes tabelas de verdade física e em lógica positiva

Estamos, claramente, em presença de um AND em lógica positiva, ou seja, de um 74x08 em TTL. Logo, os símbolos alternativos desta porta, em lógica de polaridade e em lógica positiva (como sabemos, as duas lógicas têm símbolos iguais), são:

$$\begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\geq 1} \rightarrow S = \overline{\overline{A} + \overline{B}} \quad \equiv \quad \begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\&} \rightarrow S = AB$$

Deve-se chamar a atenção que este era o resultado esperado já que, em lógica positiva, as expressões das funções às saídas dos símbolos são iguais (obtem-se uma da outra por utilização das leis de De Morgan).

Notemos, por outro lado, que os símbolos IEC podem ser gerados um a partir do outro trocando o OR pelo AND e trocando ainda as posições dos indicadores de polaridade.

7.1 b)

b) Repete-se a alínea anterior, mas agora para um AND com 2 entradas activas a L e saída activa a L. A tabela de verdade genérica de um AND com 2 entradas encontra-se na Tabela 7.2.

Sabendo os níveis de actividade das entradas e da saída da porta, podemos deduzir a tabela de verdade física do AND com entradas e saída activas a L (Figura 7.3).

Tabela 7.2: Tabela de verdade genérica para uma porta AND com 2 entradas

A	B	AB
I	I	I
I	A	I
A	I	I
A	A	A

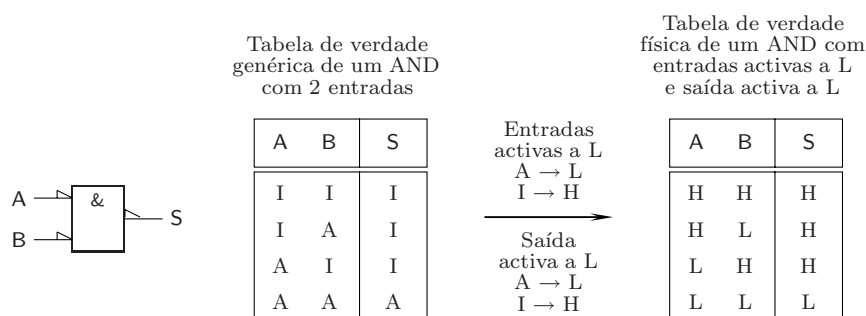


Figura 7.3: Símbolo IEC de um AND com 2 entradas activas a L e a saída activa a L, e correspondentes tabelas de verdade genérica e física

Com a tabela de verdade física do AND, podemos em seguida obter a tabela de verdade em lógica positiva, na Figura 7.4.

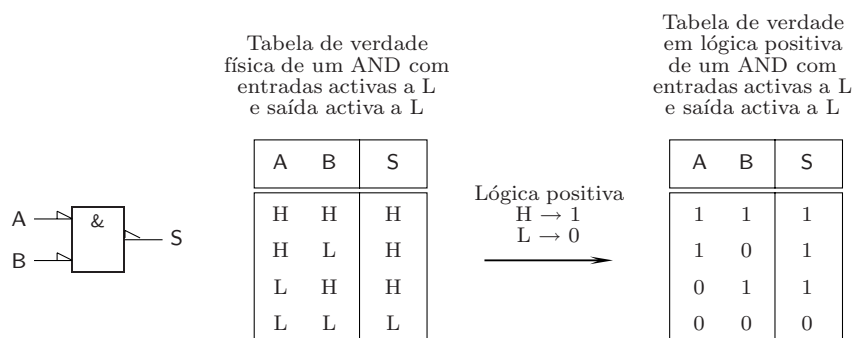


Figura 7.4: Símbolo IEC de um AND com 2 entradas activas a L e a saída activa a L, e correspondentes tabelas de verdade física e em lógica positiva

Estamos, agora, em presença de um OR em lógica positiva, ou seja, de um 74x32 em TTL. Logo, os símbolos alternativos desta porta, em lógica de polaridade e em lógica positiva, são:

$$\begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\&} \rightarrow S = \overline{\overline{AB}} \quad \equiv \quad \begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\geq 1} \rightarrow S = A + B$$

Mais uma vez, este resultado era esperado já que, em lógica positiva, as expressões das funções às saídas dos símbolos são iguais (obtem-se uma da outra

por utilização das leis de De Morgan).

Notemos ainda que os símbolos IEC podem ser gerados um a partir do outro trocando o OR com o AND e trocando as posições dos indicadores de polaridade.

7.1 c)

c) A tabela de verdade genérica de um OR com 2 entradas já é conhecida da Tabela 7.1. Vem, contudo, repetida na Figura 7.5 para facilitar a exposição, conjuntamente como as correspondentes tabelas de verdade física e em lógica positiva no caso em que as entradas são activas a H e a saída é activa a L.

Tabela de verdade genérica de um OR		Tabela de verdade física de um OR com entradas activas a H e saída activa a L		Tabela de verdade em lógica positiva de um OR com entradas activas a H e saída activa a L																																													
<table border="1"> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> <tr> <td>I</td> <td>I</td> <td>I</td> </tr> <tr> <td>I</td> <td>A</td> <td>A</td> </tr> <tr> <td>A</td> <td>I</td> <td>A</td> </tr> <tr> <td>A</td> <td>a</td> <td>A</td> </tr> </table>	A	B	S	I	I	I	I	A	A	A	I	A	A	a	A	<p>Entradas activas a H $A \rightarrow H$ $I \rightarrow L$</p> <p>Saída activa a L $A \rightarrow L$ $I \rightarrow H$</p>	<table border="1"> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> <tr> <td>L</td> <td>L</td> <td>H</td> </tr> <tr> <td>L</td> <td>H</td> <td>L</td> </tr> <tr> <td>H</td> <td>L</td> <td>L</td> </tr> <tr> <td>H</td> <td>H</td> <td>L</td> </tr> </table>	A	B	S	L	L	H	L	H	L	H	L	L	H	H	L	<p>Lógica positiva $H \rightarrow 1$ $L \rightarrow 0$</p>	<table border="1"> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0
A	B	S																																															
I	I	I																																															
I	A	A																																															
A	I	A																																															
A	a	A																																															
A	B	S																																															
L	L	H																																															
L	H	L																																															
H	L	L																																															
H	H	L																																															
A	B	S																																															
0	0	1																																															
0	1	0																																															
1	0	0																																															
1	1	0																																															

Figura 7.5: Tabelas de verdade genérica, física e em lógica positiva de um OR com 2 entradas activas a H e saída activa a L

74x02

Estamos, agora, em presença de um NOR em lógica positiva, ou seja, de um 74x02 em TTL. Embora o símbolo do OR seja evidente, já não o é o símbolo alternativo, em lógica de polaridade e em lógica positiva. Vamos ter de esperar pela resolução da alínea f) e observar a tabela de verdade em lógica positiva de um AND com entradas activas a L e saída activa a H para concluir que os símbolos alternativos para o OR são:

$$\begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\geq 1} \rightarrow S = \overline{A+B} \quad \equiv \quad \begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\&} \rightarrow S = \overline{A \cdot B}$$

Porém, mais uma vez este resultado era esperado, porque em lógica positiva as expressões das funções às saídas dos símbolos são iguais, por utilização das leis de De Morgan.

Notemos ainda que os símbolos IEC podem ser gerados um a partir do outro trocando o OR com o AND e trocando as posições dos indicadores de polaridade.

7.1 d)

d) A tabela de verdade genérica de um AND com 2 entradas já é conhecida da Tabela 7.2. Vamos, contudo, repeti-la na Figura 7.6 para facilitar a exposição, conjuntamente como as correspondentes tabelas de verdade física e em lógica positiva no caso em que as entradas são activas a H e a saída é activa a L.

74x00

Temos, agora, um NAND em lógica positiva, ou seja, um 74x00 em TTL. Embora o símbolo do AND seja evidente, já não o é o símbolo alternativo, em lógica de polaridade e em lógica positiva. Vamos ter de esperar pela resolução da alínea e) e observar a tabela de verdade em lógica positiva de um OR com entradas

Tabela de verdade genérica de um AND			Tabela de verdade física de um AND com entradas activas a H e saída activa a L			Tabela de verdade em lógica positiva de um AND com entradas activas a H e saída activa a L		
A	B	S	Entradas activas a H $A \rightarrow H$ $I \rightarrow L$			Lógica positiva $H \rightarrow 1$ $L \rightarrow 0$		
I	I	I	L	L	H	0	0	1
I	A	I	L	H	H	0	1	1
A	I	I	H	L	H	1	0	1
A	A	A	H	H	L	1	1	0

Figura 7.6: Tabelas de verdade genérica, física e em lógica positiva de um AND com 2 entradas activas a H e saída activa a L

activas a L e saída activa a H para concluir que os símbolos alternativos para o AND são:

$$\begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\&} \rightarrow S = \overline{AB} \quad \equiv \quad \begin{array}{c} A \\ B \end{array} \rightarrow \boxed{\geq 1} \rightarrow S = \overline{A} + \overline{B}$$

Mais uma vez, estávamos à espera deste resultado porque, em lógica positiva, as expressões das funções às saídas dos símbolos são iguais (leis de De Morgan).

Notemos ainda que os símbolos IEC podem ser gerados um a partir do outro trocando o OR com o AND e as posições dos indicadores de polaridade.

e) Pelo que foi afirmado no fim da alínea d) deste exercício, o OR com 2 entradas activas a L e saída activa a H deve comportar-se como um AND com 2 entradas activas a H e saída activa a L (ou seja, um NAND em lógica positiva). Vamos prová-lo.

7.1 e)

Na Figura 7.7 apresenta-se a tabela de verdade genérica de um OR com 2 entradas, conjuntamente como as correspondentes tabelas de verdade física e em lógica positiva no caso em que as entradas são activas a L e a saída é activa a H.

Tabela de verdade genérica de um OR			Tabela de verdade física de um OR com entradas activas a L e saída activa a H			Tabela de verdade em lógica positiva de um OR com entradas activas a L e saída activa a H		
A	B	S	Entradas activas a L $A \rightarrow L$ $I \rightarrow H$			Lógica positiva $H \rightarrow 1$ $L \rightarrow 0$		
I	I	I	H	H	L	1	1	0
I	A	A	H	L	H	1	0	1
A	I	A	L	H	H	0	1	1
A	A	A	L	L	H	0	0	1

Figura 7.7: Tabelas de verdade genérica, física e em lógica positiva de um OR com 2 entradas activas a L e saída activa a H

Como esperávamos, obtemos um NAND em lógica positiva, ou seja, um 74x00 em TTL (basta comparar as tabelas de verdade em lógica positiva das Figuras 7.6 e 7.7). Logo, um OR com 2 entradas activas a L e saída activa a H tem os símbolos alternativos da alínea e) deste exercício.

Notemos, mais uma vez, que os símbolos IEC podem ser gerados um a partir do outro trocando o OR com o AND e trocando ainda as posições dos indicadores de polaridade.

7.1 f)

f) Pelo que sabemos da alínea c) deste exercício, um AND com 2 entradas activas a L e saída activa a H deve comportar-se como um OR com 2 entradas activas a H e saída activa a L (um NOR em lógica positiva). Vamos prová-lo.

Na Figura 7.8 ilustra-se a tabela de verdade genérica de um AND com 2 entradas, conjuntamente como as correspondentes tabelas de verdade física e em lógica positiva no caso em que as entradas são activas a L e a saída é activa a H.

Tabela de verdade genérica de um AND			Tabela de verdade física de um OR com entradas activas a L e saída activa a H			Tabela de verdade em lógica positiva de um OR com entradas activas a L e saída activa a H				
A	B	S	Entradas activas a L $A \rightarrow L$ $I \rightarrow H$ Saída activa a H $A \rightarrow H$ $I \rightarrow L$	A	B	S	Lógica positiva $H \rightarrow 1$ $L \rightarrow 0$	A	B	S
I	I	I		H	H	L		1	1	0
I	A	I		H	L	L		1	0	0
A	I	I		L	H	L		0	1	0
A	A	A		L	L	H		0	0	1

Figura 7.8: Tabelas de verdade genérica, física e em lógica positiva de um AND com 2 entradas activas a L e saída activa a H

Como esperávamos, obtemos um NOR em lógica positiva, ou seja, um 74x02 em TTL (basta comparar as tabelas de verdade em lógica positiva das Figuras 7.5 e 7.8). Logo, um AND com 2 entradas activas a L e saída activa a H tem os símbolos alternativos da alínea c) deste exercício.

Mais uma vez, os símbolos IEC podem ser gerados um a partir do outro trocando o OR com o AND e trocando as posições dos indicadores de polaridade.

7.2 Repita o exercício anterior para as seguintes portas com 3 entradas (possivelmente necessitará de circuitos mais complexos do que uma simples porta de substituição):

- porta OR com 2 entradas activas a H e uma a L, e a saída activa a H;
- porta OR com 2 entradas activas a H e uma a L, e a saída activa a L;
- porta AND com 2 entradas activas a H e uma a L, e a saída activa a H;
- porta AND com 2 entradas activas a H e uma a L, e a saída activa a L;
- porta XOR com as 3 entradas activas a H e a saída activa a H;
- porta XOR com as 3 entradas activas a L e a saída activa a H.

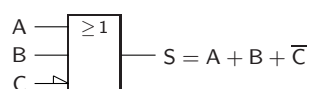
7.2 a)

Resolução: a) Esperamos agora obter, para cada uma das portas dadas neste exercício, circuitos mais complexos em lógica positiva (TTL), formados por

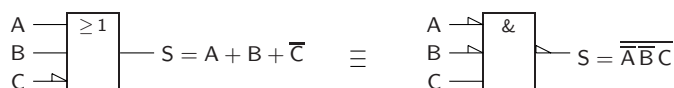
interligações de diversas portas mais simples (relembrar que em TTL, como também noutras tecnologias, não encontramos portas com níveis de actividade *diferentes* nas entradas).

Do que sabemos do texto das aulas teóricas e do Exercício 7.1, podemos de imediato deduzir os símbolos equivalentes e alternativos para as portas dadas, já que eles devem gerar expressões booleanas equivalentes para a função de saída e essas expressões devem poder ser obtidas uma da outra por aplicação das leis de De Morgan.

Por exemplo, para a porta OR desta alínea, com 2 entradas activas a H e uma a L e a saída activa a H, devemos obter a expressão $S = A + B + \overline{C}$ para o símbolo



Quanto ao símbolo equivalente, deve gerar a função $S = \overline{\overline{A} \overline{B} C}$. Logo, os símbolos equivalentes para a porta devem ser os que se seguem,



sendo gerados um a partir do outro substituindo o OR por um AND e trocando as posições dos indicadores de polaridade, tal como já tínhamos constatado para as situações mais simples do Exercício 7.1.

Iremos, contudo, provar esta manipulação simbólica, desenhando os logigramas dos circuitos equivalentes em TTL nos dois casos e constatando que são equivalentes.

Consideremos, então as tabelas de verdade genérica, física e em lógica positiva desta porta, nas Figuras 7.9 e 7.10.

Da tabela de verdade lógica deduzimos, com efeito, as equações

$$\begin{aligned} S = M_0 &= A + B + \overline{C} = \\ &= \overline{m_0} = \overline{\overline{A} \overline{B} C}. \end{aligned}$$

Os esquemas eléctricos em TTL são, então, os que se representam na Figura 7.11. De notar as portas NOT, do tipo 74x04, e a porta NAND de 3 entradas, do tipo 74x10, ambas em tecnologia TTL.

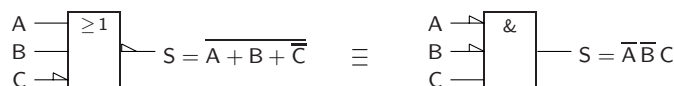
74x04

74x10

De notar que podíamos ter gerado outros esquemas. Contudo, o que se apresenta tem em consideração as portas integradas existentes em TTL, e os circuitos formados têm apenas 2 níveis.

b) Podemos de imediato desenhar os símbolos alternativos para a porta dada,

7.2 b)



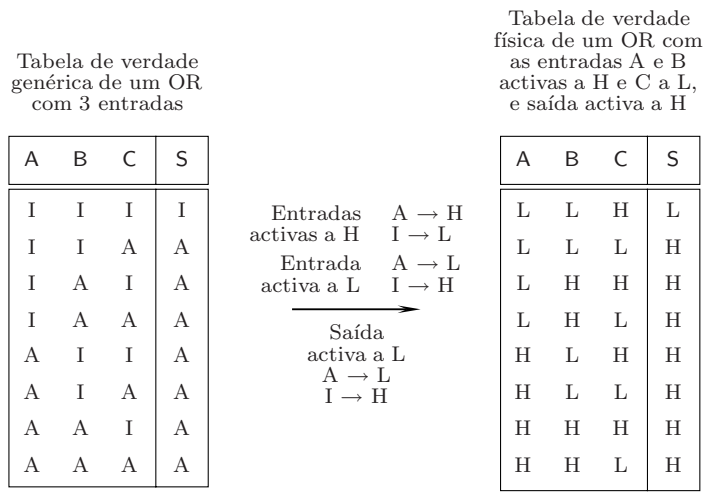


Figura 7.9: Tabelas de verdade genérica e física de um OR com 2 entradas activas a H e uma a L, e saída activa a H

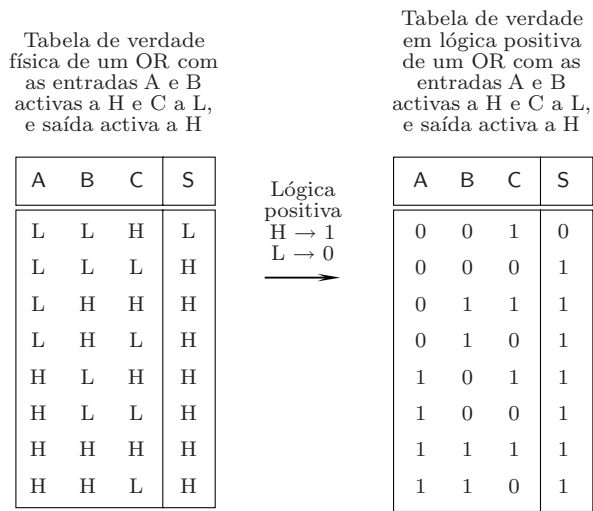


Figura 7.10: Tabelas de verdade e física e em lógica positiva de um OR com 2 entradas activas a H e uma a L, e saída activa a H

gerados um a partir do outro substituindo o OR por um AND e trocando as posições dos indicadores de polaridade.

Quanto aos circuitos equivalentes em TTL, são os da Figura 7.12. De notar a porta AND de 3 entradas em tecnologia TTL, do tipo 74x11.

Tabela de verdade genérica das portas XOR com 2 entradas

7.3 A descrição da função OU-exclusivo em lógica de polaridade traduz-se pela tabela de verdade genérica da Tabela 7.12 (de SD:AAT) para uma porta XOR com 2 entradas.

a) Desenhe o símbolo IEC desta porta que tenha as entradas activas a L e a

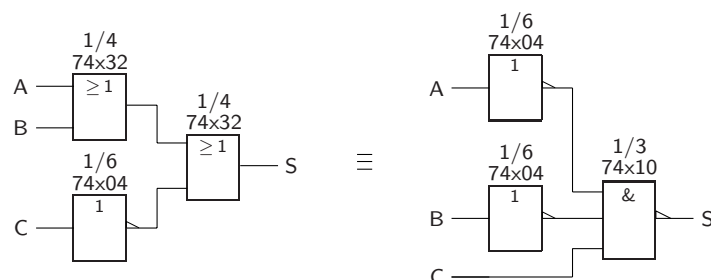


Figura 7.11: Esquemas eléctricos em TTL equivalentes à porta OR com duas entradas activas a H e uma a L, e saída activa a H

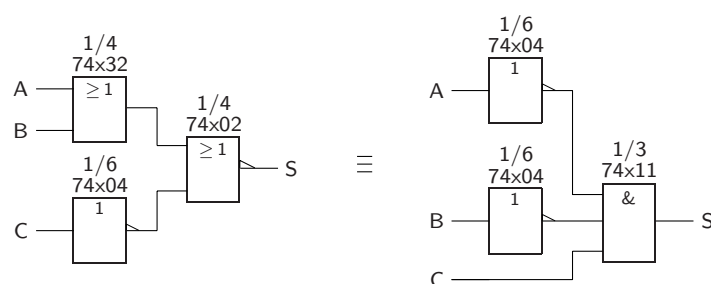


Figura 7.12: Esquemas eléctricos em TTL equivalentes à porta OR com duas entradas activas a H e uma a L, e saída activa a L

saída activa a H.

b) Repita a alínea anterior para o caso de a entrada A ser activa a L, B ser activa a H e a saída ser activa a L.

c) Mostre que as portas das alíneas a) e b) são fisicamente apenas uma, e ainda que são idênticas a um XOR com entradas activas a H e a saída activa a H.

Resolução: a) Para o símbolo deste XOR *vd.* a Figura 7.13. E dada a tabela de verdade genérica para esta porta, podemos deduzir imediatamente a correspondente tabela de verdade física (na mesma figura).

7.3 a)

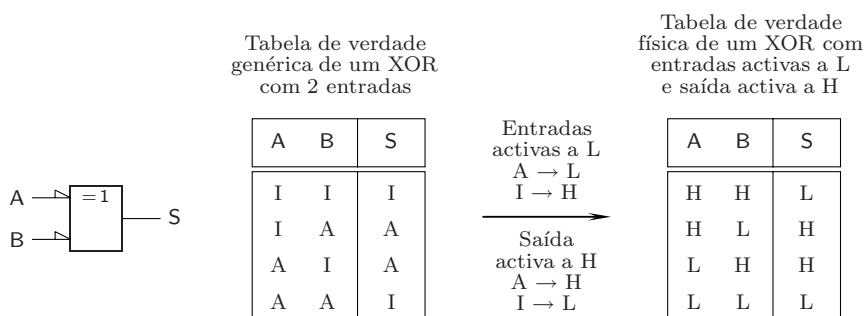


Figura 7.13: Símbolo IEC de um XOR com 2 entradas activas a L e a saída activa a H, e correspondentes tabelas de verdade genérica e física

b) Para o símbolo deste XOR *vd.* a Figura 7.14. E da tabela de verdade genérica

7.3 b)

para esta porta podemos deduzir imediatamente a correspondente tabela de verdade física (na mesma figura).

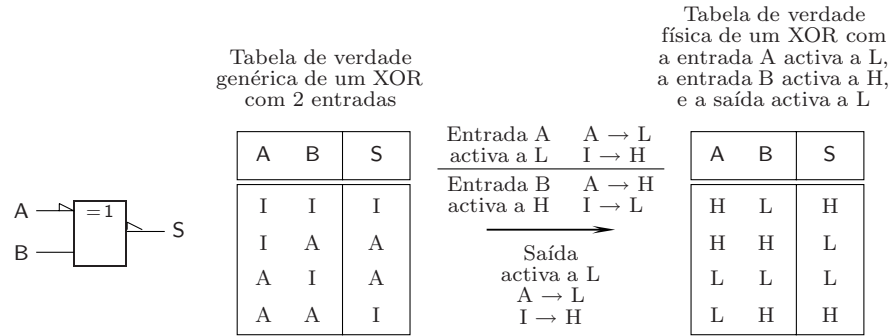


Figura 7.14: Símbolo de um XOR com a entrada A activa a L, a entrada B activa a H e a saída activa a L, e correspondentes tabelas de verdade genérica e física

7.3 c)

c) Para o símbolo deste XOR *vd.* a Figura 7.15.

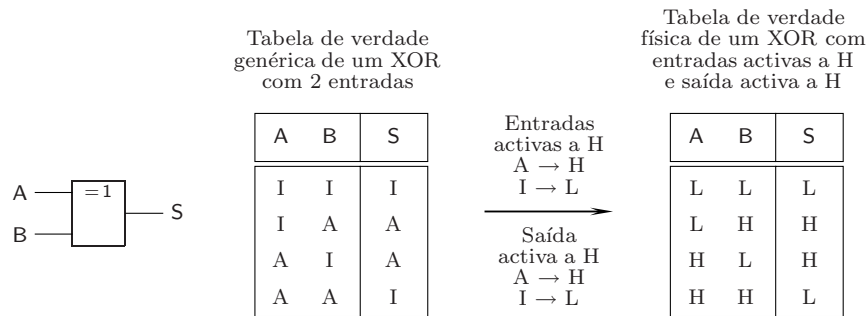


Figura 7.15: Símbolo de um XOR com as entradas activas a H e a saída activa a H, e correspondentes tabelas de verdade genérica e física

Da tabela de verdade genérica para esta porta podemos deduzir imediatamente a correspondente tabela de verdade física (na mesma figura).

Se agora compararmos as tabelas de verdade físicas dos três XORs, constatamos que são iguais. Tal resulta das seguintes equivalências algébricas:

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

que, por sua vez, são consequência de

$$\overline{X \oplus Y} = \overline{X} \oplus Y = X \oplus \overline{Y}.$$

7.4 Pretende-se implementar a função booleana simples $OUT = IN1 + IN2$ admitindo que:

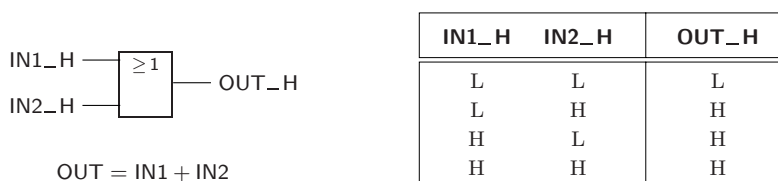
a) os níveis de actividade de $IN1$, de $IN2$ e de OUT são todos H;

- b) $IN1$ é activa a L e $IN2$ e OUT são activas a H;
 c) $IN1$ e $IN2$ são activas a L e OUT é activa a H;
 d) $IN1$ e $IN2$ são activas a H e OUT é activa a L.

Estabelecer, para todos os casos, a tabela de verdade de OUT .

Resolução: a) Como as variáveis de entrada e a função de saída são todas activas a H, resulta imediatamente o logigrama e a tabela de verdade física da Figura 7.16(a) e (b), respectivamente.

7.4 a)



(a)

(b)

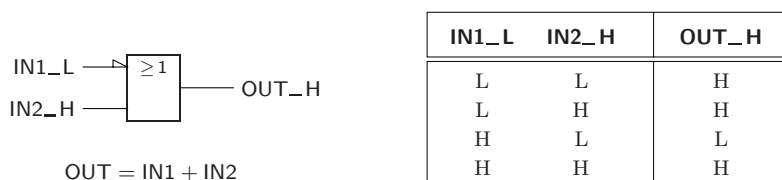
Figura 7.16: Logigrama (a) e tabela de verdade física (b) da função booleana simples $OUT = IN1 + IN2$, quando $IN1$, $IN2$ e OUT são activas a H

A tabela de verdade, vem construída da seguinte forma: (i) na primeira linha $IN1_H$ e $IN2_H$ estão a L — ou seja, as variáveis booleanas simples $IN1$ e $IN2$ estão inactivas — pelo que as duas entradas do OR estão inactivas, o que faz com que a saída do OR deva estar inactiva, isto é, a L; (ii) nas restantes linhas da tabela, $IN1_H$ ou $IN2_H$ ou ambas estão a H, pelo que a entrada correspondente está activa, o que é suficiente para que a saída do OR esteja activa, isto é, a H.

O logigrama, por sua vez, é construído no pressuposto que os níveis de actividade da porta coincidem com os níveis de actividade das variáveis de entrada e da função de saída, o que permite a sua correcta interpretação, como vimos no texto teórico.

- b) Como agora $IN1$ é activa a L e $IN2$ e OUT são activas a H, resulta imediatamente o logigrama e a tabela de verdade física da Figura 7.17(a) e (b), respectivamente.

7.4 b)



(a)

(b)

Figura 7.17: Logigrama (a) e tabela de verdade física (b) da função booleana simples $OUT = IN1 + IN2$, quando $IN1$ é activa a L e $IN2$ e OUT são activas a H

Mais uma vez, o logigrama é construído por forma a que os níveis de actividade das variáveis de entrada e da função de saída coincidam com os níveis de

actividade nas entradas e na saída da porta, respectivamente.

Notemos que a função $OUT = IN1 + IN2$ é construída à custa de uma porta OR, *independentemente dos níveis de actividade nas entradas e na saída da porta*. Naturalmente, trata-se de uma porta OR diferente da anterior (é apenas uma das $2^3 = 8$ portas OR com duas entradas que podemos construir), com uma tabela de verdade diferente, mas igualmente representativa da função booleana simples $OUT = IN1 + IN2$.

Quanto à tabela de verdade, vem agora construída da seguinte forma: (i) só para a linha em que $IN1_L$ está a H e $IN2_H$ está a L é que as duas entradas do OR estão inactivas, o que faz com que a saída venha inactiva, isto é, a L; (ii) para todas as outras linhas pelo menos uma das entradas está activa, o que é suficiente para que a saída do OR esteja activa, isto é, a H.

7.4 c)

c) Como agora $IN1$ e $IN2$ são activas a L e OUT é activa a H, resulta imediatamente o logigrama e a tabela de verdade física da Figura 7.18(a) e (b), respectivamente.

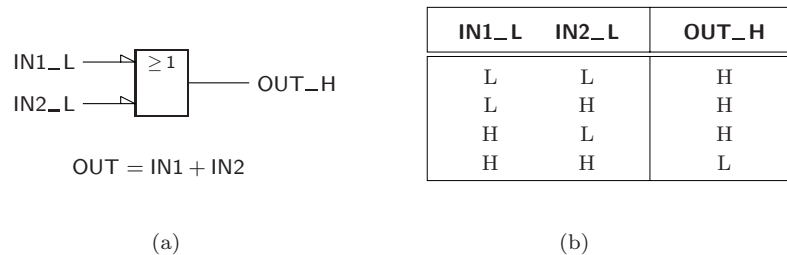


Figura 7.18: Logigrama (a) e tabela de verdade física (b) da função booleana simples $OUT = IN1 + IN2$, quando $IN1$ e $IN2$ são activas a L e OUT é activa a H

O logigrama é, ainda neste caso, construído por forma a que os níveis de actividade das variáveis de entrada e da função de saída coincidam com os níveis de actividade nas entradas e na saída da porta, respectivamente.

Quanto à função $OUT = IN1 + IN2$, vem mais uma vez construída com uma porta OR, com níveis de actividade nas entradas e saída que são, naturalmente, diferentes dos dos dois casos anteriores. A construção da tabela de verdade deve, mais uma vez, ser trivial. Menciona-se, como exemplo, que apenas para a linha $(IN1, IN2) = (H, H)$ é que as duas entradas do OR estão inactivas, o que faz com que a saída venha inactiva.

7.4 d)

d) Finalmente, consideremos o caso em que $IN1$ e $IN2$ são activas a H e OUT é activa a L. Temos, nesta situação, o logigrama e a tabela de verdade física da Figura 7.19.

Notemos, mais uma vez, que a função $OUT = IN1 + IN2$ vem construída com uma porta OR, embora com níveis de actividade nas entradas e saída que são, naturalmente, diferentes dos dos três casos anteriores. Quanto à construção da tabela de verdade, nada de relevante há a dizer.



Este exercício chama a atenção para o seguinte facto: *uma função booleana simples pode ser representada por 2^n logigramas distintos em lógica de polaridade, um por cada conjunto de níveis de actividade possíveis para as variáveis de entrada, função de saída, e funções intermédias (no total, em número de n).*

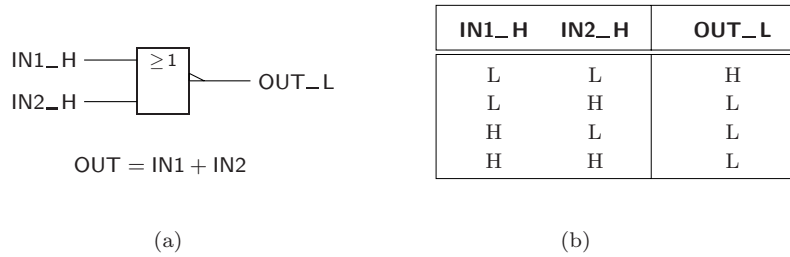


Figura 7.19: Logigrama (a) e tabela de verdade física (b) da função booleana simples $OUT = IN1 + IN2$, quando $IN1$ e $IN2$ são activas a H e OUT é activa a L

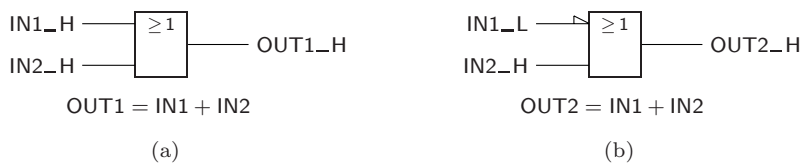
Em particular, para uma função de 2 variáveis representada por uma única porta lógica em lógica de polaridade (que não possui, por conseguinte, funções intermédias), como é o caso da função $OUT = IN1 + IN2$ deste exercício, podem-se construir $2^3 = 8$ logigramas distintos (no exercício apenas se estudaram 4 deles).

Deve, contudo, ressaltar-se que os 8 logigramas distintos *em lógica de polaridade* que representam a função booleana simples $OUT = IN1 + IN2$, quando interpretados em *lógica positiva* ou em *lógica negativa* vão representar 8 funções distintas, nomeadamente $OUT = IN1 + IN2$, $OUT = \overline{IN1} + IN2$, $OUT = IN1 + \overline{IN2}$, \dots , $\overline{OUT} = \overline{IN1} + \overline{IN2}$.

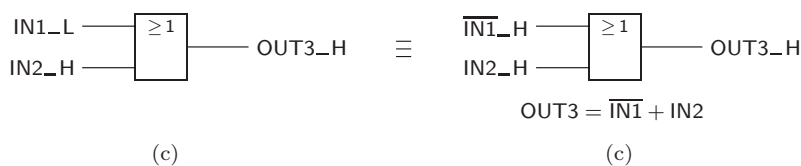
7.5 Que função ou funções booleanas simples são geradas pelas portas lógicas da Figura 7.30 (de *SD:AAT*)?

Resolução: Na Figura 7.30(a) e (b) de *SD:AAT* os níveis de actividade das variáveis de entrada e da função de saída coincidem, respectivamente, com os níveis de actividade das entradas e da saída da porta. Logo, podemos interpretar directamente a função representada como $OUT1 = OUT2 = IN1 + IN2$:

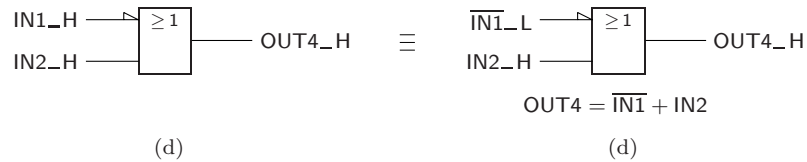
7.5



Na Figura 7.30(c) de *SD:AAT* o nível de actividade da variável de entrada $IN1$ não coincide com o nível de actividade da entrada correspondente da porta. Porém, $IN2$ e OUT têm níveis de actividade coincidentes com os da porta. Logo, devemos modificar a variável de entrada $IN1$ e o correspondente nível de actividade, como se indica a seguir:



Na Figura 7.30(d) de *SD:AAT*, mais uma vez, o nível de actividade da variável de entrada $IN1$ é o único que *não coincide* com o nível de actividade da entrada correspondente da porta. Logo, devemos modificar a variável de entrada $IN1$ e o correspondente nível de actividade, como se indica a seguir:



Ou seja, concluímos que $OUT4 = OUT3 = \overline{IN1} + IN2$.

Este exercício chama a atenção para dois factos.

O primeiro já foi salientado no Exercício 7.4 e foi traduzido pela seguinte afirmação: *uma função booleana simples pode ser representada por 2^n logigramas distintos em lógica de polaridade, um por cada conjunto de níveis de actividade possíveis para as variáveis de entrada, função de saída, e funções intermédias (no total, em número de n).*

Por exemplo, a função $OUT = IN1 + IN2$ tem $2^3 = 8$ logigramas possíveis, dos quais apenas se apresentam dois nas Figuras 7.30(a) e (b) de *SD:AAT*. Outro tanto acontece com a função $OUT = \overline{IN1} + IN2$, que também tem 8 logigramas, apenas se apresentando dois deles nas Figuras 7.30(c) e (d).



O segundo facto é o oposto do anterior, e pode ser traduzido pela seguinte afirmação: *um logigrama em lógica de polaridade representa 2^m funções distintas, uma por cada conjunto de níveis de actividade possíveis para as variáveis de entrada, função de saída, e funções intermédias (no total, em número de m).*

Em particular, um logigrama formado por uma única porta com 2 entradas (portanto, sem funções intermédias) representa $2^3 = 8$ funções distintas (é o caso do presente exercício, mas em que apenas se apresentam duas das 8 funções possíveis).

Com efeito, se compararmos as Figuras 7.30(a) e (c) de *SD:AAT* concluímos que o mesmo logigrama representa duas funções distintas, em particular $OUT1 = IN1 + IN2$ e $OUT3 = \overline{IN1} + IN2$. Outro tanto se pode concluir se compararmos as Figuras 7.30(b) e (d), que ilustram o mesmo logigrama para 2 funções diferentes, $OUT2 = IN1 + IN2$ e $OUT4 = \overline{IN1} + IN2$.

7.6 Dada a porta da Figura 7.31 (de *SD:AAT*), determinar a expressão lógica da função de saída e construir a tabela de verdade física correspondente.

7.6

Resolução: O logigrama dado é constituído por um OR com níveis de actividade nas entradas e na saída que não coincidem com os níveis de actividade das variáveis de entrada e da função de saída. Nestas condições, torna-se difícil determinar a expressão lógica da função.

Vamos, então, ajustar os níveis de actividade das variáveis e da função aos níveis de actividade das entradas e da saída da porta, com o intuito de simplificar a obtenção da expressão de OUT . Para tanto, iremos modificar as variáveis e a

função, atendendo a que $VAR_L = \overline{VAR_H}$ e a que $VAR_H = \overline{VAR_L}$, para uma variável ou função VAR qualquer.

Obtemos, assim, o logigrama colocado à direita na Figura 7.20.

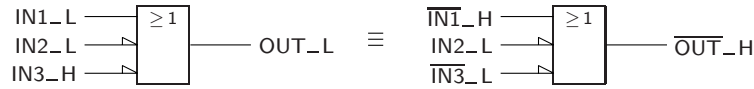


Figura 7.20: Logigrama da função booleana simples do Exercício 7.6, modificado para fazer coincidir os níveis de actividade nas entradas e na saída do OR com os níveis de actividade das variáveis de entrada e da função de saída, respectivamente. O objectivo desta transformação consiste em facilitar a obtenção da expressão lógica para a função OUT

Deste logigrama podemos agora deduzir que a função pretendida tem por expressão $\overline{OUT} = \overline{IN1} + IN2 + \overline{IN3}$ ou, se quisermos, $OUT = IN1 \cdot \overline{IN2} \cdot IN3$.

Vamos escrever de seguida a tabela de verdade física para o circuito dado. Dado serem completamente equivalentes, podemos raciocinar da mesma forma sobre o logigrama dado (à esquerda da Figura 7.20) ou sobre o logigrama modificado (à direita da figura). Vamos fazê-lo, arbitrariamente, sobre este último.

A tabela de verdade pode ser deduzida linha a linha, atendendo aos níveis de actividade às entradas e saídas da porta, tal como foi feito em situações anteriores. Para este caso temos três colunas de entrada e uma de saída, designadas, respectivamente, por $\overline{IN1_H}$, por $IN2_L$, por $\overline{IN3_L}$ e por $\overline{OUT_H}$ (Tabela 7.3).

Tabela 7.3: Tabela de verdade física para a função \overline{OUT}

	$\overline{IN1_H}$	$IN2_L$	$\overline{IN3_L}$	$\overline{OUT_H}$
	L	L	L	H
	L	L	H	H
	L	H	L	H
	L	H	H	L
	H	L	L	H
	H	L	H	H
	H	H	L	H
	H	H	H	H

A única combinação de tensões eléctricas nas entradas que desactiva \overline{OUT} →

Mas também podemos obter a tabela de uma forma mais expedita, pensando da seguinte forma: desde que pelo menos uma das entradas do OR esteja activa, a saída vem activada (a H). Ora isso acontece em todas as linhas da tabela, excepto na linha 3, em que a saída vem inactiva (a L) porque todas as entradas estão inactivas.

Vamos agora verificar, pela tabela obtida, que a expressão da função é, verdadeiramente, $OUT = IN1 \cdot \overline{IN2} \cdot IN3$. Para tanto concentremos a nossa atenção na linha referenciada (linha 3). A função OUT , que é activa a L, vem activada nessa e só nessa linha. Então, em vez de designarmos a coluna da função por $\overline{OUT_H}$, vamos redesigná-la por OUT_L .

De forma semelhante, vamos procurar as designações alternativas para as variáveis de entrada e respectivos níveis de actividade que façam salientar as condições em que essas variáveis vêm activas. Por exemplo, em vez de $\overline{IN1_H}$ escrevemos $IN1_L$ na primeira coluna, pelo facto de, na linha 3, termos um L nessa entrada, o que activa a variável $IN1$. Queremos com isso significar que, para essa linha, é a variável $IN1$ que vem activa, e não a variável $\overline{IN1}$. E de idêntica forma procedemos com as restantes entradas.

Obtemos, assim, a Tabela 7.4, que mais não é do que a tabela anterior com as redesignações apropriadas.

Tabela 7.4: Tabela de verdade física para a função $OUT = IN1 \cdot \overline{IN2} \cdot IN3$

Designações antigas	$\overline{IN1_H}$	$IN2_L$	$\overline{IN3_L}$	$\overline{OUT_H}$
	$IN1_L$	$\overline{IN2_H}$	$IN3_H$	$OUT_L = (IN1 \cdot \overline{IN2} \cdot IN3)_L$
A única combinação de tensões eléctricas nas entradas que activa OUT	L	L	L	H
	L	L	H	H
	L	H	L	H
	L	H	H	L
	H	L	L	H
	H	L	H	H
	H	H	L	H
	H	H	H	H

E agora reparemos quando é que a função OUT vem activa: quando $IN1$ está activa e quando $\overline{IN2}$ está activa e quando $IN3$ está activa. Então OUT deve ser o produto lógico destas três variáveis, como já se sabia:

$$OUT = IN1 \cdot \overline{IN2} \cdot IN3 = m_5,$$

se $IN1$ for a variável de maior peso e $IN3$ a de menor peso.

Mais uma vez, e como se afirmou no texto teórico, o índice do mintermo obtido para a função não tem que vir coincidente com o número da linha em que se situa: o índice é igual a 5 e a linha é a 3. Tal deve-se ao facto de as variáveis não serem todas activas a H.

7.7 Obter, em lógica HCT, os esquemas eléctricos correspondentes aos logigramas dos Exercícios 7.4 e 7.6 (de *SD:AAT*). Para simplificar os esquemas, não incluir os pinos dos circuitos integrados.

7.7

Resolução: Começamos por considerar os logigramas do Exercício 7.4. O logigrama da Figura 7.16(a) coincide com o seu esquema eléctrico, como se ilustra na Figura 7.21(a).

Já o logigrama da Figura 7.17(a), que usa uma porta com níveis de actividade diferentes nas entradas, terá de vir modificado. A modificação consiste em deslocar o indicador de polaridade para o início da linha $IN1_L$ e incluir um Buffer não inversor imediatamente a seguir, o que não provoca qualquer alteração do logigrama. Como resultado destas modificações, obtemos um conversor de polaridade com a linha de entrada designada por $IN1_L$ e com a linha de saída

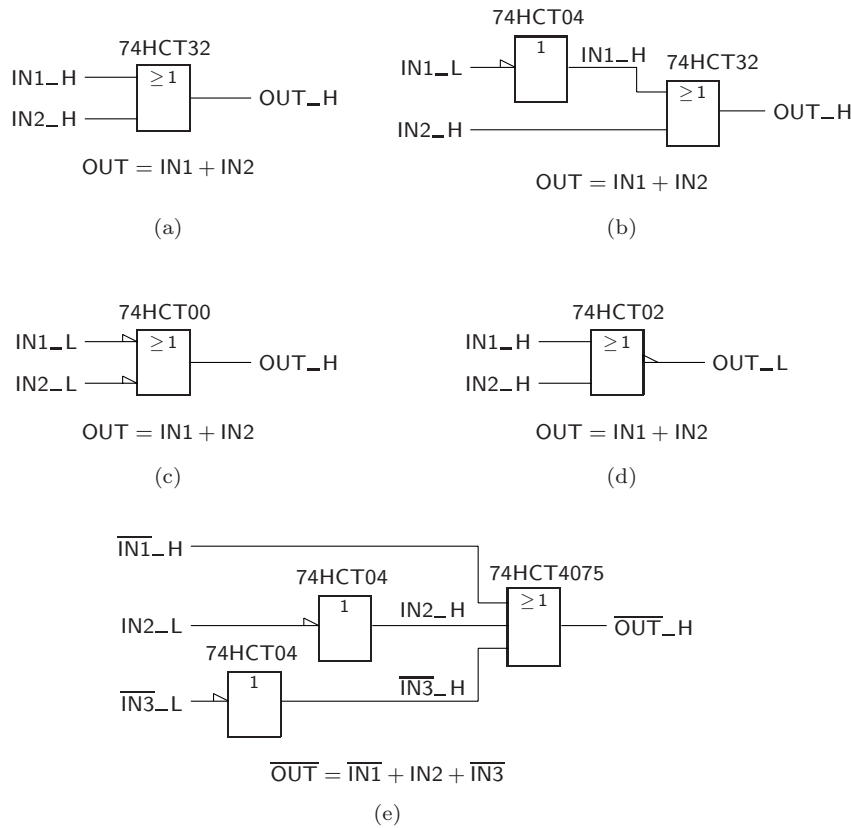


Figura 7.21: Esquemas eléctricos correspondentes aos logigramas dos Exercícios 7.4 e 7.6

designada por $IN1_H$; nestas condições, o OR “vê” nas suas entradas $IN1_H$ e $IN2_H$, com níveis de actividade que concordam com os das suas entradas, e gera na linha de saída a função $OUT = IN1 + IN2$, como se ilustra na Figura 7.21(b).

Os logigramas das Figura 7.18(a) e 7.19(a) não necessitam de qualquer alteração, pelo que os correspondentes esquemas eléctricos podem ser observados nas Figuras 7.21(c) e 7.21(d), respectivamente.

Passemos agora ao logigrama do Exercício 7.6, na Figura 7.20 à direita. Esse logigrama requiere algumas modificações, ilustradas na Figura 7.21(e). As modificações consistem em deslocar os indicadores de polaridade às entradas do OR para o início das linhas $IN2_L$ e $IN3_L$, e incluir Buffers não inversores imediatamente a seguir (o conjunto dos indicadores de polaridade e dos Buffers não inversores formam, mais uma vez, conversores de polaridade). Obtemos, assim, as designações $IN1_H$, $IN2_H$ e $IN3_H$ nas linhas de entrada do OR, pelo que este gera na saída a função $OUT = IN1 + IN2 + IN3$.

7.8 Pretende-se estabelecer o logigrama e a tabela de verdade física de uma função booleana simples, $ACTUATE(IN1, IN2, SEL, DETECT)$, que deve vir activada quando forem satisfeitas uma ou outra mas não ambas as condições

que se descrevem a seguir:

1. as entradas $IN1$ ou $IN2$ ou ambas estão activadas;
2. as entradas SEL e $DETECT$ estão activadas.

Admitir:

- a) que $IN1$, $IN2$, SEL , $DETECT$ e $ACTUATE$ são todas activas a H;
- b) que $IN1$ e $IN2$ são activas a H, e que SEL , $DETECT$ e $ACTUATE$ são activas a L.

7.8 a)

Resolução: a) Começamos por relembrar que os níveis de actividade impostos às variáveis booleanas simples de entrada e à função booleana simples de saída são consequência das respectivas funcionalidades — não necessariamente observáveis no logigrama pretendido mas certamente em algum ponto do logigrama global, do qual o que se pretende é apenas uma parte — que se consubstanciam nos correspondentes significados semânticos. Por exemplo, o facto de a variável SEL ser activa a L significa que, quando for aplicado um nível de tensão L na linha SEL_L , algo virá “seleccionado” em alguma parte do logigrama global. Naturalmente, a variável SEL será gerada noutra parte do logigrama, por outro circuito, que assegurará que a condição “ SEL vem activada quando se quiser fazer a selecção” é sempre cumprida.

Para a resolução do problema, vamos começar por determinar a expressão booleana da função que descreve a saída do circuito, $ACTUATE$, sem nos preocuparmos, para já, com o seu nível de actividade bem como com os níveis de actividade das variáveis de entrada e das funções nos pontos intermédios do logigrama. Como se pretende que a saída venha activada quando uma e só uma das condições 1 ou 2 vêm activadas, segue-se que

$$ACTUATE = COND1 \oplus COND2,$$

em que $COND1$ e $COND2$ são funções intermédias que designam, respectivamente, a primeira e a segunda condições do problema. Por sua vez, $COND1$ corresponde a uma função lógica OR com entradas $IN1$ e $IN2$, e $COND2$ é uma função lógica AND com entradas SEL e $DETECT$. Obtemos, assim, o logigrama incompleto da Figura 7.22(a), que corresponde à função booleana simples

$$ACTUATE = (IN1 + IN2) \oplus (SEL \cdot DETECT).$$

Vamos agora considerar os logigramas finais para a função $ACTUATE$, que devem obrigatoriamente incluir os níveis de actividade das entradas e das saídas das portas, bem como os níveis de actividade das variáveis booleanas simples de entrada, da função booleana simples de saída, e das funções intermédias, $COND1$ e $COND2$. Como podemos incluir ou não triângulos indicadores de polaridade nas entradas e nas saídas das 3 portas do logigrama da Figura 7.22(a), isso significa que podemos gerar $2^9 = 512$ logigramas finais distintos para a função. Porém, para este exemplo apenas estamos interessados em alguns desses logigramas, mais concretamente os que resultam: (i) dos níveis de actividade impostos para as variáveis de entrada e para a função de saída do circuito; e (ii) dos níveis de actividade *que escolhermos* para as extremidades das linhas que suportam as funções intermédias $COND1$ e $COND2$.

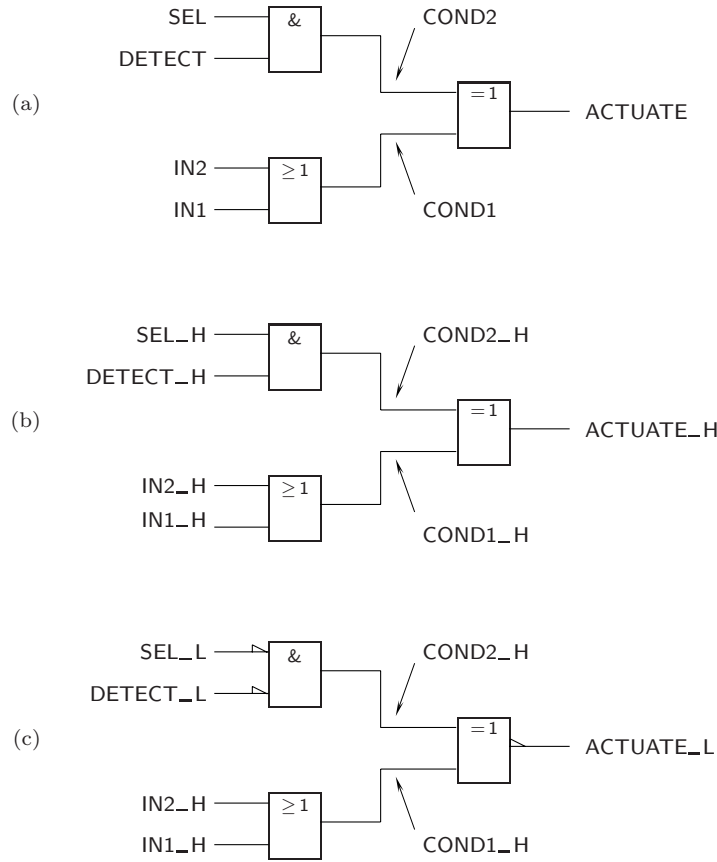


Figura 7.22: (a) Logigrama incompleto para a função $ACTUATE = (IN1 + IN2) \oplus (SEL \cdot DETECT)$, sem indicação dos níveis de actividade nas entradas, na saída e nas funções intermédias, $COND1$ e $COND2$; (b) logigrama do circuito quando $IN1$, $IN2$, SEL , $DETECT$, $ACTUATE$, $COND1$ e $COND2$ são todas activas a H; e (c) logigrama quando $IN1$ e $IN2$ são activas a H, SEL , $DETECT$ e $ACTUATE$ são activas a L, e $COND1$ e $COND2$ são ainda activas a H

Consideremos, por exemplo, as entradas do OR. Como as variáveis de entrada, $IN1$ e $IN2$, são activas a H, isso quer dizer que, quando tivermos H nas linhas correspondentes, $IN1_H$ e $IN2_H$, as entradas do OR devem vir activadas. Conclui-se, assim, que o OR deve ter as suas entradas activas a H. De idêntica forma se conclui que as duas entradas do AND devem ser activas a H para que, quando às linhas SEL_H e $DETECT_H$ se aplicarem níveis de tensão H, a porta AND veja as suas entradas activadas. Ainda pelo mesmo tipo de razões, a saída do XOR deve ser activa a H, para que se produza um H na saída do circuito quando a saída da porta estiver activada.

Nada no exemplo nos diz, contudo, qual deve ser o nível de actividade das saídas do AND e do OR, bem como das entradas do XOR, pelo que podemos escolher arbitrariamente esses níveis *desde que as saídas do AND e do OR, quando activadas, activem as entradas do XOR, para que a função dada não*

venha alterada. Ou seja, se escolhermos a saída do AND activa a H, então a entrada correspondente do XOR também deve ser activa a H; e se escolhermos a saída do AND activa a L, então a entrada correspondente do XOR também deve ser activa a L. Naturalmente, é arbitrária a escolha dos níveis de saída do AND e do OR e de entrada do XOR, desde que a restrição anterior venha cumprida. Se, arbitrariamente, escolhermos para essas entradas e saídas os níveis de actividade H, obtemos o logigrama final da Figura 7.22(b).

Vamos agora estabelecer a tabela de verdade para a função *ACTUATE*, admitindo que as funções *COND1* e *COND2* são activas a H. Reparemos, em particular: (i) que a função *ACTUATE* só deve vir activada (a H) quando *COND1* estiver activa (a H) ou *COND2* estiver activa (a H), mas não ambas activas; (ii) que a função *COND1* deve vir activada (a H) quando *IN1* ou *IN2* ou ambas estiverem activas (a H); e (iii) que a função *COND2* deve vir activada (a H) quando *SEL* e *DETECT* estiverem ambas activas (a H).

Tabela 7.5: Tabela de verdade física para as funções *COND1*, *COND2* e *ACTUATE* da alínea (a) do Exemplo 7.8, representadas no logigrama da Figura 7.22(b)

IN1_H	IN2_H	SEL_H	DETECT_H	COND1_H	COND2_H	ACTUATE_H
L	L	L	L	L	L	L
L	L	L	H	L	L	L
L	L	H	L	L	L	L
L	L	H	H	L	H	H
L	H	L	L	H	L	H
L	H	L	H	H	L	H
L	H	H	L	H	L	H
L	H	H	H	H	H	L
H	L	L	L	H	L	H
H	L	L	H	H	L	H
H	L	H	L	H	L	H
H	L	H	H	H	H	L
H	H	L	L	H	L	H
H	H	L	H	H	L	H
H	H	H	L	H	L	H
H	H	H	H	H	H	L

Nestas condições, torna-se fácil obter a tabela de verdade para a função (Tabela 7.5). Por exemplo, para a primeira linha temos que:

- as entradas da porta OR estão inactivas,

$$IN1_H = IN2_H = L,$$

pelo que a saída da porta também está inactiva ($COND1_H = L$);

- as entradas da porta AND estão inactivas,

$$SEL_H = DETECT_H = L,$$

pelo que a saída da porta também está inactiva ($COND2_H = L$);

- como a porta XOR vê as duas entradas com a mesma polaridade,

$$COND1_H = COND2_H = L,$$

a sua saída deve vir inactiva, isto é, $ACTUATE_H = L$.

Da mesma forma poderíamos obter as restantes linhas da tabela de verdade.

b) Agora SEL e $DETECT$ são activas a L, o que quer dizer que, quando tivermos L nas linhas SEL_L e $DETECT_L$, as duas entradas do AND devem vir activadas. Conclui-se, assim, que o AND deve ter as suas entradas activas a L. Por outro lado, quando tivermos H nas linhas $IN1_H$ e $IN2_H$, o OR deve ver as suas entradas activadas, o que significa que elas devem ser activas a H. E, pelo mesmo tipo de razões, a saída do XOR deve ser activa a L, para que se produza um L na saída do circuito quando a saída da porta estiver activada. Quanto às saídas do AND e do OR e às entradas do XOR, mantemo-las ainda arbitrariamente activas a H, como na alínea anterior. Obtemos, assim, o logigrama final da Figura 7.22(c).

7.8 b)

Notemos, mais uma vez, que *nem sempre os níveis de actividade nas extremidades de uma linha têm que coincidir entre si, ou até coincidir com o nível de actividade da variável ou função que se suporta nessa linha*. A dupla coincidência verificada no presente exemplo resulta de querermos estabelecer um circuito em que se moldam os níveis de actividade nas entradas e saídas das portas aos níveis de actividade das variáveis de entrada e da função de saída do circuito. No caso geral as coincidências apontadas podem não se estabelecer.

Tabela 7.6: Tabela de verdade física para as funções $COND1$, $COND2$ e $ACTUATE$ da alínea (b) do Exemplo 7.8, representadas no logigrama da Figura 7.22(c)

IN1_H	IN2_H	SEL_L	DETECT_L	COND1_H	COND2_H	ACTUATE_L
L	L	L	L	L	H	L
L	L	L	H	L	L	H
L	L	H	L	L	L	H
L	L	H	H	L	L	H
L	H	L	L	H	H	H
L	H	L	H	H	L	L
L	H	H	L	H	L	L
L	H	H	H	H	L	L
H	L	L	L	H	H	H
H	L	L	H	H	L	L
H	L	H	L	H	L	L
H	L	H	H	H	L	L
H	H	L	L	H	H	H
H	H	L	H	H	L	L
H	H	H	L	H	L	L
H	H	H	H	H	L	L

Podemos em seguida escrever a tabela de verdade física da função $ACTUATE$, admitindo que as funções $COND1$ e $COND2$ são activas a H (Tabela 7.6). Por exemplo, para a primeira linha temos agora que:

- as entradas da porta OR estão inactivas (a L), pelo que a saída da porta também está inactiva (a L);
- as entradas da porta AND estão activas (a L), pelo que a saída da porta também está activa (a H);
- como a porta XOR vê as duas entradas com polaridades diferentes, a sua saída deve vir activa, isto é, a L.

Da mesma forma poderíamos obter as restantes linhas da tabela de verdade.

7.10 Pretende-se implementar um circuito lógico que acende uma luz sob comando dos terminais *IN_L* e *TOL_L*. A função que vai permitir acender a luz deverá ser activa a L e será comandada pelos seguintes sinais:

- (1) ligar a luz (*TOL_H*);
- (2) inibir (*IN_L*);
- (3) emergência (*EMERG_L*); e
- (4) a ocasião não é adequada (*TNR_H*).

A luz deverá acender-se desde que a ocasião seja adequada, o comando de luz não seja inibido pela variável *IN*, e seja dada uma ordem para ligar a luz. Se, contudo, se verificar uma emergência, a luz deverá acender-se, independentemente dos outros comandos. Desenhe um logigrama em lógica de polaridade para o circuito, e estabeleça o correspondente esquema eléctrico em lógica positiva.

7.10

Resolução: A luz acende-se se ocorrer uma de duas condições, ou as duas conjuntamente. Portanto

$$LUZ = COND1 + COND2.$$

A condição 1 é dada por

$$COND1 = TOL \cdot \overline{IN} \cdot \overline{TNR}.$$

A condição 2 é dada por

$$COND2 = EMERG.$$

Logo, o logigrama é o da Figura 7.23 se apenas considerarmos a estrutura lógica do problema.

Levando agora em linha de conta os níveis em que as variáveis e a função são activas, obtemos o logigrama da Figura 7.24.

E, a partir deste logigrama, deduz-se o esquema eléctrico da Figura 7.25, em lógica positiva (TTL), formado por um inversor 74x04, por um NAND 74x10 com 3 entradas, e por um AND 74x08 com 2 entradas.

7.16 É dado o logigrama da Figura 7.34 (de *SD:AAT*). Determine a expressão booleana da função *OUT(IN1, IN2, IN3, IN4)*.

7.16

Resolução: Vamos começar por obter as duas designações para a linha de saída

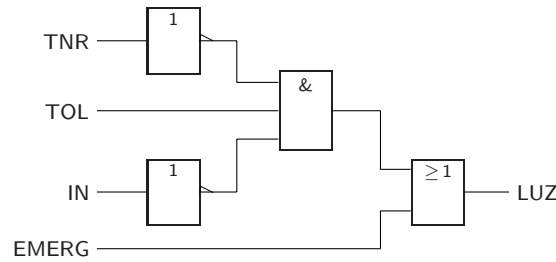


Figura 7.23: Logigramma com a estrutura lógica do problema do Exercício 7.10

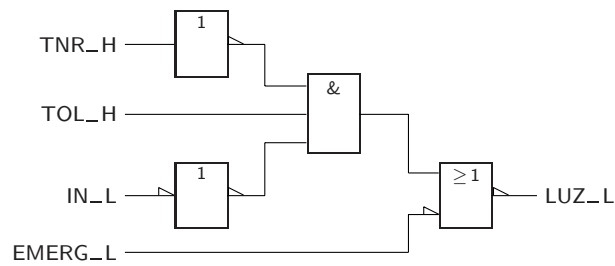


Figura 7.24: Logigramma com a estrutura lógica do problema do Exercício 7.10, que leva em linha de conta os níveis em que as variáveis e a função são activas

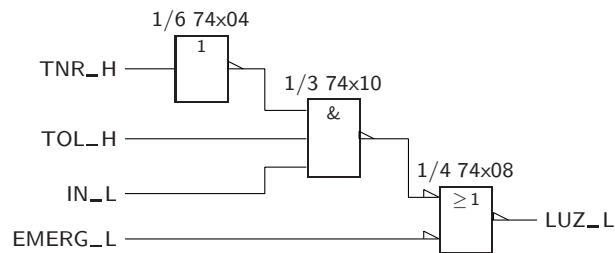


Figura 7.25: Esquema eléctrico (em TTL) do problema do Exercício 7.10

da porta AND. A função à saída desta porta é um produto lógico, e porque a porta tem saída activa a L, a função é activa a L.

Na Figura 7.26(a) apresenta-se a estrutura dessa função, faltando preencher os dois factores do produto lógico. A questão que fica por resolver é se esses factores envolvem *as variáveis de entrada ou os seus complementos*.

Consideremos a variável de entrada $IN1$, que é activa a H. Como já sabemos, podemos designar a linha que suporta essa variável (aliás, qualquer linha) pelo menos de duas maneiras diferentes, mas equivalentes. Neste caso apenas duas designações são possíveis para cada uma das linhas do logigramma (por manifesta falta de alternativas semânticas para INi e para OUT).

Uma das designações para a linha já é dada: $IN1_H$. A outra é, como já sabemos, $\overline{IN1_L}$. A questão a pôr é a seguinte: para a expressão da função à saída do AND, qual das duas expressões é a correcta?

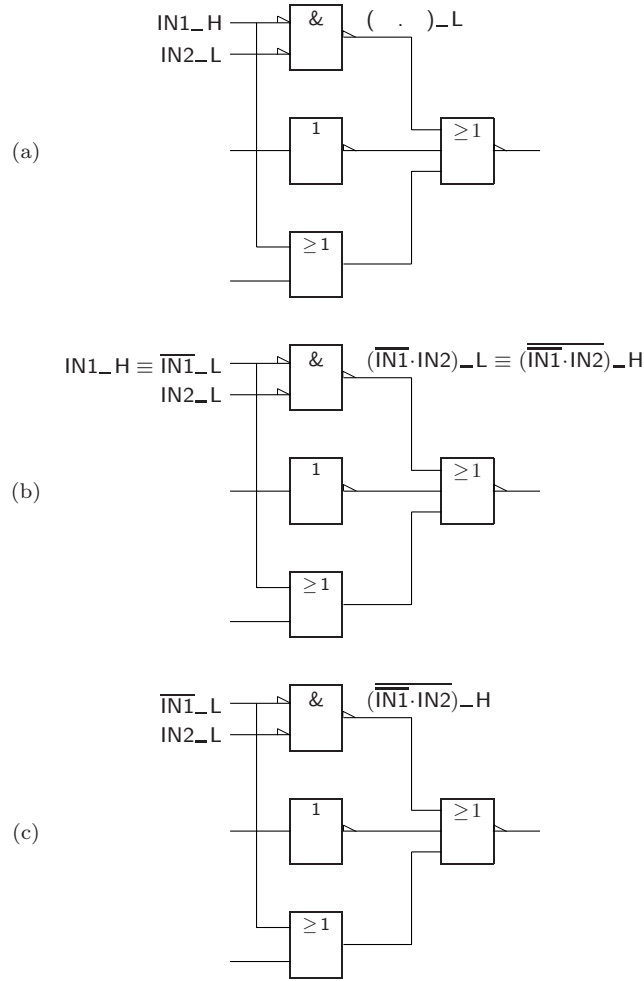


Figura 7.26: (a) O produto lógico à saída do AND é activo a L porque a porta tem saída activa a L. (b) As duas designações alternativas para a linha de saída do AND. (c) Expressão da função à saída do AND que aproveitamos para a expressão da função de saída do circuito, porque o seu nível de actividade coincide com o da entrada do OR final

Se repararmos que um L na linha de entrada não só torna a variável $\overline{IN1}$ activa como também torna activa a entrada correspondente da porta, então na expressão da função interessa considerar $\overline{IN1}$ em vez de $IN1$. Ou seja, consideramos a variável ou o seu complemento por forma a que o correspondente nível de actividade seja igual ao nível de actividade da entrada da porta.

Segue-se que a expressão da função à saída do AND é da forma

$$(\overline{IN1} \dots)_L.$$

Pelas mesmas razões, concluímos que o segundo factor do produto envolve a variável $IN2$, ou seja, aproveitamos a designação indicada para a linha, $IN2_L$, em vez da designação alternativa $\overline{IN2}_H$. Então, a expressão à saída do AND

é, como ilustra a Figura 7.26(b),

$$(\overline{IN1} \cdot IN2)_{-L}.$$

Mas, naturalmente, também esta linha possui uma designação alternativa, também indicada na Figura 7.26(b):

$$\overline{(\overline{IN1} \cdot IN2)}_{-H}.$$

Das duas expressões apenas vamos reter uma delas, a que tem o nível de actividade igual ao da entrada da porta seguinte (o OR final). Como essa entrada é activa a H, vamos reter apenas a expressão

$$\overline{(\overline{IN1} \cdot IN2)}_{-H}.$$

para a função à saída do AND, como ilustra a Figura 7.26(c).

Passemos agora à expressão da função à saída do Buffer. Sendo $IN3_{-L}$ a designação da linha de entrada (poderíamos considerar a designação alternativa, $\overline{IN3}_{-H}$, mas neste caso isso não adianta), a designação da linha de saída do Buffer é, como mostra a Figura 7.27(a).

$$IN3_{-H}$$

ou então

$$\overline{IN3}_{-L}.$$

No primeiro caso o Buffer comporta-se como um conversor de polaridade (porque mantém a variável $IN3$ mas muda-lhe a polaridade), e no segundo como um inversor (porque mantém a polaridade de $IN3$ mas complementa-a).

Vamos agora obter a expressão da função à saída do OR de entrada, como mostra a Figura 7.28.

Agora interessa-nos reter a designação

$$\overline{IN4}_{-H}$$

em vez de

$$IN4_{-L}$$

para a entrada inferior do OR, e

$$IN1_{-H}$$

para a outra entrada, para que os níveis de actividade destas variáveis coincidam com os correspondentes níveis de actividade das entradas do OR. Segue-se que a linha de saída do OR, que é activa a H, tem as designações possíveis indicadas na Figura 7.28(a),

$$(IN1 + \overline{IN4})_{-H}$$

ou

$$\overline{(IN1 + \overline{IN4})}_{-L},$$

sendo que apenas nos interessa a primeira designação, pelos motivos explanados atrás. É o que mostra a Figura 7.28(b).

Retomemos agora, na Figura 7.29, as conclusões anteriores.

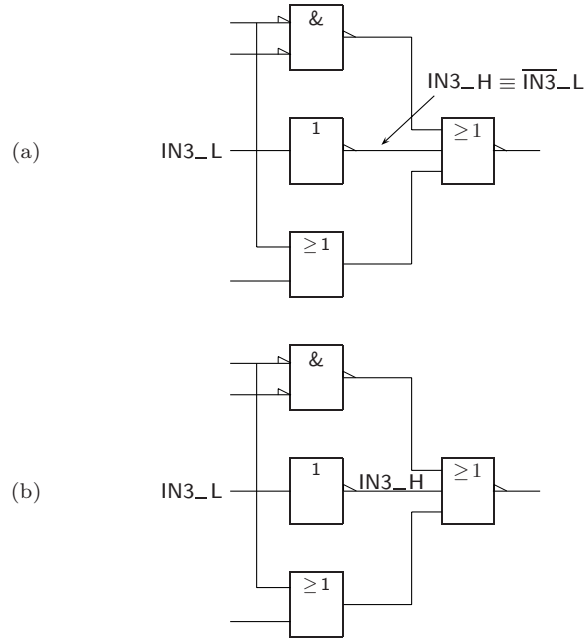


Figura 7.27: (a) A designação da linha de saída do Buffer é uma das duas indicadas. (b) Utiliza-se apenas a designação da função cujo nível de actividade coincide com o da entrada do OR final

Estamos, agora, em posição de determinar a expressão booleana da função *OUT*. Esta função deve ser activa a L, porque o OR final tem a saída activa a L. Logo, a linha de saída deve ter a designação

$$\left[(\overline{\overline{IN1} \cdot IN2}) + IN3 + IN1 + \overline{IN4} \right]_{-L},$$

ou a designação alternativa

$$\left[\overline{\overline{\overline{\overline{IN1} \cdot IN2}} + IN3 + IN1 + \overline{IN4}} \right]_{-H}.$$

Como queremos gerar *OUT_H*, segue-se que

$$OUT = \left[\overline{\overline{\overline{\overline{\overline{IN1} \cdot IN2}} + IN3 + IN1 + \overline{IN4}}} \right],$$

que podemos em seguida simplificar algebricamente para

$$\begin{aligned} OUT &= \left[\overline{\overline{\overline{\overline{\overline{IN1} \cdot IN2}} + IN3 + IN1 + \overline{IN4}}} \right] \\ &= \overline{IN1 + \overline{IN2} + IN3 + IN1 + \overline{IN4}} \\ &= \overline{IN1 + \overline{IN2} + IN3 + \overline{IN4}} \\ &= \overline{IN1} \cdot IN2 \cdot \overline{IN3} \cdot IN4, \end{aligned}$$

embora tal não fosse pedido.

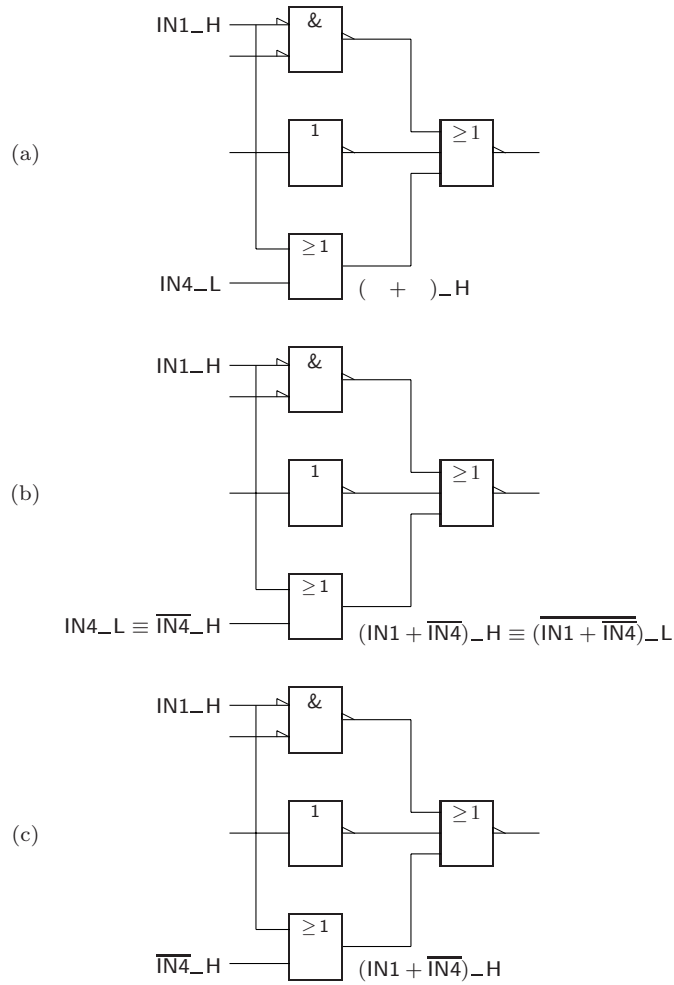


Figura 7.28: (a) A designação da linha de saída do OR de entrada é uma das duas indicadas. (b) Utiliza-se apenas a designação da função cujo nível de actividade coincide com o da entrada do OR final

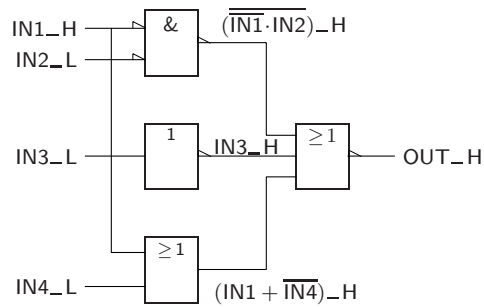


Figura 7.29: Designações que irão contribuir para a expressão da função de saída *OUT*

7.17 Considere os logigramas da Figura 7.35 de *SD:AAT*, todos semelhantes. Estabeleça as tabelas de verdade físicas para as três funções, e deduza directamente as suas expressões lógicas a partir das tabelas. Em seguida, confirme as expressões analisando os logigramas correspondentes.

7.17 a)

Resolução: a) Na Figura 7.30 reproduz-se o logigrama do circuito.

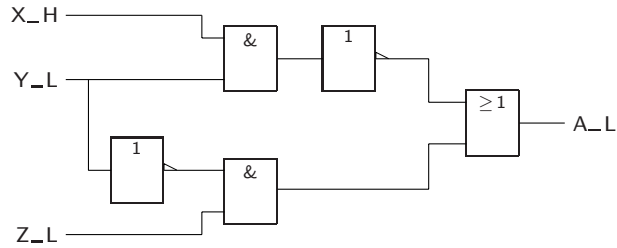


Figura 7.30: Logigrama utilizado no Exercício 7.17 a)

Vamos então estabelecer a tabela de verdade física para a função A , atendendo ao seguinte facto: em todas as linhas em que pelo menos uma das entradas do OR final vier a H, a saída do circuito vem a H.

Ora, basta que se verifique $Y_L = L$ e $Z_L = H$ para que a entrada inferior do OR venha a H. Por outro lado, para que a entrada superior do OR venha a H, basta que $X_H = L$, ou $Y_L = L$, ou ambas. Quando $X_H = H$ e $Y_L = H$, a saída do OR vem a L. Podemos, então, obter a tabela de verdade física da função (Tabela 7.7).

Tabela 7.7: Tabela de verdade física para a função A do Exercício 7.17 a)

X_H	Y_L	Z_L	A_L
L	L	L	H
L	L	H	H
L	H	L	H
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	L
H	H	H	L

Claramente, a função A deve vir dada pela soma de dois mintermos (não necessariamente, m_6 e m_7 , em que 6 e 7 são os números das linhas em que os mintermos ocorrem). Vamos prová-lo através da observação directa da tabela, verificando, para as linhas 6 e 7, quais as variáveis que estão activas e quais as que estão inactivas.

Em relação à linha 6, podemos afirmar que A vem activa quando X estiver activa e \bar{Y} estiver activa e Z estiver activa. Ou seja, o L para a função A na linha 6 corresponde ao mintermo $X\bar{Y}Z = m_5$, se X for a variável de maior peso e Z a de menor peso.

Em relação à linha 7, podemos afirmar que A vem activa quando X estiver

activa e \bar{Y} estiver activa e \bar{Z} estiver activa. Ou seja, o L na linha 7 corresponde ao mintermo $X \bar{Y} \bar{Z} = m_4$.

Segue-se que

$$A = m_4 + m_5 = X \bar{Y} \bar{Z} + X \bar{Y} Z.$$

Esta mesma expressão poderia ter sido obtida directamente do logigrama dado, como mostra a Figura 7.31.

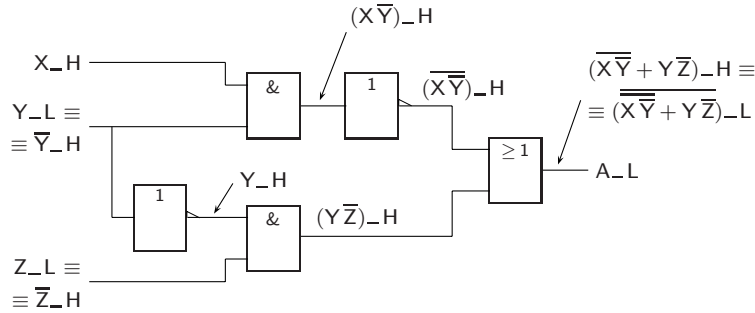


Figura 7.31: Logigrama a analisar para o Exercício 7.17 a)

Com efeito, a expressão que obtemos para a função A ,

$$A = \overline{\overline{X \bar{Y} + Y \bar{Z}}},$$

pode vir simplificada para:

$$\begin{aligned} A &= X \bar{Y} \cdot (\bar{Y} + Z) \\ &= X \bar{Y} + X \bar{Y} Z \\ &= X \bar{Y} \bar{Z} + X \bar{Y} Z \\ &= m_4 + m_5, \end{aligned}$$

(se X for a variável com maior peso e Z a de menor peso), como constatámos anteriormente.

b) Na Figura 7.32 reproduz-se o logigrama do circuito.

7.17 b)

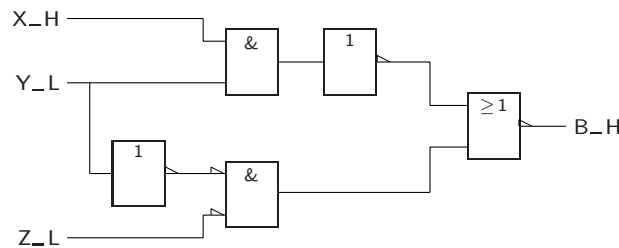


Figura 7.32: Logigrama utilizado no Exercício 7.17 b)

Notemos como, em relação ao logigrama anterior, apenas foram alterados os níveis de actividade das entradas do AND inferior e da saída. Como sabemos, estas mudanças devem originar uma função de saída, B , completamente diferente da anterior (aliás, *bastava termos alterado um, e apenas um, nível de*

actividade numa entrada ou numa saída de uma porta, ou o nível de actividade de uma variável de entrada, ou da função de saída, para que a expressão da função viesse alterada).



Para a geração da tabela de verdade física da função B vamos atender a que, em todas as linhas em que pelo menos uma das entradas do OR final vier a H, a saída do circuito vem a L.

Ora, basta que se verifique $Y_L = H$ e $Z_L = L$ para que a entrada inferior do OR venha a H. Por outro lado, para que a entrada superior do OR venha a H, basta que $X_H = L$, ou $Y_L = L$, ou ambas. Podemos, então, obter a tabela de verdade física da função na Tabela 7.8.

Tabela 7.8: Tabela de verdade física para a função B do Exercício 7.17 b)

X_H	Y_L	Z_L	B_H
L	L	L	L
L	L	H	L
L	H	L	L
L	H	H	L
H	L	L	L
H	L	H	L
H	H	L	L
H	H	H	H

omo facilmente constatamos, a função B apenas vem activa na última linha da tabela, quando $X_H = H$ e $\bar{Y}_H = H$ e $\bar{Z}_H = H$. Em resumo,

$$B = X \bar{Y} \bar{Z}$$

$$= m_4,$$

se X for a variável com maior peso e Z a de menor peso.

Vamos agora verificar esta expressão directamente através do logigrama, como mostra a Figura 7.33.

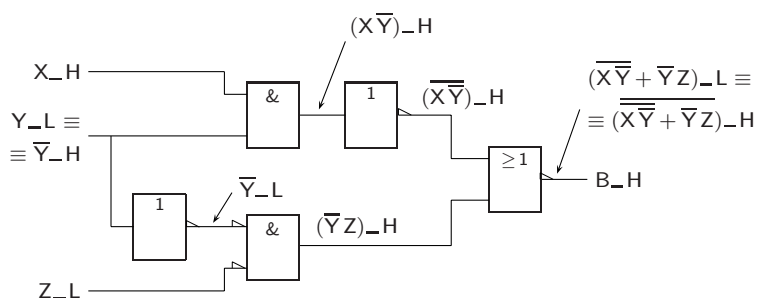


Figura 7.33: Logigrama a analisar para o Exercício 7.17 b)

Deduz-se do logigrama que

$$B = \overline{X \bar{Y} + \bar{Y} Z},$$

expressão esta que pode vir simplificada para:

$$\begin{aligned} B &= X \bar{Y} \cdot (Y + \bar{Z}) \\ &= X \bar{Y} \bar{Z} \\ &= m_4, \end{aligned}$$

mais uma vez se X for a variável com maior peso e Z a de menor peso.

c) Na Figura 7.34 reproduz-se o logigrama do circuito, e na Tabela 7.9 apresenta-se a correspondente tabela de verdade física. .

7.17 c)

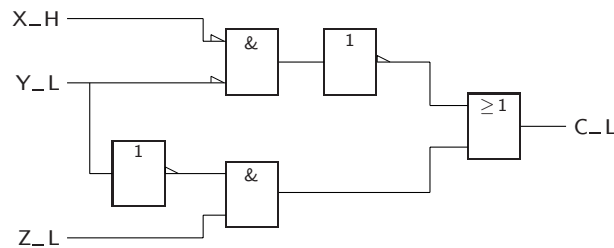


Figura 7.34: Logigrama utilizado no Exercício 7.17 c)

Tabela 7.9: Tabela de verdade física para a função C do Exercício 7.17 c)

X_H	Y_L	Z_L	C_L
L	L	L	L
L	L	H	H
L	H	L	H
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	H

Deduz-se da primeira linha desta tabela que

$$\begin{aligned} C &= \bar{X} Y Z \\ &= m_3, \end{aligned}$$

se X for a variável com maior peso e Z a de menor peso.

Na Figura 7.35 indicam-se as expressões booleanas das funções intermédias e final.

Deduz-se do logigrama que

$$C = \overline{\overline{\bar{X} Y} + Y \bar{Z}}$$

ou, mais simplesmente,

$$\begin{aligned} C &= \bar{X} Y \cdot (\bar{Y} + Z) \\ &= \bar{X} Y Z \\ &= m_3, \end{aligned}$$

mais uma vez se X for a variável com maior peso e Z a de menor peso.

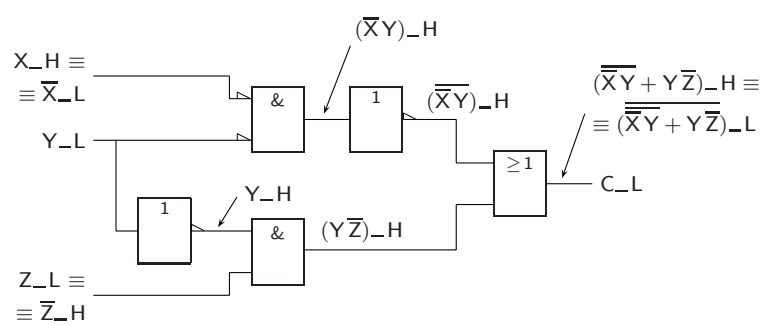


Figura 7.35: Logigrama a analisar para o Exercício 7.17 c)

Capítulo 9

Codificadores e Descodificadores

9.1 Na parte (a) da Figura 9.8 (de *SD:AAT*) apresenta-se um decodificador binário de 3 bits com entradas activas a H, enquanto que na parte (b) se apresenta um outro decodificador, em tudo idêntico ao anterior excepto pelas entradas, que são agora activa a L. Se a estes decodificadores se aplicar às entradas a quantidade booleana geral $(INA, INB, INC) = (H, L, L)$, qual é a saída que vem activa nos dois casos? Porquê?

Resolução: Notemos que, para os dois decodificadores, a variável INA está suportada na linha que se encontra ligada à entrada 4, com peso 4. Segue-se que essa variável tem peso 4. Por idênticas razões podemos concluir que INB tem peso 2 e que INC tem peso 0.

9.1

Relembremos que a saída de um decodificador que vem activada em cada instante é a que resulta da soma dos pesos das entradas activadas nesse instante.

Ora, no caso da Figura 9.8(a) as entradas são todas activas a H. Logo, quando a quantidade booleana geral $(INA, INB, INC) = (H, L, L)$ vem aplicada a essas entradas, apenas a entrada ligada a INA_H vem activa. Como esta entrada tem peso 4, segue-se que é a saída 4 que vem activada.

Pelo contrário, no caso da Figura 9.8(b) as entradas são todas activas a L. Logo, quando a quantidade booleana geral $(INA, INB, INC) = (H, L, L)$ vem aplicada a essas entradas, vêm activadas as entradas ligadas a INB_H e a INC_H . Como estas entradas possuem, respectivamente, os pesos 2 e 1, segue-se que a soma dos pesos das entradas activas é igual a $2 + 1 = 3$ e é a saída 3 que vem activada neste caso.

9.3 Escrever a tabela de verdade física do decodificador BCD da Figura 9.3 (de *SD:AAT*).

Resolução: Começamos por considerar o símbolo IEC deste decodificador, ao qual acrescentamos possíveis designações para as linhas de entrada e de saída. Obtemos, assim, o logograma da Figura 9.1.

9.3

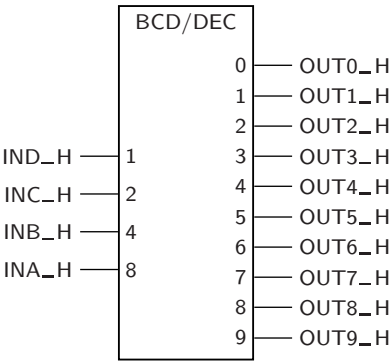


Figura 9.1: Símbolo IEC do decodificador BCD da Figura 9.3 de *SD:AAT*, ao qual se acrescentaram possíveis designações para as linhas de entrada e de saída

Com base neste logigrama, podemos imediatamente deduzir a tabela de verdade física para o decodificador (Tabela 9.1).

Tabela 9.1: Tabela de verdade física do decodificador BCD da Figura 9.3 de *SD:AAT*

IN				OUT									
A_H	B_H	C_H	D_H	9_H	8_H	7_H	6_H	5_H	4_H	3_H	2_H	1_H	0_H
L	L	L	L	L	L	L	L	L	L	L	L	L	H
L	L	L	H	L	L	L	L	L	L	L	L	H	L
L	L	H	L	L	L	L	L	L	L	L	H	L	L
L	L	H	H	L	L	L	L	L	L	H	L	L	L
L	H	L	L	L	L	L	L	L	H	L	L	L	L
L	L	L	H	L	L	H	L	H	L	L	L	L	L
L	H	H	L	L	L	L	H	L	L	L	L	L	L
L	H	H	H	L	L	H	L	L	L	L	L	L	L
H	L	L	L	L	H	L	L	L	L	L	L	L	L
H	L	L	H	H	L	L	L	L	L	L	L	L	L

9.6 Como será o símbolo IEC do decodificador expandido da Figura 9.6 (de *SD:AAT*)?

9.6 *Resolução:* Ver a Figura 9.2.

De notar que o qualificador geral BIN/HEX pode ser substituído pelo qualificador BIN/1-OF-16.

74x138 **9.12** Diga como poderá utilizar 9 decodificadores do tipo 74x138 para implementar um decodificador com 6 linhas de entrada e 64 linhas de saída. O decodificador 74x138 tem o símbolo IEC que se ilustra na Figura 9.9 (de *SD:AAT*).

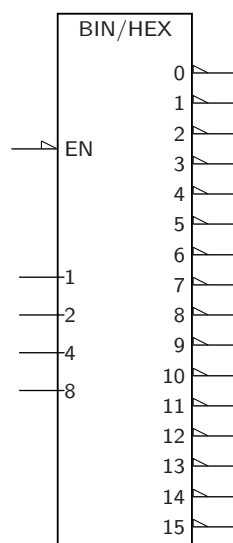


Figura 9.2: Símbolo IEC do decodificador expandido da Figura 9.6 de *SD:AAT*

Resolução: Ver a Figura 9.3.

9.12

É de referir que o decodificador 74x138 tem 3 entradas de Enable, sendo duas activas a L e uma activa a H, que são combinadas internamente numa porta AND para fornecer, também internamente, o sinal de EN das saídas. De notar ainda que as saídas são todas activas a H, pelo que, se EN estiver inactivo, as saídas vêm todas a H.

9.15 Implementar a função booleana simples $F(a, b, c) = \sum m(1 - 3, 7)$ utilizando um decodificador binário de 3 bits com saídas activas a H, como o da Figura 9.1. Que funções e correspondentes níveis de actividade se obtêm nas saídas do decodificador?

Resolução: Como sabemos da Subsecção 9.1.2, cada uma das saídas activas a H de um decodificador gera um mintermo também activo a H, com um índice que é igual ao do qualificador da saída correspondente. Por exemplo, a saída 0 do decodificador gera o mintermo m_0 , a saída 1 gera o mintermo m_1 , etc.

9.15

Vamos provar esta afirmação para uma das saídas do decodificador dado, por exemplo a saída 0. Admitamos que designamos as linhas de entrada de dados do decodificador por a_H , b_H e c_H , sendo a a variável de maior peso (por isso, virá ligada à entrada de dados de maior peso do decodificador).

A saída 0 só vem activa (a H) quando nas entradas se aplicar a quantidade booleana geral $(a_H, b_H, c_H) = (L, L, L)$. Nessas condições obtém-se, nessa saída, a função $\bar{a}\bar{b}\bar{c}$, ou seja, m_0 . Por isso podemos designar a linha de saída correspondente à saída 0 por m_0_H .

Naturalmente, outro tanto acontece com as restantes saídas, como mostra a Figura 9.4.

Como neste exercício queremos gerar a função $F(a, b, c) = m_1 + m_2 + m_3 + m_7$, devemos utilizar um OR (porque queremos uma *soma* de mintermos). E como

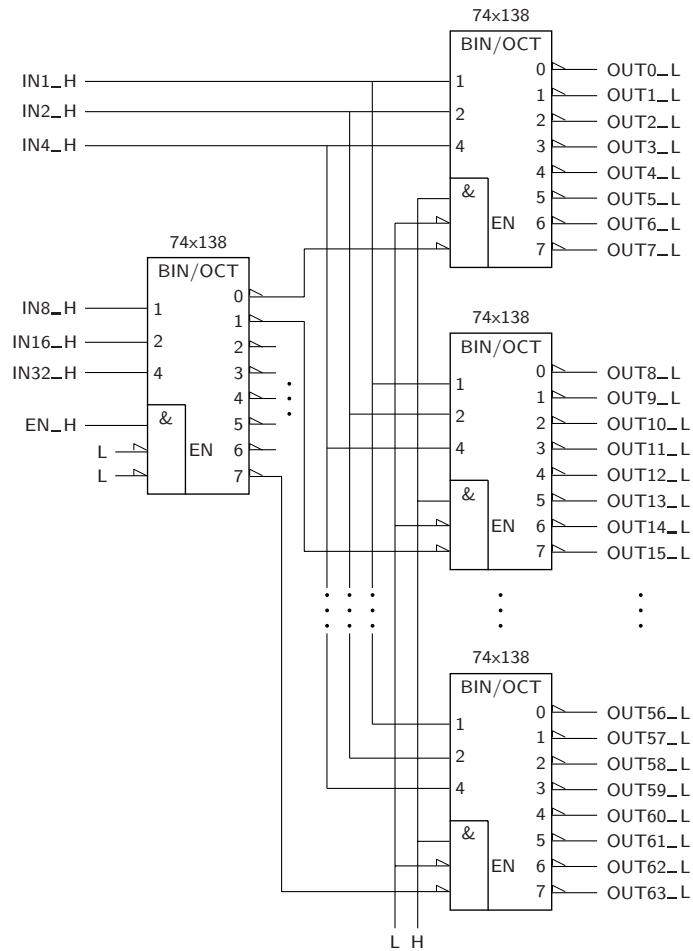


Figura 9.3: Logigrama de um decodificador com 6 linhas de entrada e 64 linhas de saída, construído com decodificadores do tipo 74x138

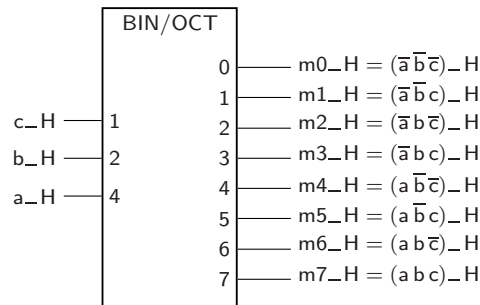


Figura 9.4: Um decodificador binário de 3 bits com saídas activas a H gera, em cada uma delas, um mintermo activo a H

temos que somar 4 mintermos, precisamos de um OR com 4 entradas. Por outro lado, como os mintermos são activos a H nas saídas do decodificador, o OR

também deverá ter as entradas activas a H (para que na expressão da função F apareçam os m_i). Finalmente, queremos gerar F activo a H, pelo que a saída do OR deve ser activa a H.

Em resumo, precisamos de um OR com 4 entradas activas a H e saída activa a H para gerar F_H , como mostra o logigrama da Figura 9.5.

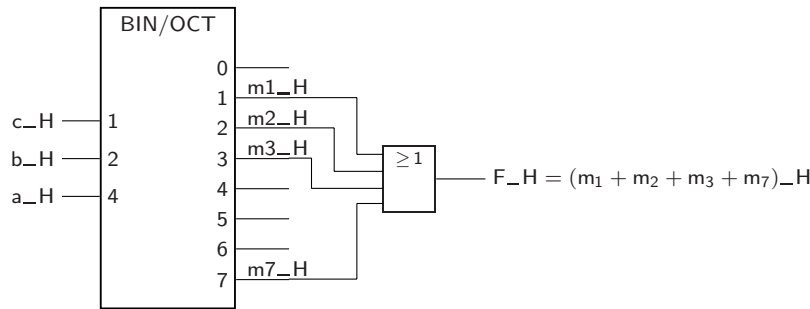


Figura 9.5: Logigrama com a implementação da função booleana simples $F(a, b, c)_H = (m_1 + m_2 + m_3 + m_7)_H$, que usa um decodificador binário de 3 bits com saídas activas a H

Notemos que o logigrama da Figura 9.5 só não é esquema eléctrico em lógica positiva, por exemplo TTL ou CMOS, porque não existem ORs de 4 entradas nessa lógica. Ou seja, para passarmos do logigrama ao esquema eléctrico em lógica positiva temos de substituir o OR final por ORs com menos entradas.

Uma solução possível consiste em usar 3 ORs de 2 entradas do tipo 74HCT32, como mostra a Figura 9.6, o que permite utilizar apenas um integrado para além do decodificador (admite-se o uso de uma tecnologia CMOS compatível com TTL).

74HCT32

De notar que o decodificador integrado é um 74HCT238, em tudo semelhante ao 74x138 — usado, por exemplo, no texto teórico *SD:AAT* — excepto pelas saídas, que são activas a H em vez de a L.

74HCT238

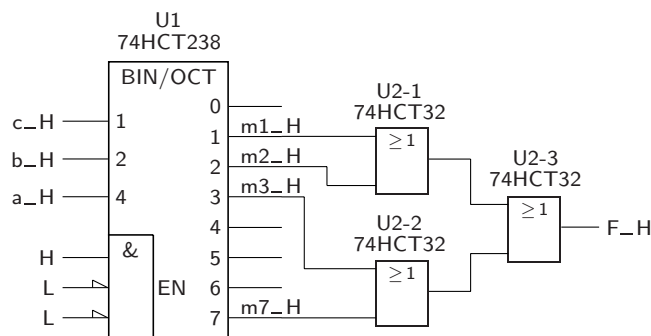


Figura 9.6: Esquema eléctrico possível (sem indicação dos pinos) de implementação da função booleana simples $F(a, b, c)_H = (m_1 + m_2 + m_3 + m_7)_H$, que usa o 74HCT238, um decodificador binário de 3 bits com saídas activas a H

9.16 Implementar a função booleana simples $F(a, b, c) = \sum m(1-3, 7)$ do exercício anterior, mas utilizando agora um decodificador binário de 3 bits com saídas activas a L, como o 74x138 da Figura 9.9 (de *SD:AAT*). Que funções e correspondentes níveis de actividade se obtêm nas saídas do decodificador?

9.16

Resolução: Consideremos o exercício anterior, onde um decodificador binário com saídas activas a H permitia obter os diversos mintermos *ativos a H* da função.

Como agora queremos utilizar um decodificador com saídas activas a L, os mintermos gerados nas saídas vêm agora *ativos a L*, como mostra a Figura 9.7.

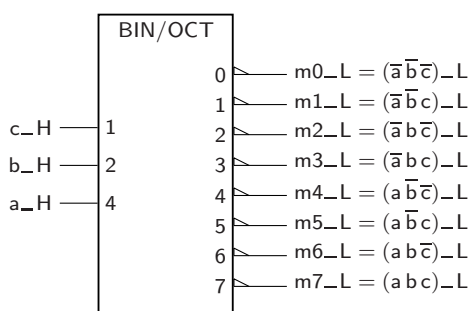


Figura 9.7: Um decodificador binário de 3 bits com saídas activas a L gera, em cada uma delas, um mintermo activo a L

Como neste exercício queremos ainda gerar a função $F(a, b, c) = m_1 + m_2 + m_3 + m_7$, devemos utilizar um OR (porque queremos uma *soma* de mintermos). E como temos que somar 4 mintermos, precisamos de um OR com 4 entradas. Por outro lado, como os mintermos são activos a L nas saídas do decodificador, o OR também deverá ter as entradas activas a L (para que na expressão da função F apareçam os m_i). Finalmente, queremos gerar F activo a H, pelo que a saída do OR deve ser activa a H.

Em resumo, precisamos de um OR com 4 entradas activas a L e saída activa a H para gerar F_H , como mostra o logigrama da Figura 9.8.

Se quisermos passar do logigrama da Figura 9.8 para o esquema eléctrico em lógica positiva, por exemplo, temos que perceber qual é, nessa lógica, a porta que corresponde a um OR com 4 entradas activas a L e saída activa a H. Para tanto basta lembrarmo-nos do símbolo alternativo para este OR, que é um AND com entradas activas a H e saída activa a L (um NAND de 4 entradas em lógica positiva, por exemplo metade de um 74HCT20).

74HCT20



O facto de obtermos um NAND em lógica positiva não quer dizer que o esquema eléctrico deva incluir o símbolo habitual para um NAND, porque então perderíamos a informação semântica de que estamos a fazer uma soma de mintermos. Ou seja, queremos, quer no logigrama, quer no esquema eléctrico, preservar o OR final. O que fazemos é, no esquema eléctrico, indicar explicitamente, através da designação respectiva, que se trata de um NAND em lógica positiva, mas mantemos o símbolo inicial do OR com entradas activas a L e saída activa a H.

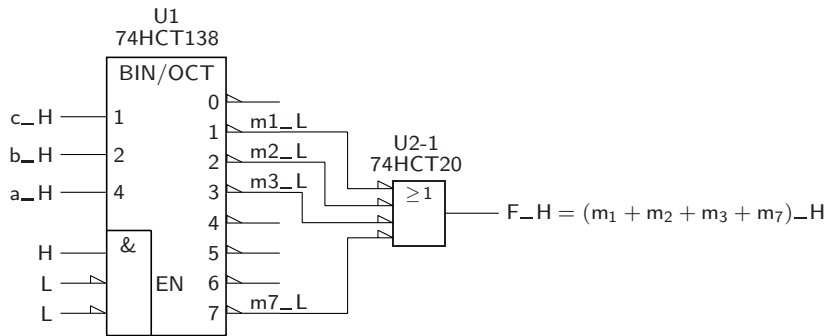


Figura 9.8: Logigrama e esquema eléctrico (sem indicação dos pinos) de implementação da função booleana simples $F(a, b, c)_H = (m_1 + m_2 + m_3 + m_7)_H$, que usa o 74HCT138, um decodificador binário de 3 bits com saídas activas a L

Ou seja, o logigrama da Figura 9.8 é também esquema eléctrico em lógica positiva (admitiu-se a utilização de uma tecnologia CMOS compatível com TTL), e a indicação do NAND nessa lógica aparece explicitamente através da designação 74HCT20.

9.17 Implementar a função booleana simples $F(a, b, c) = \prod M(0, 4 - 6)$ utilizando um decodificador binário de 3 bits com saídas activas a H, como o da Figura 9.1 de *SD:AAT* (de notar que esta função é a mesma dos Exercícios 9.15 e 9.16). Que funções e correspondentes níveis de actividade se obtêm nas saídas do decodificador?

Resolução: Agora vamos utilizar um decodificador com saídas activas a H e queremos fazer um produto de maxtermos da função.

9.17

Põe-se, então, a seguinte questão: será que, nas saídas do decodificador, podemos ver aparecer os maxtermos de F ?

A resposta a esta questão pode ser dada se atendermos a que uma linha pode ser designada de quatro maneiras diferentes em alternativa, desde que a variável ou função que se suporta na linha possua conteúdo semântico — no caso de não possuir conteúdo semântico, apenas podemos gerar duas designações alternativas, em vez de quatro. No nosso caso vamos obter quatro variantes, como veremos já de seguida.

Vamos, então, procurar as designações alternativas às que foram utilizadas na Figura 9.4, isto é, m_i_H , com $0 \leq i \leq 7$.

Como sabemos da teoria da Secção 7.2 de *SD:AAT*, podemos dar à saída m_i_H as designações

$$m_i_H \quad \text{ou} \quad \overline{m_i_L}.$$

Mas se atendermos a que o complemento de um mintermo é o maxtermo com o mesmo índice, ou seja, que $\overline{m_i} = M_i$, podemos ainda arranjar mais duas alternativas:

$$\begin{aligned}
 &m_i_H, \quad \text{ou} \\
 &\overline{m_i}_L, \quad \text{ou} \\
 &M_i_L, \quad \text{ou} \\
 &\overline{M_i}_H.
 \end{aligned}$$

Todas destas designações servem para identificar cada uma das linhas de saída do decodificador, pelo que podemos escolher qualquer uma delas. Porém, como queremos que a função F implemente o produto de alguns dos seus *maxtermos*, devemos escolher para a linha i de saída a designação que envolve M_i , ou seja, M_i_L .

Obtemos, assim, o logigrama parcial da Figura 9.9.

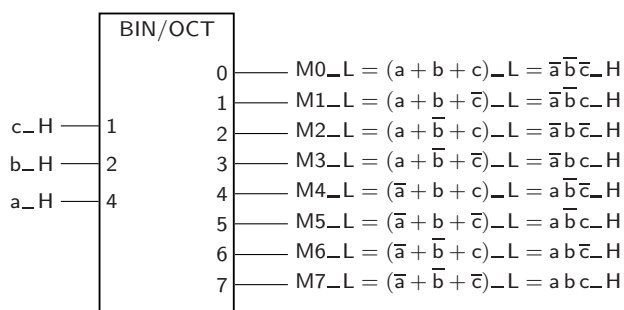


Figura 9.9: Um decodificador binário de 3 bits com saídas activas a H gera, em cada uma delas, um maxtermo activo a L

Como agora queremos gerar a função $F(a, b, c) = M_0 M_4 M_5 M_6$, devemos utilizar um AND (porque queremos um *produto* de maxtermos). E como temos que fazer o produto de 4 maxtermos, precisamos de um AND com 4 entradas. Por outro lado, como os maxtermos são activos a L nas saídas do decodificador, o AND também deverá ter as entradas activas a L (para que na expressão da função F apareçam os M_i). Finalmente, queremos gerar F activo a H, pelo que a saída do AND deve ser activa a H.

Em resumo, precisamos de um AND com 4 entradas activas a L e saída activa a H para gerar F_H , como mostra o logigrama da Figura 9.10. Em lógica positiva o AND com entradas activas a L e saída activa a H é um NOR de 4 entradas, por exemplo metade de um integrado 74HCT4002.

74HCT4002

9.18 Implementar a função booleana simples $F(a, b, c) = \prod M(0, 4 - 6)$ utilizando um decodificador binário de 3 bits com saídas activas a L, como o 74x138 da Figura 9.9 de *SD:AAT* (de notar que esta função é a mesma dos Exercícios 9.15 e 9.16). Que funções e correspondentes níveis de actividade se obtêm nas saídas do decodificador?

9.18

Resolução: Agora vamos utilizar um decodificador com saídas activas a L e queremos fazer um produto de maxtermos da função. Naturalmente, devemos designar as linhas de saída como se indica na Figura 9.11.

Com efeito, sabemos do Exercício 9.16 que o decodificador gera mintermos activos a L nas suas saídas activas a L. Mas como $m_i_L = M_i_H$, segue-se que

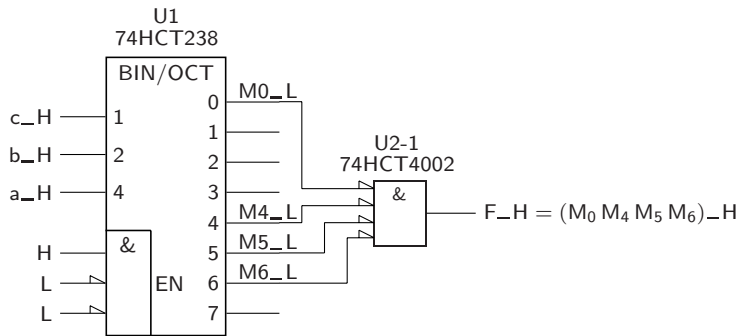


Figura 9.10: Logograma de implementação da função booleana simples $F(a, b, c)_H = (M_0 M_4 M_5 M_6)_H$, que usa um decodificador binário de 3 bits com saídas activas a H

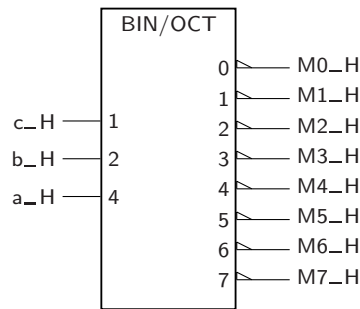


Figura 9.11: Um decodificador binário de 3 bits com saídas activas a L gera, em cada uma delas, um maxtermo activo a H

devemos designar as linhas de saída como indica a figura, para podermos em seguida fazer o produto de maxtermos pretendido.

O produto de maxtermos requiere, então, um AND com 4 entradas e saída activas a H (metade de um circuito integrado 74HCT21), como ilustra a Figura 9.12.

74HCT21

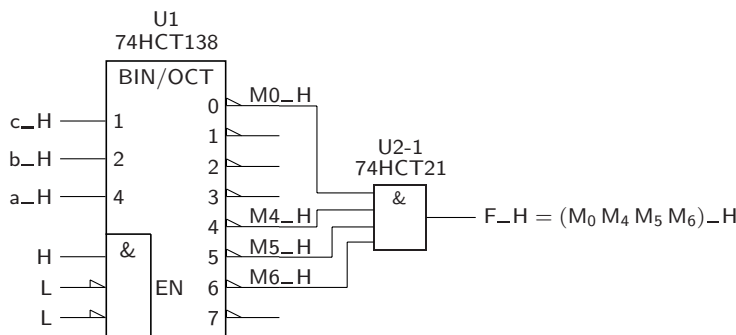


Figura 9.12: Logograma e esquema eléctrico (sem indicação dos pinos) de implementação da função booleana simples $F(a, b, c)_H = (M_0 M_4 M_5 M_6)_H$, que usa um decodificador binário de 3 bits com saídas activas a L

9.25 Traçar o logigrama de um codificador de prioridades com 4 entradas, I0 a I3, e duas saídas, A1 e A0. A entrada I3 deverá ter prioridade sobre I2 que, por sua vez, deverá ter prioridade sobre I1, e esta sobre I0. Prever ainda a existência de uma entrada de Enable e de duas outras saídas, uma de Enable e outra de Grupo, em que esta última indica se, estando o codificador activo, há pelo menos uma entrada activa. Todas as entradas deverão ser activas a H.

9.25

Resolução: Começemos por considerar uma tabela que define o funcionamento global do codificador de prioridades (Tabela 9.2).

Tabela 9.2: Tabela que define o funcionamento global do codificador de prioridades

El_H	Alguma entrada activa?	EO_H	GS_H
L	×	L	L
H	Não	H	L
H	Sim	L	H

Nota: EI = Entrada de Enable
EO = Saída de Enable
GS = Sinal de grupo

Como se pode ver, quando a entrada EI está a L (inactiva), todas as saídas estão inactivas (e, portanto, a L). O circuito está, nessas condições, inibido.

Quando EI está activa, as saídas podem vir activadas.

A saída EO serve para ligar vários destes decodificadores em cadeia, por forma a aumentar o número de entradas a codificar. A ligação é feita com a saída EO deste codificador ligada à entrada EI do codificador seguinte. Nessas circunstâncias, a presença de uma entrada activa neste codificador (segunda coluna da tabela) provoca a inibição do codificador seguinte (porque foi encontrada a entrada mais prioritária dos dois). Pelo contrário, se não existir nenhuma entrada activa neste codificador, a função de codificação de prioridades “é passada” ao codificador seguinte na cadeia.

A saída GS indica se, estando este codificador “Enabled”, pelo menos uma das suas entradas está activa. Isso permite validar a configuração que o codificador apresenta nas saídas A1 e A0 e, em particular, separar a situação correspondente a (L, L) nessas saídas quando não há entradas activas, da situação (L, L) nessas saídas quando a entrada IO está activa.

Com base nestes dados, podemos construir agora a tabela de verdade física para o codificador de prioridades (Tabela 9.3).

Com base nesta tabela podemos construir os mapas de Karnaugh da Figura 9.13, e deduzir as seguintes equações lógicas:

$$A1 = EI \cdot (I3 + I2)$$

$$A0 = EI \cdot (I3 + \overline{I2} \cdot I1).$$

Tabela 9.3: Tabela de verdade física para o codificador de prioridades

EI_H	I3_H	I2_H	I1_H	I0_H	A1_H	A0_H	E0_H	GS_H
L	×	×	×	×	L	L	L	L
H	L	L	L	L	L	L	H	L
H	L	L	L	H	L	L	L	H
H	L	L	H	L	L	H	L	H
H	L	L	H	H	L	H	L	H
H	L	H	L	L	H	L	L	H
H	L	H	L	H	H	L	L	H
H	L	H	H	L	H	L	L	H
H	L	H	H	H	H	L	L	H
H	H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L	H
H	H	L	H	L	H	H	L	H
H	H	L	H	H	H	H	L	H
H	H	H	L	L	H	H	L	H
H	H	H	L	H	H	H	L	H
H	H	H	H	L	H	H	L	H
H	H	H	H	H	H	H	L	H

		$I3\ I2$			
$I1\ I0$		00	01	11	10
	00	0	1	1	1
	01	0	1	1	1
	11	0	1	1	1
	10	0	1	1	1

A1, com $EI = 1$

		$I3\ I2$			
$I1\ I0$		00	01	11	10
	00	0	0	1	1
	01	0	0	1	1
	11	1	0	1	1
	10	1	0	1	1

A0, com $EI = 1$

Figura 9.13: Mapas de Karnaugh para as funções de saída A1 e A0

Finalmente, podemos obter

$$\begin{aligned}
 EO &= EI \cdot (\overline{I3} \cdot \overline{I2} \cdot \overline{I1} \cdot \overline{I0}) \\
 GS &= EI \cdot (I3 + I2 + I1 + I0) \\
 &= EI \cdot (\overline{\overline{I3} \cdot \overline{I2} \cdot \overline{I1} \cdot \overline{I0}}) \\
 &= EI \cdot \overline{EO}.
 \end{aligned}$$

Podemos, então, obter o logigrama da Figura 9.14.

Para que servem os "buffers" à esquerda do logigrama?

9.29 Implementar as funções booleanas simples

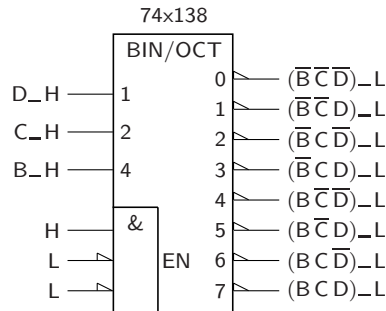


Figura 9.15: Logigrama com um decodificador do tipo 74x138 ligado às 3 variáveis booleanas simples de menor peso da função $f(A, B, C, D)$

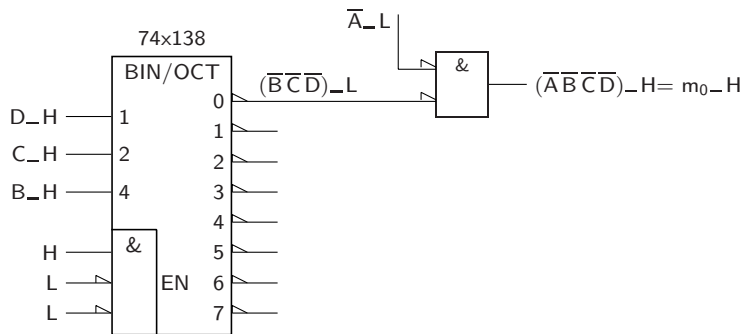


Figura 9.16: Logigrama com a geração do mintermo $m_0\text{--H}$ de $f(A, B, C, D)$, usando para tanto um decodificador do tipo 74x138 ligado às 3 variáveis de menor peso da função

Mas há uma maneira mais simples de resolver o problema, que recorre às entradas de Enable do decodificador (essas entradas foram activadas em permanência na solução anterior e, por isso, desaproveitadas).

Se, por exemplo, designarmos por $\overline{A}\text{--H}$ a linha ligada à entrada de Enable activa a H, isto é, se aplicarmos o literal \overline{A} a essa entrada, então as saídas do 74x138 passam a incluir esse literal. Podemos, desta forma, obter nas saídas do decodificador todos os mintermos de f que contêm o literal \overline{A} , isto é, m_0 a m_7 , como mostra o logigrama da Figura 9.18.

Em alternativa, obteríamos o mesmo resultado se aplicássemos o literal \overline{A} a uma das linhas de Enable activas a L, desde que \overline{A} seja activa a L e designemos a linha correspondente por $\overline{A}\text{--L}$. E se aplicássemos o literal A em vez do literal \overline{A} a uma dessas linhas de Enable, com o nível de actividade adequado, obteríamos os mintermos m_8 a m_{15} da função.

Evidentemente, com o logigrama da Figura 9.18 não temos maneira de obter directamente das saídas do decodificador os mintermos que nos faltam, com índices superiores a 7 (que utilizam o literal A em vez do literal \overline{A}). A única forma de os obter consiste em recorrer a portas AND, de forma semelhante à que utilizámos na primeira solução, como mostra a Figura 9.19.

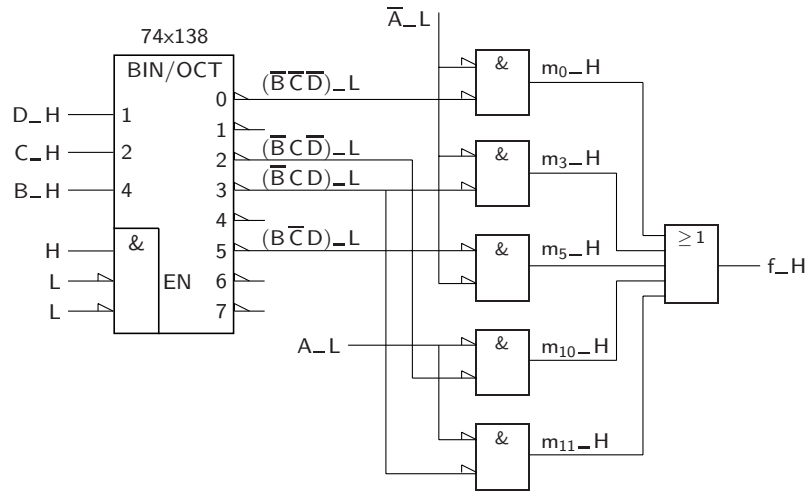


Figura 9.17: Logigrama que implementa a primeira forma canónica de $f(A, B, C, D)$, e que utiliza um decodificador do tipo 74x138 e alguma lógica adicional

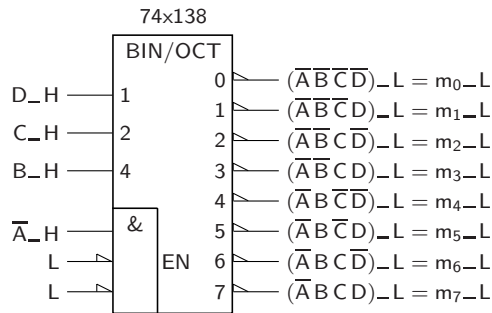


Figura 9.18: Logigrama com a geração dos primeiros 8 mintermos de $f(A, B, C, D)$, que utiliza exclusivamente um decodificador do tipo 74x138

Também é evidente que poderíamos recorrer a dois decodificadores 74x138 para gerar todos os mintermos da função, se o enunciado do exercício o permitisse (e não o permite). Nesse caso, num dos decodificadores geraríamos os mintermos m_0 a m_7 à custa do literal \bar{A} aplicado a uma entrada de Enable, e os mintermos m_8 a m_{15} à custa do literal A aplicado a uma entrada de Enable do outro decodificador.

9.29 b)

b) Neste caso temos uma função g com 5 variáveis, pelo que um decodificador (e só um, como exige o enunciado do exercício) apenas pode gerar maxtermos num dos conjuntos $\{M_0, \dots, M_7\}$ ou $\{M_8, \dots, M_{15}\}$ ou $\{M_{16}, \dots, M_{23}\}$ ou $\{M_{24}, \dots, M_{31}\}$. Se admitirmos que o decodificador gera M_0 , M_5 e M_7 , precisamos de implementar os restantes 3 maxtermos com lógica adicional. Qualquer outra solução exige a geração de mais maxtermos com lógica suplementar.

Para a geração de M_0 , de M_5 e de M_7 precisamos de ter $A_H = B_H = L$, o que exige uma porta AND para activar uma entrada de Enable do 74x138. Escolhemos, arbitrariamente, uma das duas entradas de Enable activas a L para

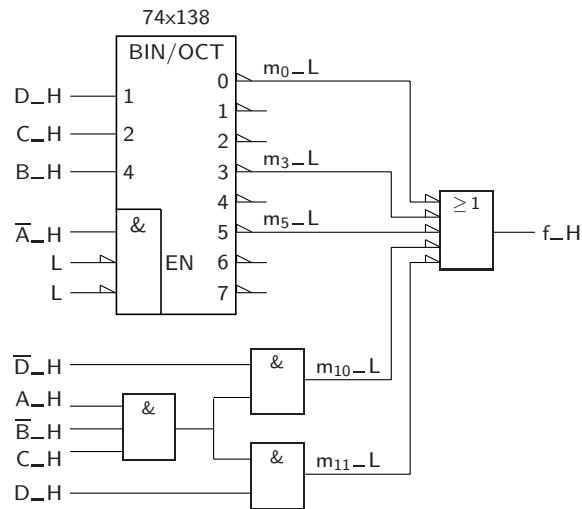


Figura 9.19: Logigrama que implementa a primeira forma canónica de $f(A, B, C, D)$, recorrendo a um descodificador do tipo 74x138 e a algumas portas adicionais

o fazer, como mostra a Figura 9.20.

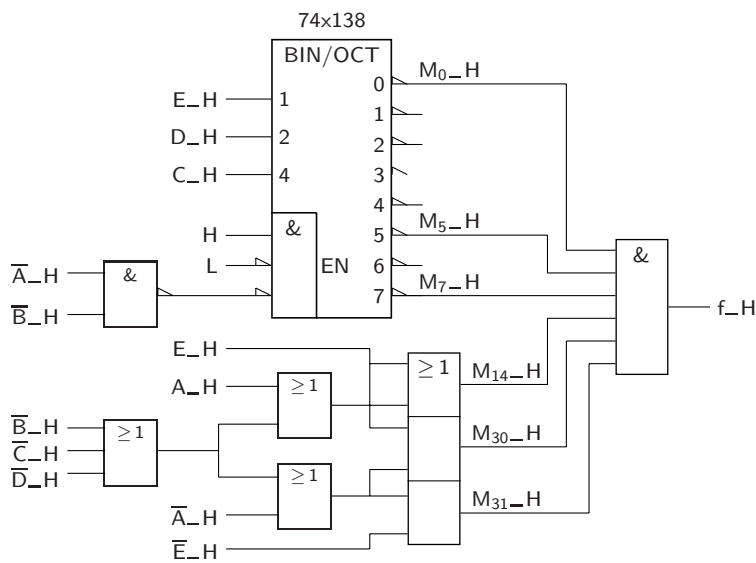


Figura 9.20: Logigrama que implementa a segunda forma canónica de $g(A, B, C, D, E)$, recorrendo a um descodificador do tipo 74x138 e a algumas portas adicionais

De notar a geração dos maxtermos activos a H nas saídas do descodificador (ver a resolução do Exercício 9.18).

Capítulo 10

Multiplexers e Demultiplexers

10.1 Desenhar a tabela de verdade física e o logigrama com a estrutura interna de um multiplexer com 4 entradas de dados, admitindo que as entradas de dados e a saída são activas a L, e que as entradas de selecção são activas a H.

Resolução: Ver a Figura 10.1 com o logigrama pedido.

10.1

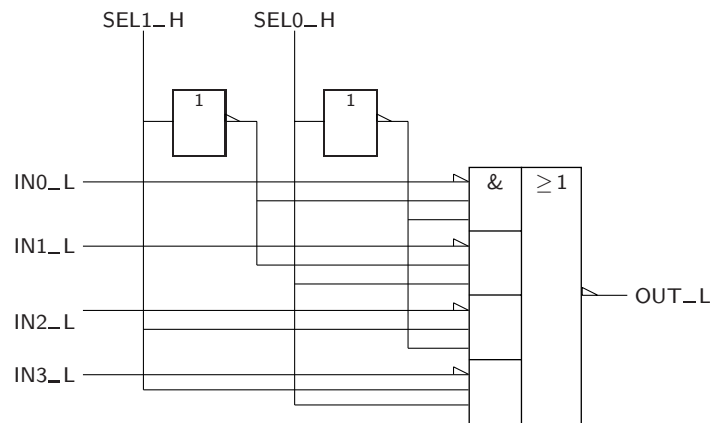


Figura 10.1: Logigrama com a estrutura interna do multiplexer do Exercício 10.1

É de notar que apenas foram introduzidas duas alterações em relação ao logigrama do multiplexer da Figura 10.3 de *SD:AAT*, que tem todas as entradas e a saída activas a H:

1. mudaram-se os níveis de actividade das entradas de dados e da saída para L, como pretendido pelo enunciado; e
2. mudaram-se os níveis de actividade das entradas dos ANDs ligadas às entradas de dados, e da saída do OR ligada à saída do multiplexer.

O objectivo destas alterações consistiu em pôr as entradas e saídas das portas mencionadas no ponto 2 com o mesmo nível de actividade das entradas de dados e da saída do multiplexer, o que, como sabemos, não altera a funcionalidade do circuito.

Naturalmente, para *implementar em lógica positiva* o logigrama da Figura 10.1, deparamo-nos com a dificuldade de necessitarmos de portas AND com duas entradas activas a H e uma a L, o que, como sabemos, não existe. Então, se quisermos gerar o esquema eléctrico nessa lógica, teremos de substituir os níveis de actividade a L nas entradas dos ANDs por conversores de polaridade.

O logigrama da Figura 10.2 sugere essas alterações.

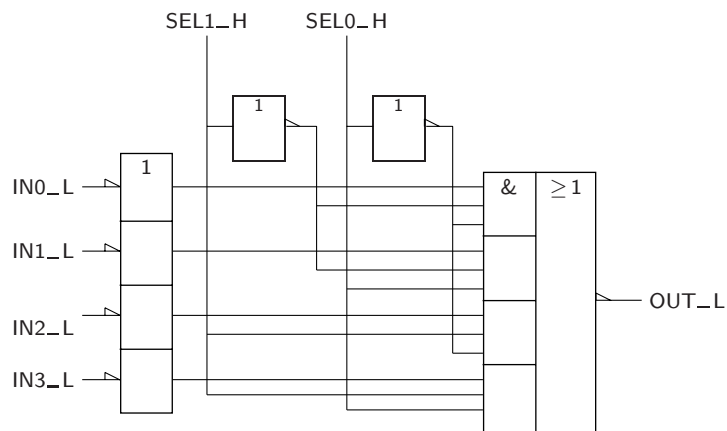


Figura 10.2: Logigrama alternativo para o multiplexer do Exercício 10.1, em que as entradas dos ANDs são agora todas activas a H

Quanto à tabela de verdade física deste multiplexer, podemos obtê-la imediatamente a partir de um dos dois logigramas anteriores, como se ilustra na Tabela 10.1. Basta, para tanto, atender à funcionalidade do multiplexer e aos níveis de actividade das entradas e da saída.

Tabela 10.1: Tabela de verdade física do multiplexer das Figuras 10.1 e 10.2

IN3_L	IN2_L	IN1_L	IN0_L	SEL1_H	SEL0_H	OUT_L
×	×	×	L	L	L	L
×	×	×	H	L	L	H
×	×	L	×	L	H	L
×	×	H	×	L	H	H
×	L	×	×	H	L	L
×	H	×	×	H	L	H
L	×	×	×	H	H	L
H	×	×	×	H	H	H

Por exemplo, quando *SEL1* e *SEL0* estão ambas inactivas (nas primeiras duas linhas da tabela), o multiplexer copia para a saída *OUT_L* o nível de tensão

que se encontrar aplicado à entrada *IN0_L*. Com efeito, neste caso a soma das potências de 2 correspondentes às entradas de selecção que estão activas é igual a 0.

Para as restantes linhas da tabela aplicaríamos o mesmo raciocínio. Por exemplo, se *SEL1* está inactiva mas *SEL0* está activa (as terceira e quarta linhas da tabela), o multiplexer copia para a saída *OUT_L* o nível de tensão que estiver aplicado à entrada *IN1_L*, porque agora a soma das potências de 2 correspondentes às entradas de selecção activas é igual a 1.

10.2 Desenhar o símbolo IEC de um multiplexer idêntico ao do da Figura 10.4 (de *SD:AAT*), mas com saída “tri-state”.

Resolução: Ver a Figura 10.3 com o símbolo IEC pedido. 10.2

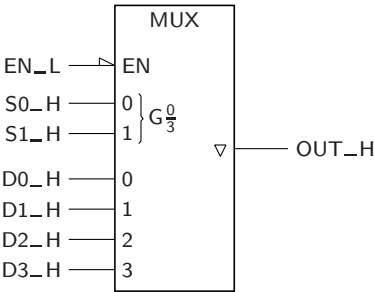


Figura 10.3: Símbolo IEC do multiplexer da Figura 10.4 de *SD:AAT*, mas com saída “tri-state”

10.3 Escrever a tabela de verdade física do multiplexer do exercício anterior.

Resolução: Ver a Tabela 10.2. 10.3

Tabela 10.2: Tabela de verdade física do multiplexer do exercício anterior

D3_H	D2_H	D1_H	D0_H	S1_H	S0_H	EN_L	OUT_H
×	×	×	×	×	×	H	Hi-Z
×	×	×	L	L	L	L	L
×	×	×	H	L	L	L	H
×	×	L	×	L	H	L	L
×	×	H	×	L	H	L	H
×	L	×	×	H	L	L	L
×	H	×	×	H	L	L	H
L	×	×	×	H	H	L	L
H	×	×	×	H	H	L	H

De notar que, quando a entrada de Enable está inactiva, a saída do multiplexer fica em alta impedância (representada por Hi-Z). Pelo contrário, quando a en-

trada de Enable está activa, o multiplexer desempenha a sua função normal de selecção de uma entrada.

10.4 Desenhar o logigrama de um multiplexer com uma estrutura em árvore e com 16 entradas de dados, formado por um primeiro nível com multiplexers de 2 entradas de dados e um segundo nível formado por um multiplexer com 8 entradas de dados. Admitir que as entradas e a saída são todas activas a H.

10.4

Resolução: Ver a Figura 10.4.

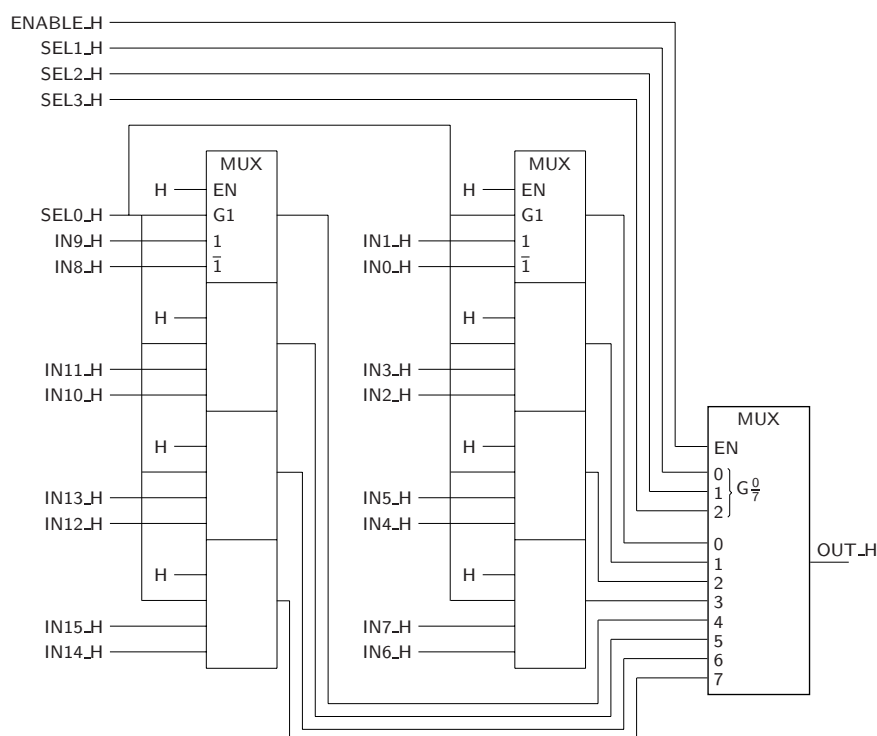


Figura 10.4: Logigrama de um MUX com 16 entradas de dados, realizado à custa de uma árvore com dois níveis formada por um multiplexer de 8 entradas de dados na raiz e por oito multiplexers de 2 entradas de dados nas folhas

74x251

10.6 Diga como pode ligar dois multiplexers como o da Figura 10.20 (de *SD:AAT*), do tipo 74x251, de modo a construir um multiplexer com 16 entradas e 1 saída. Use a lógica discreta suplementar que entender necessária.

10.6

Resolução: Ver a Figura 10.5 com o logigrama pedido.

Quando *S3* está activa (a H), é feito o Enable do multiplexer de baixo. Quando *S3* está inactiva (a L), é a vez do multiplexer de cima ficar Enabled. As variáveis *S2*, *S1* e *S0* seleccionam a entrada dos dois multiplexers, simultaneamente.

Como apenas um dos multiplexers está Enabled de cada vez, o outro vê a saída em alta impedância. Por consequência, apenas uma entrada de dados *IN0* a *IN15* pode, de cada vez, aparecer em *OUT*. Como as entradas de dados

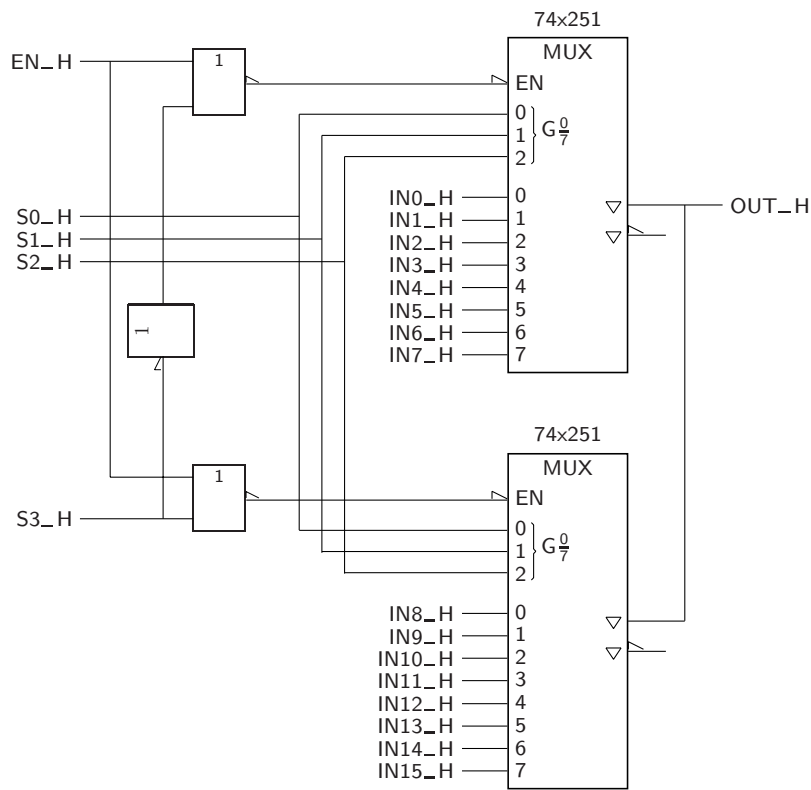


Figura 10.5: Logigrama com o multiplexer pedido no Exercício 10.6

e a saída têm a mesma polaridade (H), o nível de tensão aplicado à entrada seleccionada aparece, sem alteração, na saída.

Capítulo 12

Latches

12.7 Na Tabela 12.3 (de *SD:AAT*), identificar com setas adequadas a transição ou transições entre linhas da tabela que correspondem à situação em que se tem $S_H = R_H = H$, com $Q_H = L$ e $Q_L = L$, e em que se provoca a transição simultânea para L de S_H e R_H , admitindo que os tempos de propagação das portas são diferentes. O que podemos deduzir em relação ao estado final do latch?

Resolução: Ver a Tabela 12.1.

12.7

Tabela 12.1: Tabela de verdade física de um latch SR onde se acrescentam possíveis transições entre estados do circuito; as transições começam no estado $(Q, \overline{Q}) = (L, L)$ e conduzem a um estado final que não podemos prever

S_H	R_H	Q_H _(t+Δt)	Q_L _(t+Δt)
L	L	Q_H _(t)	Q_L _(t)
L	H	L	H
H	L	H	L
H	H	L	L

Nesta tabela, construída a partir da Tabela 12.3 de *SD:AAT*, partimos da situação descrita na última linha, em que temos nas entradas $(S, R) = (H, H)$ e em que o estado do latch é $(Q, \overline{Q}) = (L, L)$, como pede o enunciado.

Se o OR que tem por entrada R é mais rápido do que o outro OR, então passamos a ter transitoriamente $(S, R) = (H, L)$ e o latch passa ao estado $(Q, \overline{Q}) = (H, L)$, na terceira linha (setas à esquerda). Naturalmente, o outro OR há-de mudar a sua saída de H para L, e então caímos então na situação final estabelecida no enunciado e identificada com a outra seta à esquerda, correspondente à primeira linha da tabela, em que $(S, R) = (L, L)$ e o latch mantém o estado anterior, isto é, o estado $(Q, \overline{Q}) = (H, L)$.

Consideremos agora a situação traduzida pelas setas da direita na tabela. Partimos da mesma situação inicial, com $(S, R) = (H, H)$ e $(Q, \overline{Q}) = (L, L)$. Agora,

porém, é o OR que tem por entrada S que é mais rápido, pelo que passamos a ter transitoriamente $(S, R) = (L, H)$ e o latch passa ao estado $(Q, \overline{Q}) = (L, H)$, na segunda linha. Quando, finalmente, o outro OR mudar a saída de H para L , passamos à primeira linha da tabela, em que $(S, R) = (L, L)$ e o latch mantém o estado anterior; só que, agora, o estado final é $(Q, \overline{Q}) = (L, H)$.

Em resumo, o estado final do latch não pode ser previsto nas condições enunciadas.

12.8 Completar o diagrama temporal da Figura 12.18 (de *SD:AAT*), representativo do funcionamento de um latch SR em determinadas condições de níveis de tensão nas entradas, admitindo que inicialmente $Q_H = H$ e $Q_L = L$, e que o tempo de propagação das duas portas é de 10 ns. Como se comporta o latch nestas condições?

12.8

Resolução: Ver a Figura 12.1.

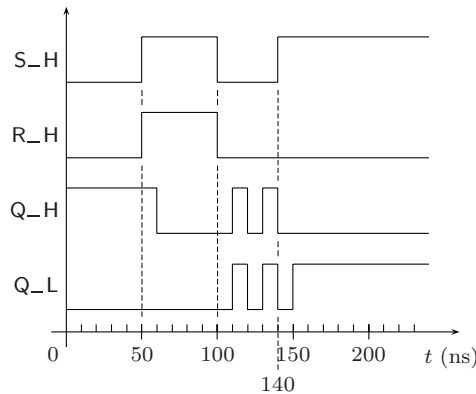


Figura 12.1: Diagrama temporal que descreve o comportamento de um latch SR em determinadas condições

Devemos chamar a atenção para duas situações.

1) Entre os instantes $t = 50$ ns e $t = 100$ ns está-se a tentar fazer o Set e o Reset *simultâneo* ao latch por activação das variáveis S e R . Como foi discutido no texto das teóricas, o latch não consegue responder a esta dupla imposição e força as suas duas saídas a L (a partir do instante $t = 60$ ns e enquanto esta situação se mantiver, até $t = 100$ ns).

2) Quando S_H e R_H passam *simultaneamente* de H para L no instante $t = 100$ ns, as portas OR vêm, nesse instante e simultaneamente, as suas duas entradas a L , pelo que irão colocar nas suas saídas o nível H no instante $t = 110$ ns. Nesse mesmo instante $t = 110$ ns as portas vêm uma das entradas a H e a outra a L , pelo que responderão colocando as saídas a L no instante $t = 120$ ns. E este ciclo repete-se a partir de $t = 120$ ns enquanto S_H e R_H se mantêm a L , isto é, até ao instante $t = 140$ ns.

Ou seja, as saídas do latch oscilam e mantêm-se iguais — isto é, não são complementares — entre $t = 110$ ns e $t = 140$ ns. A oscilação nas saídas dura todo o tempo em que as entradas S e R estiverem inactivas, depois de terem estado activas e sido desactivadas simultaneamente (e, não esquecer, admitindo que os

tempos de propagação das portas são *exactamente* iguais). A partir do instante $t = 140$ ns os níveis nas entradas mudam, ficando $S_H = H$ e $R_H = L$, e o latch faz o Set no instante $t = 150$ ns aí se mantendo até ao fim do diagrama temporal.

Na prática, é muito difícil a conjugação simultânea dos dois factores que conduzem à oscilação referida: S e R mudarem simultaneamente e os tempos de propagação das portas terem exactamente o mesmo valor. Se só ocorrer o primeiro factor (a mudança simultânea de S e de R) mas os tempos de propagação das portas forem diferentes, caímos na situação descrita no Exercício 12.7, em que não há oscilação mas *é impossível prever o estado final do latch*.

12.12 Considere o latch SR controlado da Figura 12.13 (de *SD:AAT*), com entradas assíncronas de Preset e de Clear activas a L. Mostre que o comportamento deste latch é o que se explicou no texto que acompanha a figura.

Resolução: Por comodidade, vamos repetir na Figura 12.2 a Figura 12.13 de *SD:AAT*, com o logigrama do latch. Acrescentam-se ao logigrama identificadores das portas lógicas.

12.12

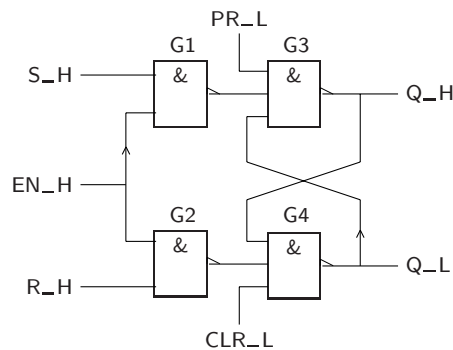


Figura 12.2: Logigrama de um latch SR controlado ao qual se acrescentaram entradas assíncronas de Preset e de Clear, activas a L, e identificadores das portas AND

Vamos agora estudar o comportamento deste latch.

Quando as variáveis PR e CLR estão inactivas, o latch comporta-se como um latch SR controlado normal. Ou seja, com a entrada EN desactivada o latch mantém o seu estado, e com a entrada EN activada o latch é um latch SR simples.

Consideremos agora que a entrada PR está activa e que a entrada CLR está inactiva. Nessas condições vem forçado na saída Q o nível H, independentemente dos níveis nas restantes entradas. Fez-se, então, o Preset assíncrono do latch.

Se, agora, tivermos a entrada CLR activa e a entrada PR inactiva, a situação vem mais complexa. Contudo, notemos que no texto se afirma que a colocação de um estado inicial no latch, por activação de uma das entradas assíncronas, deve ser conduzida com a entrada EN desactivada. O que faz todo o sentido, já que o latch não está, nessa situação inicial, a funcionar “normalmente”. Nessas

condições, a saída das portas G1 e G2 estão a H. Por outro lado, a activação da entrada CLR coloca a H a saída da porta G4. Segue-se, então, que a porta G3 tem todas as entradas a H, o que gera um L na saída Q. Fez-se, então, o Clear assíncrono do latch.

Uma forma alternativa de incluir entradas assíncronas de Preset e de Clear usa o logigrama da Figura 12.3.

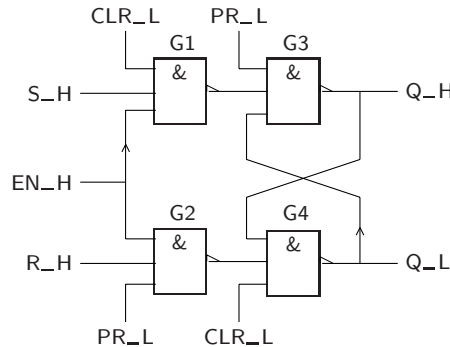


Figura 12.3: Logigrama alternativo para um latch SR controlado com entradas assíncronas de Preset e de Clear, activas a L

Neste caso, as entradas assíncronas actuam *independentemente* da entrada de Enable, o que de certa forma obvia ao “inconveniente” do latch anterior. Em resumo, podemos neste caso fazer o Preset ou o Clear do latch mesmo quando EN está activa, como facilmente se percebe pelo logigrama.

Por exemplo, a activação de PR força um H em Q e um H à saída da porta G2, independentemente do nível em EN. Nessas condições, a porta G4 tem todas as entradas a H, pelo que a saída \overline{Q} vem a L, uma situação que é estável.

E outro tanto acontece quando fazemos o Clear do latch.

Latch JK controlado

12.13

12.13 Considere a modificação da Figura 12.21 (de *SD:AAT*), que transforma um latch SR controlado num **latch JK controlado**. A ideia por detrás desta modificação é tentar obviar aos problemas levantados pelos latches SR controlados, apontados na página 200 de *SD:AAT* e nos Exercícios 12.7 e 12.8 (dificuldade em prever o estado final do latch ou o latch entrar em oscilação quando as entradas estão todas a H e se muda a entrada de Enable de H para L). Será que o latch JK controlado resolve esses problemas? Analise o seu funcionamento.

Resolução: Para facilitar a resolução deste exercício, vamos ilustrar na Figura 12.4 dois logigramas: (i) o logigrama do latch JK controlado da Figura 12.21 de *SD:AAT*; e (ii) uma outra “leitura” do latch, que resulta de considerarmos que ele é formado por um latch $\overline{S}\overline{R}$ controlado antecedido por duas portas AND (esta “leitura” vai facilitar a análise do seu comportamento).

Do logigrama do latch JK da Figura 12.4(b) podemos tirar as seguintes equações lógicas,

$$S = \overline{J \cdot \overline{Q}}$$

$$R = \overline{K \cdot \overline{Q}},$$

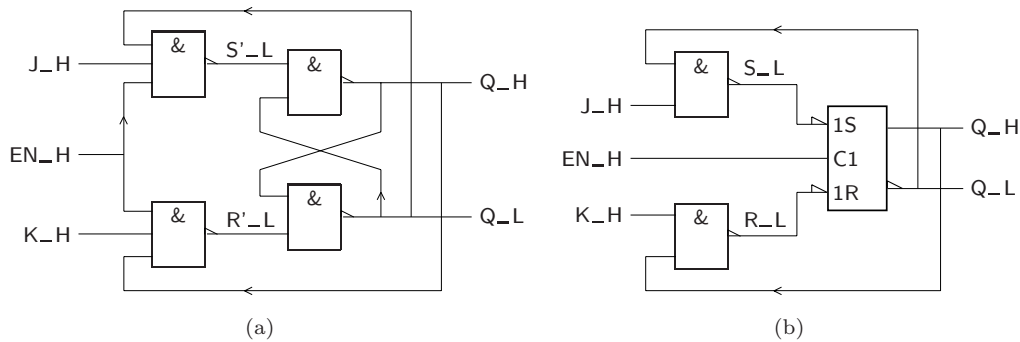


Figura 12.4: (a) Logigrama do latch JK controlado (da Figura 12.21 de *SD:AAT*); (b) Outra versão do logigrama

e delas deduzir a Tabela 12.2, em que se admite que a entrada de Enable está permanentemente activa. Se essa entrada alguma vez vier inactiva, o latch $\overline{S}\overline{R}$ mantém o seu estado e, por conseguinte, também o latch JK controlado o mantém.

Tabela 12.2: Tabela de verdade física que descreve o comportamento temporal de um latch JK controlado, admitindo que a entrada de Enable está sempre activa. A tabela é obtida por análise do circuito combinatório de entrada da Figura 12.4(b)

Linha	$J_H(t)$	$K_H(t)$	$Q_H(t)$	$Q_L(t)$	$S_L(t)$	$R_L(t)$	$Q_H(t+\Delta t)$
1	L	L	L	H	H	H	$Q_H(t)$
2	L	L	H	L	H	H	$Q_H(t)$
3	H	L	L	H	L	H	H
4	H	L	H	L	H	H	$Q_H(t)$
5	L	H	L	H	H	H	$Q_H(t)$
6	L	H	H	L	H	L	L
7	H	H	L	H	L	H	H
8	H	H	H	L	H	L	$Q_L(t)$

Analisemos a tabela. Na linha 1 temos $J_H=L$ e $Q_L=H$, pelo que $S_H=H$. Identicamente, temos $K_H=L$ e $Q_H=L$, pelo que $R_H=H$. Com $S_H=H$ e $R_H=H$, nem se faz o Set nem o Reset do latch $\overline{S}\overline{R}$, pelo que o latch JK mantém o seu estado.

Na linha 2 temos $J_H=L$ e $Q_L=L$, pelo que $S_H=H$. Identicamente, temos $K_H=L$ e $Q_H=H$, pelo que $R_H=H$. Mais uma vez, com $S_H=H$ e $R_H=H$ não se faz o Set nem o Reset do latch $\overline{S}\overline{R}$, pelo que o latch JK mantém o seu estado.

Estas duas linhas caracterizam-se, no seu conjunto, por terem $J_H = K_H = L$. Então, podemos afirmar que, para estes níveis nas entradas J e K, o latch JK controlado mantém o estado, como se indica na última coluna da tabela.

Se, agora, passarmos às linhas 3 e 4, concluímos que $Q_H(t+\delta t) = H$ para a primeira e que o latch mantém o estado, $Q_H(t+\Delta t) = Q_H(t) = H$, para a segunda. Ou seja, em ambos os casos faz-se o Set do latch JK controlado, como se indica na última coluna da tabela.


Comutação

Da mesma forma poderíamos analisar os dois pares de linhas seguintes, concluindo-se pelos valores que se indicam na última coluna da tabela. Podemos observar, então, que o latch JK controlado se comporta como um latch SR controlado (no qual teve origem) *excepto* quando $J_H = K_H = H$, porque nesse caso o latch muda o seu estado (dizemos que o latch **comuta**).

Então, deduzimos finalmente a tabela de verdade física deste latch (Tabela 12.3), o que completa o processo de análise ao seu funcionamento.

Tabela 12.3: Tabela de verdade física do latch JK controlado

EN_H	J_H	K_H	Q_H(t+Δt)	Q_L(t+Δt)	Função
L	×	×	Q_H(t)	Q_L(t)	Manutenção
H	L	L	Q_H(t)	Q_L(t)	Manutenção
H	L	H	L	H	Reset
H	H	L	H	L	Set
H	H	H	Q_L(t)	Q_H(t)	Comutação



O comportamento deste latch sofre de um inconveniente muito sério *que impede a sua utilização em condições normais*. Com efeito, consideremos a situação em que $J_H = K_H = H$ e $Q_H = L$. Quando aplicamos um impulso positivo à entrada de Enable, o latch comuta, passando Q a H, como mostra a sua tabela de verdade física ou a linha 7 da Tabela 12.2. Naturalmente, a mudança em Q ocorre Δt depois de ter aparecido o flanco ascendente do impulso em EN (Figura 12.5), em que Δt é o tempo de propagação através das duas portas AND em série da Figura 12.4(a).

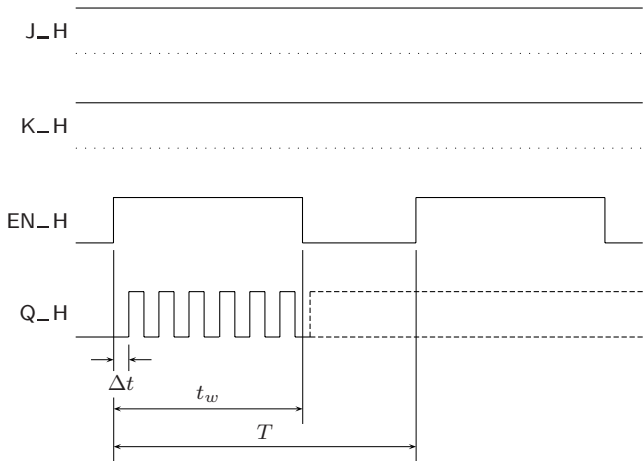


Figura 12.5: Diagrama temporal onde se mostra a *corrida* que ocorre quando $J_H = K_H = H$ e Q_H muda de L para H

Nesse instante temos $J_H = K_H = Q_H = H$ e, se a entrada de EN ainda estiver activa, o latch *volta a comutar*, passando Q novamente a L após Δt , como mostra a tabela de verdade física do latch ou a linha 8 da Tabela 12.2. E este processo repete-se enquanto EN estiver activa. Ou seja, *enquanto a entrada de Enable estiver activa o latch oscila*.

Este fenómeno de oscilação designa-se por **corrida** e é típica do funcionamento dos **circuitos sequenciais assíncronos** (que não estudamos), que é o que este circuito é.

A única forma de evitar esta corrida consiste em fazer-se $t_w < \Delta t < T$. Porém, com os circuitos integrados actuais os tempos de propagação Δt são muito reduzidos, em geral muito mais curtos do que a duração t_w do impulso na entrada. Então, na prática esta desigualdade *não vem satisfeita*, e o estado final do latch é, nas circunstâncias descritas acima, indeterminado.

Há, contudo, algumas alternativas que podemos contemplar para evitar corridas no circuito. A primeira consiste em utilizar um flip-flop JK, como estudaremos no capítulo seguinte. A segunda consiste em incluir na entrada de Enable um circuito que encurte o impulso de entrada para valores inferiores a Δt . A terceira é incluir **linhas de atraso** em série com as realimentações, por forma a aumentar Δt e torná-lo maior do que t_w .

Pela razões apontadas este latch não tem utilidade prática, preferindo-se, em geral, optar pela solução mais simples e económica de o substituir por um flip-flop JK.

Este fenómeno de oscilação não pode ser apercebido directamente na Tabela 12.2 porque ela foi deduzida do circuito combinatório de entrada do latch, sem levar em consideração as realimentações das saídas para as entradas.

Corrida

Circuitos sequenciais assíncronos

Linha de atraso

Capítulo 13

Flip-flops

13.1 Identificar, na Figura 13.4 (de *SD:AAT*), os instantes ou os intervalos de tempo em que ocorre o fenómeno de “one’s catching”.

Resolução: Vamos repetir a Figura 13.4 de *SD:AAT* na Figura 13.1 — para facilitar a análise do diagrama temporal — acrescentando-lhe algumas identificações temporais (instantes t_0 a t_5).

13.1

Reparemos agora no pequeno impulso positivo (**pico**) que ocorre na entrada K sensivelmente a meio do logigrama, entre t_1 e t_2 , e admitamos que esse impulso “não devia estar lá”. Ou seja, admitamos que o que se pretendia era que K se mantivesse a L até ao impulso final entre t_4 e t_5 .

Pico

Se tal acontecesse, o “master” não mudaria as suas saídas no instante t_1 , e estas manter-se-iam a $Q'_H = H$ e $Q'_L = L$ enquanto durasse $CP_H = H$ (intervalo de tempo entre t_0 e t_3). E, nessas condições, o “master” não passaria o seu estado L para o “slave” no instante t_3 .

Ou seja, o efeito do pico fez-se sentir nas saídas do flip-flop, que o “agarrou” (daí o nome “one’s catching” dado a este fenómeno característico dos flip-flops master-slave).

13.2 Desenhar o símbolo IEC de um flip-flop SR master-slave sem entradas assíncronas de Set e de Reset.

Resolução: Ver a Figura 13.2.

13.2

Na parte (a) da figura representa-se um flip-flop que comuta nos flancos descendentes.

Símbolo de um flip-flop
SR master-slave

Não esquecer que, num flip-flop master-slave, o instante a partir da qual a entrada C1 vem activada (isto é, neste caso imediatamente após um flanco ascendente de um impulso de relógio, quando C1 vem a H) *não é* o instante em que o flip-flop comuta (muda de estado, se tiver de mudar). Com efeito, o **símbolo de atraso** colocado junto às saídas (ou seja, o **qualificador de saída** \neg) indica que o flanco de comutação *é o que se segue imediatamente*, que é como quem diz, o flanco descendente.

Símbolo de atraso
Qualificador de saída \neg

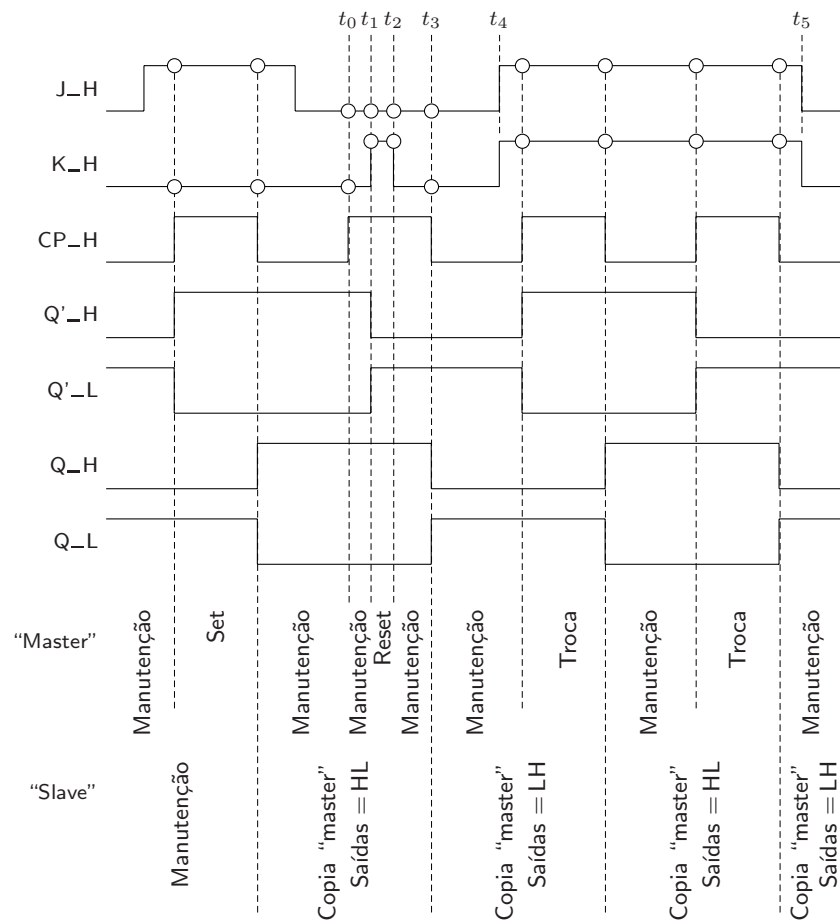


Figura 13.1: Diagrama temporal com o comportamento de um flip-flop JK master-slave que ilustra o fenómeno de “one’s catching”

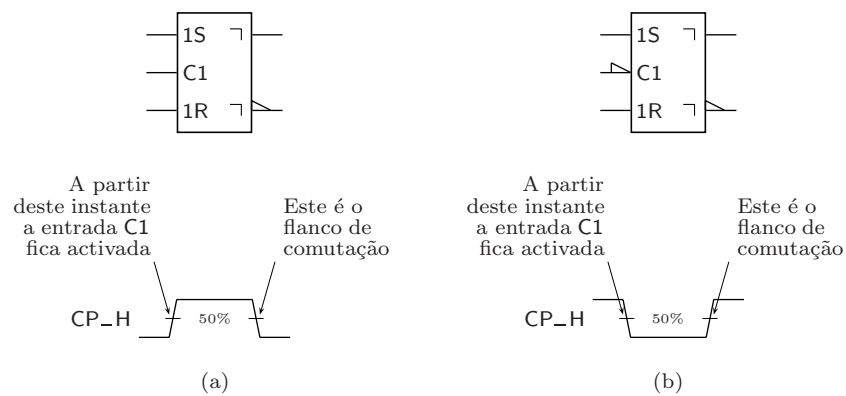


Figura 13.2: (a) Símbolo IEC de um flip-flop SR master-slave sem entradas assíncronas que comuta nos flancos descendentes; (b) flip-flop semelhante mas que comuta nos flancos ascendentes

Na parte (b) da figura representa-se um flip-flop que comuta nos flancos ascendentes. Não esquecer que, agora, a entrada C1 vem activada imediatamente após ficar a L, ou seja, imediatamente a seguir a um flanco descendente de um impulso de relógio.

13.3 Desenhar o símbolo IEC de um flip-flop SR master-slave com entradas assíncronas de Set e de Reset (ou de Preset e de Clear). Todas as entradas, síncronas e assíncronas, devem ser activas a L. Descrever o funcionamento deste flip-flop com um diagrama temporal onde se realce o efeito das entradas síncronas e assíncronas.

Resolução: Ver a Figura 13.3.

13.3

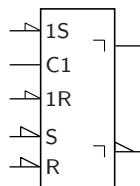


Figura 13.3: Símbolo IEC de um flip-flop SR master-slave com entradas assíncronas de Set e de Reset activas a L que comuta nos flancos descendentes

Notemos que nada nos obriga a considerar a entrada de relógio activa a L (já que não é uma entrada síncrona ou assíncrona, *é a entrada que controla as entradas síncrons*). Escolhemos, por isso e arbitrariamente, C1 activa a H, o que quer dizer que o flip-flop comuta nos flancos descendentes dos impulsos de relógio.

Notemos ainda a diferença entre as entradas síncronas e assíncronas.

As primeiras possuem os **qualificadores de entrada 1S e 1R**, que indicam um efeito de “disparo” pela entrada de relógio, C1 (**dependência de controlo**, C) Ou seja, os níveis de tensão nas entradas síncronas apenas são levados em consideração pelo flip-flop enquanto C1 estiver activada, a H.

Qualificadores de entrada 1S, 1R e C1
Dependência de controlo (C)

Pelo contrário, as entradas assíncronas possuem os **qualificadores de entrada S e R**. Estas entradas não dependem dos impulsos de relógio e actuam as saídas assim que estiverem activas (mas apenas uma de cada vez, ou nenhuma delas).

Qualificadores de entrada S e R

Na Figura 13.4 apresenta-se um diagrama temporal que ilustra um exemplo de funcionamento típico do flip-flop.

Começamos por admitir que os latches “master” e “slave” estão no estado H.

Entre t_0 e t_1 não há actividade nas entradas assíncronas, pelo que o “master” reage às entradas síncronas, que “ordenam” um reset do latch (não esquecer que as entradas síncronas são activas a L).

A partir de t_2 e até t_2 , o “slave” copia do “master” o estado L.

Entre t_2 e t_3 o “master” e o “slave” fazem um reset assíncrono porque a entrada S está activa (não esquecer que as entradas assíncronas também são activas a L).

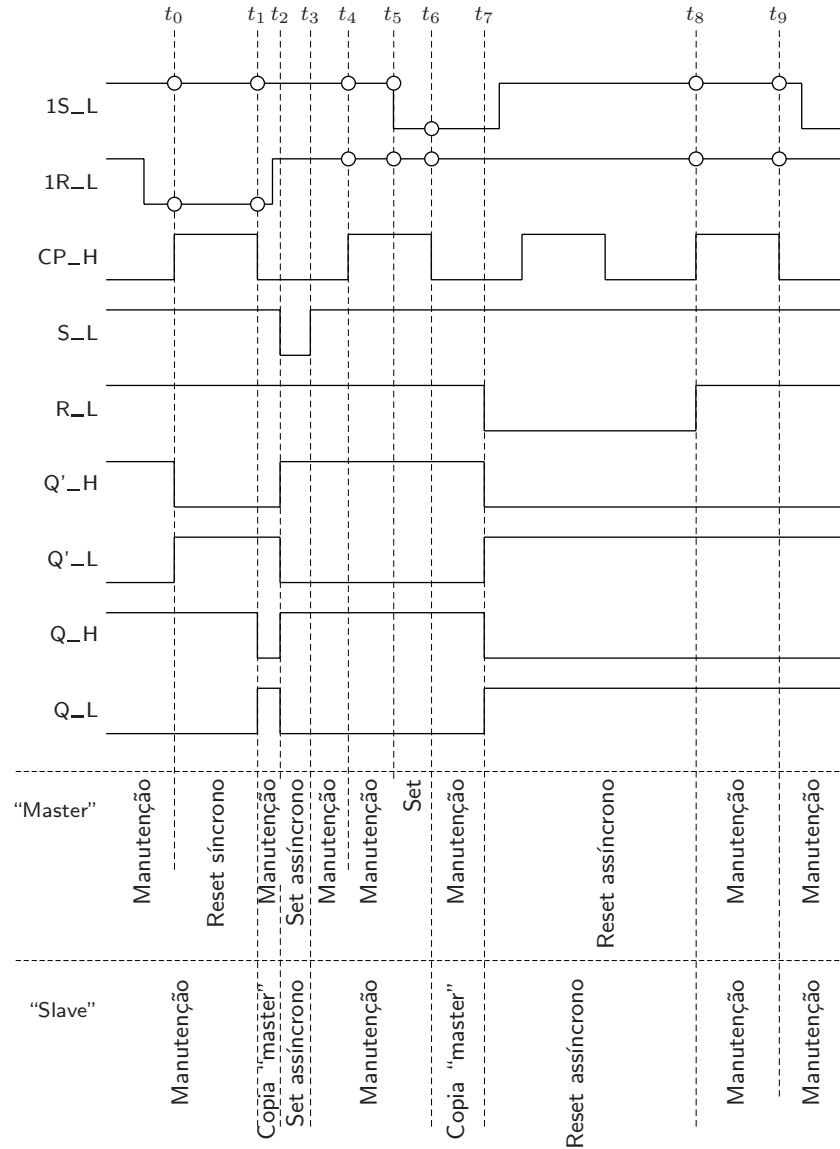


Figura 13.4: Diagrama temporal de funcionamento do flip-flop da figura anterior

Entre t_4 e t_6 o "master" está receptivo às entradas síncronas, que começam por manter o seu estado, entre t_4 e t_5 , em seguida, entre t_5 e t_6 , "ordenam" o set do "master".

Entre t_7 e t_8 existe um reset assíncrono do "master" e do "slave".

Em todos os restantes intervalos de tempo, os dois latches mantêm os respectivos estados.

13.4 Construir um flip-flop JK master-slave a partir de:

- a) um flip-flop D master-slave;
- b) um latch D controlado.

Resolução: a) Como se trata de construir um flip-flop a partir de outro que possui uma lógica de funcionamento diferente, vamos começar por estabelecer as tabelas de verdade (por exemplo, lógicas, mas podiam ser físicas) para os dois, como se mostra na Tabela 13.1.

13.4 a)

Tabela 13.1: Tabelas de verdade lógicas para os flip-flops do tipo JK e D

$J_H(t)$	$K_H(t)$	$Q_H(t+\Delta t)$	$D_H(t)$	$Q_H(t+\Delta t)$
0	0	$Q_H(t)$	0	0
0	1	0	1	1
1	0	1		
1	1	$\overline{Q_H(t)}$		

Para simplificar a tabela, apenas se indicam os níveis na linha de saída Q_H e omite-se a entrada de relógio, pressupondo que as mudanças de estado nessa saída apenas ocorrem, se ocorrerem, no flanco de comutação dos flip-flops (que, aliás, não vem imposto, pelo que o podemos escolher).

O problema consiste, então, em determinar quais os níveis que a entrada D tem de assumir para que o flip-flop se comporte como um JK, isto é, realize as transições características daquele tipo de flip-flop.

Escrevendo todas as configurações possíveis que um flip-flop JK pode ter nas linhas de entrada $J_H(t)$ e $K_H(t)$ e na linha de saída $Q_H(t)$, determina-se, a partir da tabela de verdade do flip-flop D, o valor correspondente a $Q_H(t+\Delta t)$ e, conseqüentemente, o valor que $D_H(t)$ terá de assumir:

$J_H(t)$	0	1	0	1	0	1	0	1
$K_H(t)$	0	0	1	1	0	0	1	1
$Q_H(t)$	0	0	0	0	1	1	1	1
$Q_H(t+\Delta t)$	0	1	0	1	1	1	0	0
$D_H(t)$	0	1	0	1	1	1	0	0

Desta tabela podemos deduzir as equações lógicas de $D_H(t)$ em função de $J_H(t)$, de $K_H(t)$ e de $Q_H(t)$. Para tanto estabelecemos o quadro de Karnaugh que se segue. Devemos notar que podemos escrever este quadro, dado estar em jogo um circuito combinatório. Com efeito, as variáveis booleanas estão todas definidas no mesmo instante, t .

	$J_H(t) \quad K_H(t)$			
$Q_H(t)$	00	01	11	10
0	0	0	1	1
1	1	0	0	1
	$D_H(t)$			

Do quadro de Karnaugh deduzimos a equação

$$D = J\overline{Q} + \overline{K}Q,$$

*Equação de excitação
de um flip-flop D*

também conhecida como **equação de excitação do flip-flop D**.

Falta agora implementar a parte correspondente ao modo de sincronização do flip-flop. Como o flip-flop D desta alínea é também do tipo master-slave, isto é, do mesmo tipo do flip-flop JK pretendido, não precisamos de ter qualquer preocupação com a questão do sincronismo.

Então, podemos obter o logigrama do flip-flop JK pretendido na Figura 13.5.

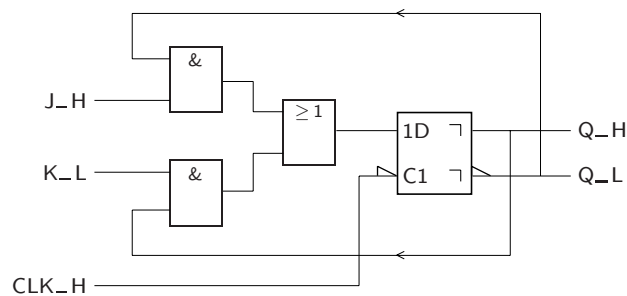


Figura 13.5: Flip-flop JK master-slave construído à custa de um flip-flop D, também do tipo master-slave

De notar que se optou por usar K_L numa das linhas de entrada, em vez de $\overline{K_H}$, o que obrigaria à existência de mais uma porta NOT.

13.4 b)

b) Neste caso, a resolução, no que se refere à equação de excitação do flip-flop, é idêntica à anterior.

No entanto, para que o flip-flop JK tenha um comportamento master-slave é necessário que as mudanças de nível nas saídas se dêem num flanco do impulso de relógio diferente do que é responsável pela excitação do flip-flop (flanco a partir do qual as entradas são avaliadas).

Para obter este comportamento a partir de um latch controlado, é necessário considerar dois latches numa estrutura master-slave, um que reage a um flanco e outro que reage ao outro flanco, como mostra a Figura 13.6.

De notar que se optou por usar dois latches diferentes, um com Enable activo a H e outro a L, para garantir os flancos diferentes que foram mencionados acima. Porém, é evidente que podíamos ter optado por usar dois latches iguais e por atacá-los com níveis diferentes de CLK. Nesse caso precisaríamos de uma porta NOT para controlar o segundo latch.

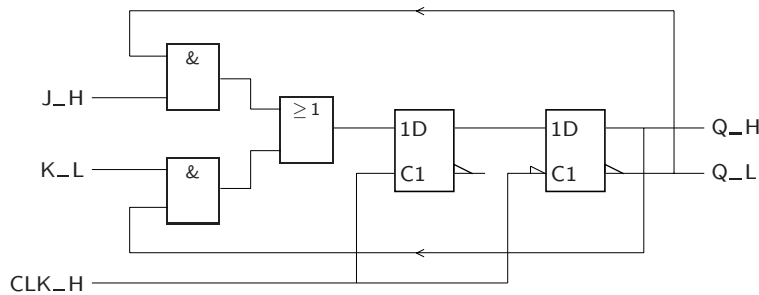


Figura 13.6: Flip-flop JK master-slave construído à custa de dois latches D controlados

13.6 O flip-flop hipotético A, do tipo edge-triggered, é obtido por transformação de um flip-flop JK do mesmo tipo como mostra a Figura 13.14 (de *SD:AAT*). Será que o flip-flop A é facilmente utilizável na prática, ou apresenta problemas?

13.6

Resolução: Vamos construir a tabela de verdade (física) do flip-flop A, para o que teremos de recorrer à tabela de verdade (física) de um flip-flop JK. É o que fazemos na Tabela 13.2.

Tabela 13.2: Tabelas de verdade físicas para os flip-flops do tipo JK e A

A_H _(t)	Q_H _(t)	J_H _(t)	K_H _(t)	Q_H _(t+Δt)
L	L	L	L	L
L	H	H	L	H
H	L	L	H	L
H	H	L	H	L

A partir desta tabela de verdade podemos construir a **tabela de excitações** do flip-flop A, como se indica na Tabela 13.3.

Tabela de excitações de um flip-flop A

Tabela 13.3: Tabela de excitações do flip-flop A

Q_H _(t) → Q_H _(t+Δt)	A_H _(t)
L → L	×
L → H	Impossível!
H → L	H
H → H	L

Verificamos que o flip-flop A não permite a transição de L para H, pelo que, se alguma vez ficar no estado L, nunca mais abandonará esse estado. Nestas circunstâncias, este flip-flop dificilmente poderá ser utilizado na prática, só o podendo em casos particulares em que a transição de L para H não seja necessária.

13.7 Para o circuito representado na Figura 13.15 (de *SD:AAT*), estabelecer o diagrama temporal da saída *S* entre t_0 e t_1 , admitindo que em t_0 se tem $(Q1, Q2, Q3) = (L, H, H)$.

13.7

Resolução: Ver a Figura 13.7.

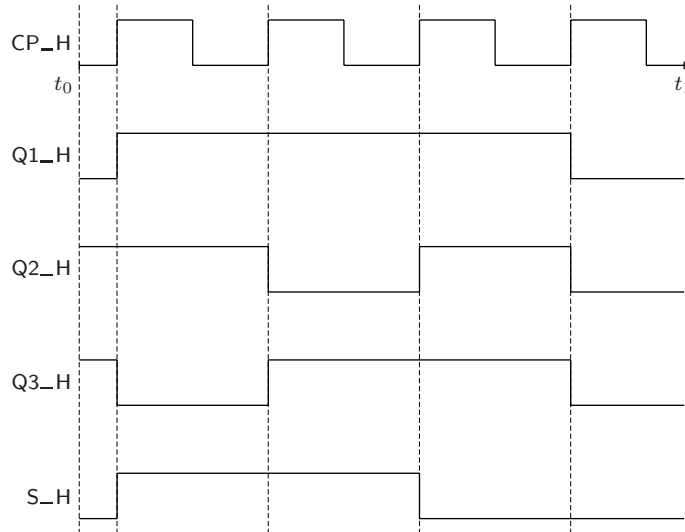


Figura 13.7: Diagrama temporal de funcionamento do circuito do Exercício 13.7

De notar que o diagrama temporal da figura não levou em consideração os tempos de propagação dos flip-flops.

13.8 Considere o circuito representado na Figura 13.16 (de *SD:AAT*) e admita que os flip-flops utilizados possuem $t_h = 5$ ns e $t_{su} = 4$ ns, e que as portas lógicas possuem o mesmo tempo de atraso, $t_{pd} = 10$ ns.

Analisando o circuito apresentado, e tendo em consideração as características indicadas, responda às seguintes perguntas.

- Qual o tipo de flip-flop utilizado?
- Qual o tempo de atraso mínimo de um flip-flop para que o circuito funcione correctamente? E qual é a frequência máxima de funcionamento do circuito nessas circunstâncias?

13.8 a)

Resolução: a) De acordo com a simbologia da norma IEC 60617-12, os flip-flops são do tipo edge-triggered e comutam nos flancos descendentes dos impulsos de relógio.

13.8 b)

b) O tempo de atraso (propagação) mínimo dos flip-flops tem de ser igual ao t_h dos mesmos, porque se fosse maior não poderíamos garantir que, por exemplo, o flip-flop do meio funcionasse correctamente. Com efeito, se o t_{pd} do flip-flop da esquerda fosse maior do que o t_h do flip-flop do meio, isso significaria que, quando o do meio ainda precisava da sua entrada estável, ela poderia já ter mudado porque é a saída do flip-flop da esquerda, que reagiria ao fim do seu tempo de propagação.

Consideremos, então, que $t_{\text{pd FF}} = t_h = 5 \text{ ns}$. Nestas circunstâncias, podemos calcular o período mínimo de relógio, que é igual ao somatório do $t_{\text{su FF}}$ com o $t_{\text{pd FF}}$ e ainda com o tempo de propagação mais crítico no circuito combinatório formado pelas duas portas que se encontram entre os flip-flops. Ora este último tempo é igual a 20 ns porque, na pior situação, o sinal tem que se propagar através das duas portas.

Temos, então, que o período mínimo entre impulsos consecutivos de relógio vem dado por

$$t_{\min} = 4 + 5 + 20 = 29 \text{ ns},$$

a que corresponde a frequência máxima de relógio

$$f_{\max} = \frac{1}{t_{\min}} = \frac{1}{29} = 33,3 \text{ MHz}.$$

13.14 O circuito da Figura 13.18 (de *SD:AAT*) é baseado num flip-flop D edge-triggered e constitui uma proposta de aproveitamento deste circuito para substituir uma porta NOT (!).

Com efeito, enquanto a variável *IN* de entrada está inactiva (a L), o Reset assíncrono do flip-flop actua e *OUT_L* fica a H. Por outro lado, quando *IN* muda de L para H, vem aplicado à entrada de relógio do flip-flop um flanco ascendente e, como a entrada *D* está activa, a função *OUT* de saída passa a L. Ou seja, aparentemente $OUT = \overline{IN}$.

Os parâmetros temporais do flip-flop estão indicados na figura: $t_{\text{su,D max}}$ é o tempo máximo de preparação da entrada D, $t_{\text{su,RS max}}$ é o tempo máximo de preparação das entradas R e S, t_h é o tempo de manutenção das entradas, $t_{\text{pd,RS-}\overline{Q} \text{ max}}$ é o tempo máximo de propagação desde as entradas R e S até à saída \overline{Q} , e $t_{\text{pd,C-}\overline{Q} \text{ max}}$ é o tempo máximo de propagação desde a entrada de relógio até à saída \overline{Q} .

Explique porque é que o circuito não funciona.

Resolução: O circuito é deveras engenhoso. O comportamento temporal parece correcto, como mostra a Figura 13.8.

13.14

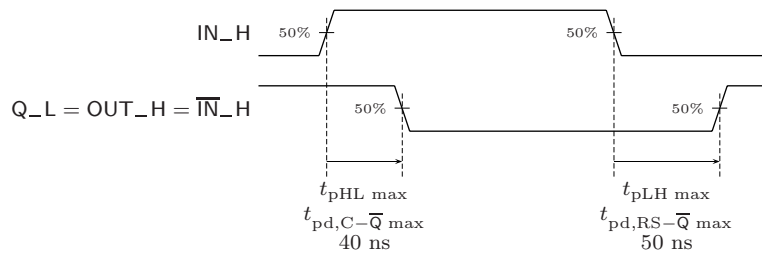


Figura 13.8: “Diagrama temporal” do circuito do Exercício 13.14

De notar os tempos máximos de propagação da “porta NOT”,

$$t_{\text{pHL max}} = t_{\text{pd,C-}\overline{Q} \text{ max}} = 40 \text{ ns}$$

e

$$t_{\text{pLH max}} = t_{\text{pd,RS-}\overline{\text{Q}} \text{ max}} = 50 \text{ ns}.$$

Só que o circuito não funciona correctamente, porque *vem violado* o tempo $t_{\text{su,RS max}} = 6 \text{ ns}$. Com efeito, quando aparece um qualquer flanco ascendente na entrada de relógio do flip-flop, a entrada R já devia estar estável 6 ns antes (o que não acontece, porque R muda simultaneamente com o flanco de relógio).

Reparemos, contudo, que este é o único parâmetro temporal que vem violado. Assim, o tempo de preparação das entradas S e D não vêm violados porque elas estão permanentemente ligadas a H (a única coisa que temos de assegurar é que *o primeiro flanco de relógio* ocorre pelo menos 20 ns depois de o circuito vir alimentado electricamente). Por outro lado, o tempo de manutenção também não vem violado porque é igual a 0 ns.

Capítulo 14

Contadores

14.2 Considere o logigrama da Figura 14.24 (de *SD:AAT*), constituído por um contador binário assíncrono e por um multiplexer com 8 entradas de dados.

a) Desenhe o diagrama temporal da saída F_H quando uma sequência de 10 impulsos é aplicada à entrada CLK_H . Suponha que inicialmente o contador tem o valor de contagem $0_{(10)}$. Não dê relevo à existência de estados instáveis nem de atrasos nos circuitos.

b) Entrando agora em conta com a existência de estados instáveis e de atrasos nos circuitos, desenhe o diagrama temporal pormenorizado — que inclua as variáveis $I2$, $I1$, $I0$ e a função F — das transições resultantes da passagem do estado de contagem $3_{(10)}$ para o estado de contagem $4_{(10)}$.

Resolução: a) Repare-se que temos um contador assíncrono ascendente com flip-flops do tipo edge-triggered a comutar nos flancos ascendentes (o que está bem, uma vez que podemos ter contadores ascendentes a comutar em qualquer dos tipos de flanco).

14.2 a)

A resolução deste exercício é muito simples. Para cada estado de contagem do contador (e o contador, sendo binário com 3 bits, conta de LLL a HHH), o multiplexer coloca na saída um dos valores aplicados às suas entradas de dados, mais concretamente o que corresponde ao referido estado de contagem. No nosso caso, como as entradas pares estão todas ligadas ao nível L e as ímpares ao nível H, os estados de contagem pares produzirão um L na saída da multiplexer, e os estados de contagem ímpares produzirão um H. Donde, o diagrama temporal da Figura 14.1.

b) A situação é, agora, mais complexa.

14.2 b)

No diagrama temporal da Figura 14.2 ilustram-se as formas de onda pedidas no caso da passagem do estado estável **3** para o estado estável **4**.

Note-se que, neste caso, se passa do estado de contagem LHH para o estado HLL à saída do contador, ou seja, mudam todos os flip-flops — o que gera o maior número possível de estados instáveis na transição. É claro, outra transição em que tal acontece é na transição do estado HHH ($7_{(10)}$) para o estado LLL ($0_{(10)}$). Mas já a transição do estado LLH ($1_{(10)}$) para LHL ($2_{(10)}$), por exemplo, apenas

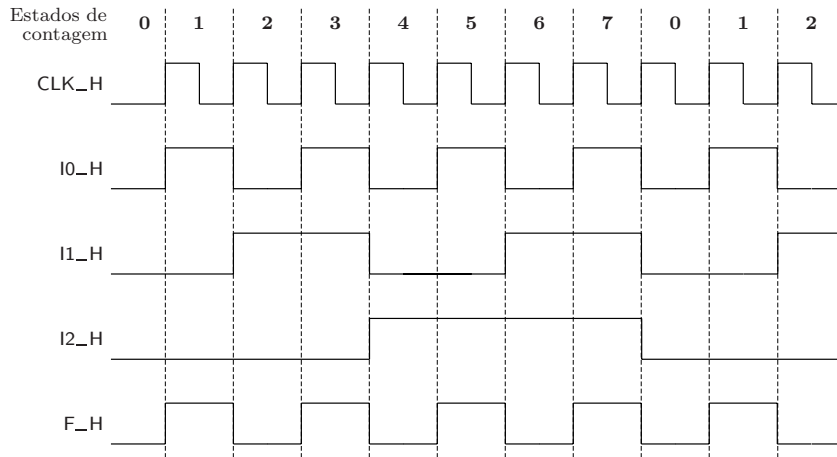


Figura 14.1: Diagrama temporal de funcionamento do circuito do Exercício 14.2

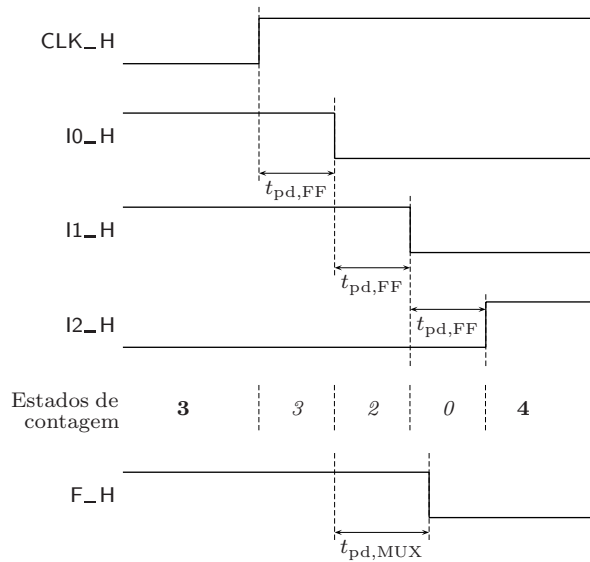


Figura 14.2: Diagrama temporal parcial de funcionamento do circuito do Exercício 14.2, quando se considera a existência de estados instáveis e de atrasos nos circuitos

faz mudar dois flip-flops. E a transição do estado LLL ($0_{(10)}$) para LLH ($1_{(10)}$), por exemplo, apenas faz mudar um flip-flop.

Nas linhas $I0_H$ a $I2_H$ são postos em evidência os efeitos dos tempos de propagação dos flip-flops, simbolizados por $t_{pd,FF}$. Estes tempos manifestam-se porque o contador é assíncrono.

Note-se que se vai assistir à passagem do estado estável **3** ao estado estável **4** através, e por esta ordem, dos estados instáveis *3*, *2* e *0* (a itálico na figura). Isso significa que, em termos da saída F_H , se vai simplesmente assistir a uma passagem do nível H para o nível L quando se sair do estado ímpar (instável) *3*

para o estado par (instável) 2, continuando a L pelo estado par (instável) 0 e (estável) 4.

Essa mudança de nível surge, naturalmente, com um atraso $t_{pd,MUX}$ em relação à passagem de 3 para 2, que é o tempo de propagação das entradas de dados para a saída do multiplexer.

$t_{pd,MUX}$

14.4 Estabelecer o símbolo IEC do contador assíncrono integrado 74HCT393, fabricado em tecnologia HCMOS compatível com TTL. Esse contador é constituído por dois contadores/divisores de frequência independentes e idênticos, cada um de módulo 16 e com uma entrada de Reset activa a H. Os contadores contam ascendentemente a cada flanco descendente dos impulsos aplicados às suas entradas de relógio. Os flip-flops utilizadas nos contadores são do tipo edge-triggered.

74HCT393

Resolução: Dado que os contadores que compõem o 74HCT393 são totalmente independentes, na sua simbologia não se inclui bloco de controlo comum. Essa simbologia será então constituída por dois blocos independentes, um para cada contador parcial, com uma estrutura simplificada que apenas inclui a entrada de Reset, activa a H e com o qualificador CT=0, a entrada de relógio, com o qualificador + apostro à simbologia dos flip-flops edge-triggered, e os qualificadores de saída CT0 a CT3 (Figura 14.3).

14.4

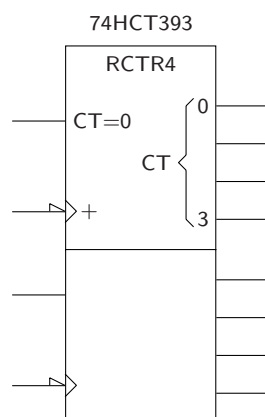


Figura 14.3: Símbolo do contador assíncrono binário 74HCT393, de acordo com a norma IEC

Finalmente, os 2 blocos possuem um qualificador geral RCTR4, que indica que cada um dos contadores é assíncrono e é constituído por 4 flip-flops.

Relembrar que apenas o bloco superior necessita de qualificadores, já que o bloco inferior “herda” todos os qualificadores do bloco que se encontra imediatamente por cima dele (apenas é necessário incluir qualificadores diferentes no bloco de baixo, se for caso disso — o que não acontece neste caso).

14.5 Estabelecer o símbolo IEC do contador assíncrono integrado 74HC4024, fabricado em tecnologia HCMOS. Trata-se de um contador/divisor de frequência por 128, com uma entrada de Reset activa a H. O contador conta ascendente-

74HC4024

mente a cada flanco descendente dos impulsos aplicados à entrada de relógio. Os flip-flops utilizadas nos contadores são do tipo edge-triggered.

14.5

Resolução: O símbolo IEC deste contador pode ser imediatamente obtido, como se indica na Figura 14.4.

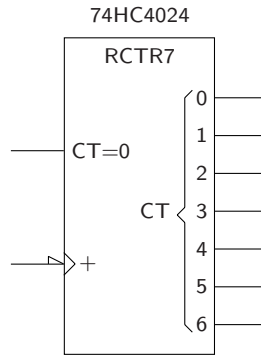


Figura 14.4: Símbolo do contador assíncrono 74HC4024, de acordo com a norma IEC

14.20 Utilizando um contador como o da Figura 14.28 (de *SD:AAT*), projecte um contador com a seguinte sequência de contagem:

$$\dots, 0, 1, 2, 3, 4, 5, 6, 7, 0, \dots$$

Indique ainda se optou por um carregamento em paralelo síncrono ou assíncrono, e justifique.

14.20

Qualificador de entrada
 $\bar{1}, 2D$

Resolução: O contador que é dado possui carregamento síncrono, como pode ser percebido pelo qualificador de entrada $\bar{1}, 2D$, que indica uma dependência de controlo do relógio C2. Logo, iremos utilizar carregamento em paralelo síncrono.

Repare-se que não é possível utilizar a entrada de Reset para forçar o estado de contagem $0_{(10)}$ porque o Reset é assíncrono. Com efeito, $CT=0$ não depende de C2. Se dependesse, o símbolo IEC teria $2CT=0$ em vez de $CT=0$.



Este problema insere-se num conjunto mais vasto de problemas em que é dado um contador integrado — que conta segundo um determinado código, em geral no CBN — e em que queremos, usando esse contador como peça base, obter um outro contador com uma sequência de contagem arbitrária. Este género de problemas pode ser resolvido, no caso geral, recorrendo ao diagrama de blocos da Figura 14.5 (ver também o Exercício 14.28).

Lógica de detecção de
estados

Estado actual (EA) de
um contador

Lógica de carregamento
em paralelo

Estado seguinte (ES) de
um contador

Neste diagrama distinguem-se duas lógicas combinatórias:

- uma **lógica de detecção de estados** que, a partir do **estado actual (EA) do contador**, traduzido pelos níveis de tensão às saídas dos seus flip-flops, obtém um sinal de carregamento em paralelo para o contador integrado; e
- uma **lógica de carregamento em paralelo** que, também a partir do estado actual do contador, decide qual a configuração binária (quantidade booleana geral) a carregar em paralelo, isto é, o **estado seguinte (ES) do contador**.

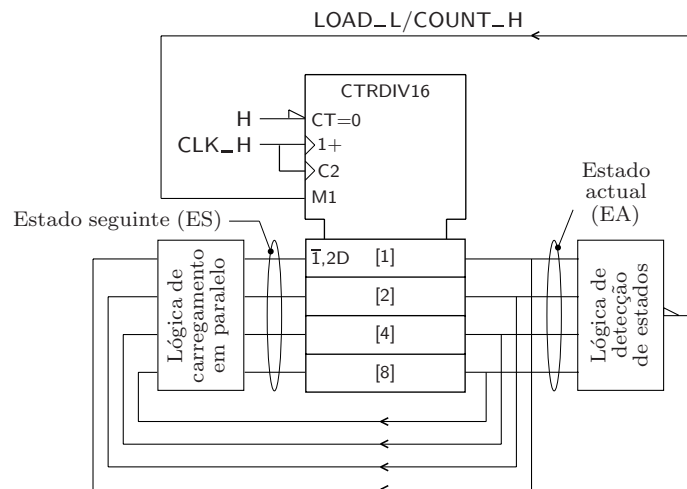


Figura 14.5: Diagrama de blocos de um contador que conta segundo uma sequência de contagem arbitrária e que utiliza um contador integrado que conta segundo o CBN

Quando a lógica de detecção de um determinado EA, digamos $n_{(10)}$, activa o sinal de carregamento em paralelo, o contador integrado carrega sincronamente o ES no contador (por exemplo, $k_{(10)}$). Enquanto o sinal de carregamento em paralelo estiver inactivo, o contador conta. Deste modo se consegue a sequência de contagem

$$\dots, k, k+1, k+2, k+3, \dots, n-2, n-1, n, k, k+1, \dots$$

para o contador global.

Como neste exercício se pretende a sequência de contagem

$$\cdots, 0, 1, 2, 3, 4, 5, 6, 7, 0, \cdots,$$

a lógica de detecção deve detectar o estado actual $n_{(10)} = 7$ e carregar o estado seguinte $k_{(10)} = 0$.

No caso da Figura 14.5 o carregamento em paralelo faz-se quando a entrada M1 estiver a L, pelo que o sinal de carregamento em paralelo deve ser activo a L e a lógica de detecção deve ter saída activa a L. Daí que uma designação semântica apropriada para o sinal seja *LOAD_L*. Naturalmente, porque o contador integrado conta quando não carrega em paralelo, outra designação semanticamente correcta será *COUNT_H*.

Finalmente, notemos na Figura 14.5 como a entrada de Reset está permanentemente desactivada.

Obtemos, então, o logigrama da Figura 14.6,

14.25 Utilize um contador binário síncrono de módulo 16 com Reset e carregamento em paralelo síncronos, para realizar um contador com a sequência de contagem

$\cdots, 0, 1, 2, 3, 4, 5, 0, 1, \cdots,$

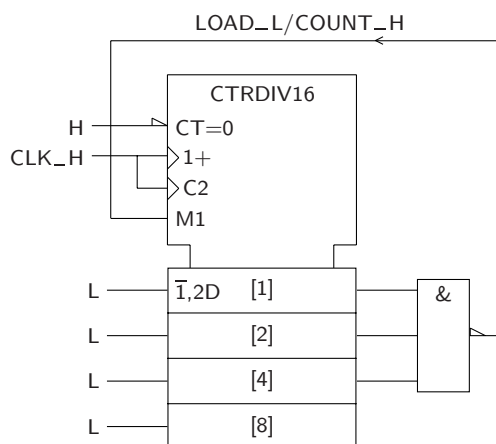


Figura 14.6: Logigramma do contador do Exercício 14.20

isto é, com módulo 6. Desenhe um diagrama temporal que mostre o funcionamento do contador. Modifique o logigramma que obteve para acomodar um contador integrado do tipo 74LS163A como o da Figura 14.31 (de *SD:AAT*).

74LS163A

14.25

Resolução: Como o contador em questão possui *Reset síncrono*, podemos modificar o modo de actuação que utilizámos no Exercício 14.20: em vez de fazer um carregamento em paralelo após detecção do estado 5, podemos fazer o Reset do contador após detecção desse estado. A vantagem desta metodologia é que não necessitamos de lógica de carregamento em paralelo (por muito simples que ela fosse neste caso, já que bastaria carregar LLLL nas entradas de carregamento em paralelo do contador). Obtemos, assim, o diagrama de blocos da Figura 14.7.

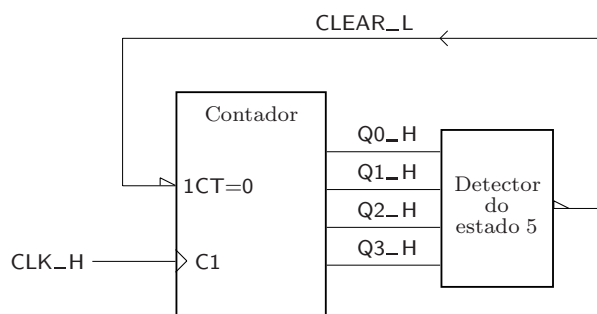


Figura 14.7: Diagrama de blocos do contador do Exercício 14.25

Quando o estado de contagem 5 é detectado, gera-se um sinal *CLEAR_L* e a entrada de Reset síncrono do contador vem activada, recomeçando a contagem a partir do estado 0. O diagrama temporal de funcionamento deste contador vem representado na Figura 14.8.

Notemos que o detector do estado 5 gera um sinal *CLEAR* activo a L só durante as ocorrências desse estado; com efeito, o sinal fica activado logo após o contador entrar no estado 5 ($t_{pd,DET}$ é o tempo de propagação do detector) e fica desactivado quando o contador passar ao estado 0. Notemos ainda que, enquanto

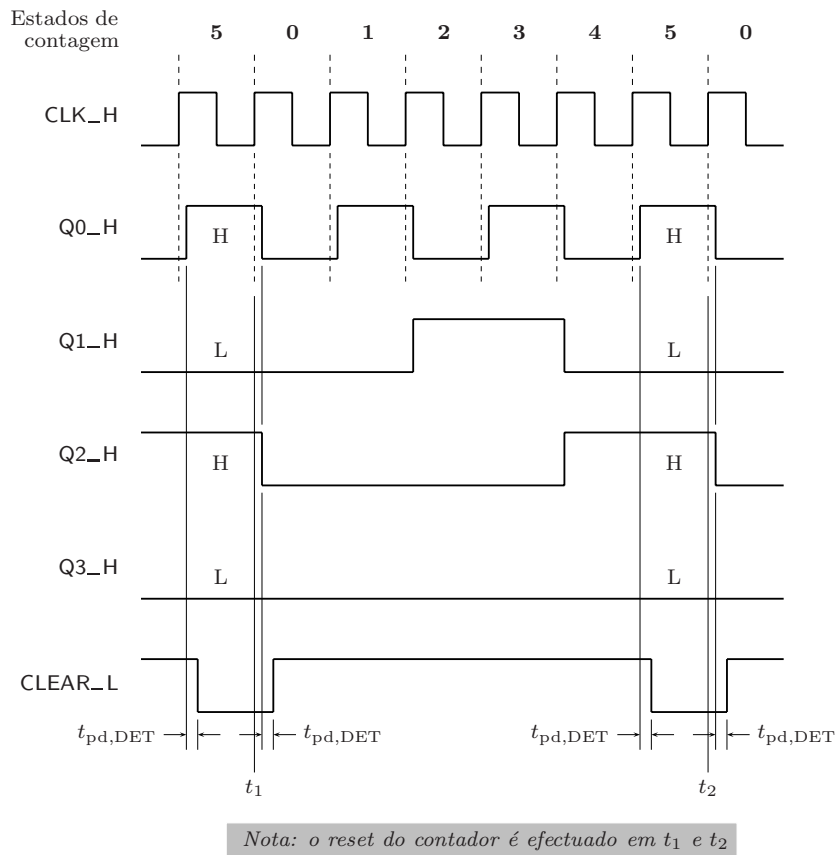


Figura 14.8: Diagrama temporal de funcionamento do contador da Figura 14.7

CLEAR está activo, ocorrem flancos ascendentes de *CLK_H* nos instantes t_1 e t_2 . O Reset do contador, sendo síncrono, só vem então efectivado em t_1 e t_2 , já que o Reset só pode ocorrer quando a entrada correspondente estiver a L e aparecer um flanco ascendente em *CLK_H*.

Podemos finalmente obter, na Figura 14.9, o logigrama do contador pretendido, admitindo que se utiliza um contador integrado em tecnologia TTL do tipo 74LS163A. Notemos que o detector do estado 5 pode vir simplificado neste caso já que o contador apenas passa pelos estados de contagem indicados na Tabela 14.1 (contudo, ver o Exercício 14.29).

14.26 Repetir o Exercício 14.25 mas utilizando agora um contador com Reset assíncrono.

Resolução: Agora já não podemos detectar o estado 5 para forçar um Reset do contador, como no exercício anterior. Precisaremos, pelo contrário, de um detector do estado 6, que irá gerar no contador um estado instável 6 na transição do estado estável 5 para o estado estável 0.

O aparecimento de um estado instável num contador síncrono resulta exclusivamente do facto de a função de Reset ser *assíncrona*. Com efeito, sempre que se

14.26

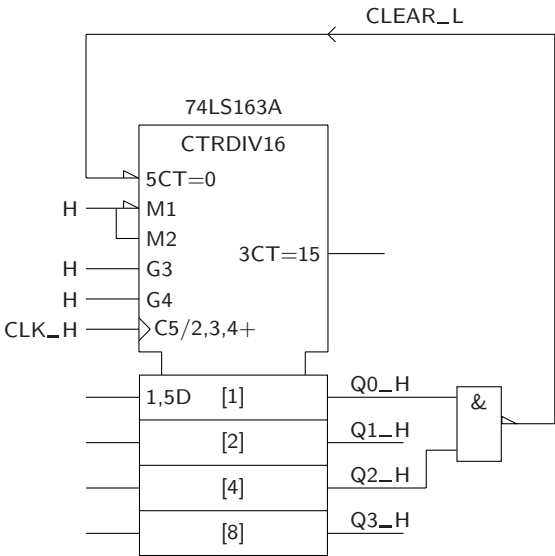


Figura 14.9: Logigrama do contador do Exercício 14.25, admitindo que se utiliza um contador integrado do tipo 74LS163A

Tabela 14.1: Tabela de estados de contagem do contador do Exercício 14.25

Q ₃ –H	Q ₂ –H	Q ₁ –H	Q ₀ –H
L	L	L	L
L	L	L	H
L	L	H	L
L	L	H	H
L	H	L	L
L	H	L	H

implementa uma função assíncrona num circuito síncrono (seja ele ou não um contador), o aparecimento de um ou mais estados instáveis torna-se inevitável.

O diagrama de blocos do contador é então o que se indica na Figura 14.10 e o diagrama temporal correspondente está representado na Figura 14.11.

Por seu turno, na Figura 14.12 representa-se a expansão do diagrama temporal anterior na transição do estado estável **5** para o estado estável **0**.

Em relação a esta última figura notemos que, após um intervalo de tempo $t_{pd,DET}$ depois de o contador entrar no estado instável 6, em que $t_{pd,DET}$ é o tempo de propagação do detector, o sinal *CLEAR* vem activado, e t_{CLEAR} depois é feito o Reset ao contador. O intervalo de tempo t_{CLEAR} é medido desde a entrada CT=0 de Reset do contador até às saídas dos flip-flops.

Para finalizar este exercício, podemos obter, na Figura 14.13, o logigrama do contador pretendido, admitindo que se utiliza um contador integrado do tipo 74LS161A com Reset assíncrono.

74LS161A

Tal como acontecia no exercício anterior, também neste caso a detecção do

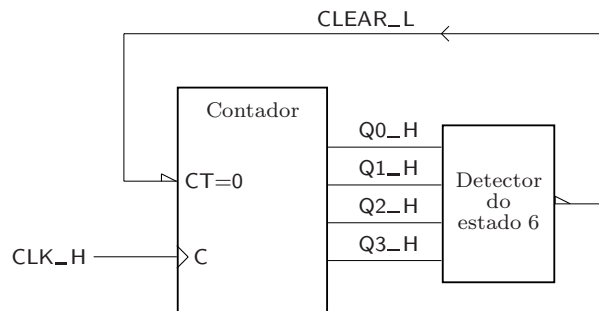


Figura 14.10: Diagrama de blocos do contador do Exercício 14.26

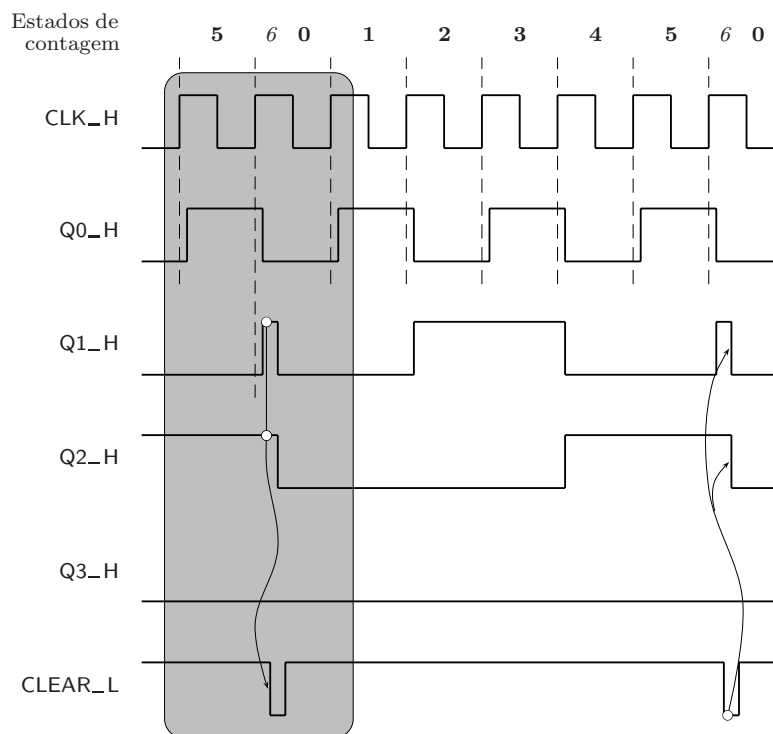


Figura 14.11: Diagrama temporal de funcionamento do contador do Exercício 14.26

estado 6 pode vir simplificada atendendo à Tabela 14.2 (contudo, ter em atenção o Exercício 14.29).

14.27 Utilizar um contador integrado do tipo 74LS163A para implementar uma década (contador com 10 estados), com sequência de contagem

$$\dots, 6, 7, 8, 9, \dots, 14, 15, 6, 7, \dots$$

Resolução: Nos Exercícios 14.20, 14.25 e 14.26 utilizaram-se sequências de con-

14.27

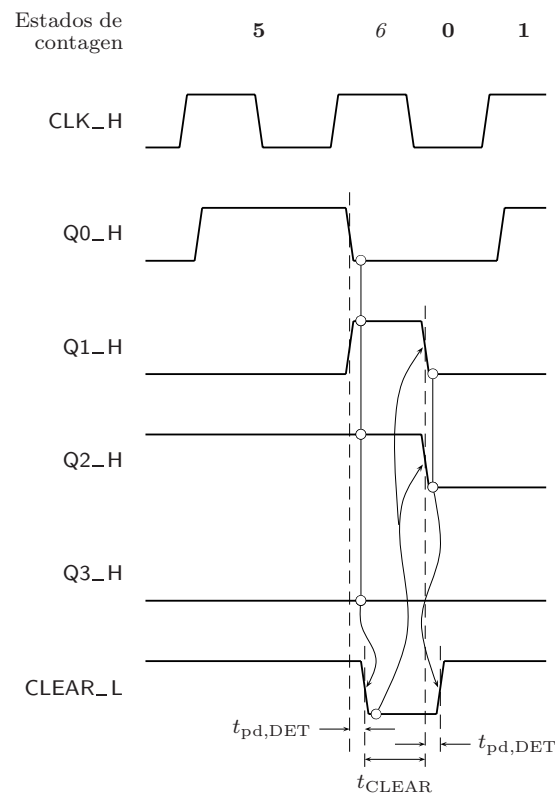


Figura 14.12: Expansão do diagrama temporal do contador do Exercício 14.26, na transição do estado estável **5** para o estado estável **0**

Tabela 14.2: Tabela de estados de contagem do contador do Exercício 14.26

Q ₃ _H	Q ₂ _H	Q ₁ _H	Q ₀ _H
L	L	L	L
L	L	L	H
L	L	H	L
L	L	H	H
L	H	L	L
L	H	L	H
L	H	H	L

tagem como

$$\dots, 0, 1, 2, 3, 4, 5, 6, 7, 0, \dots$$

ou

$$\dots, 0, 1, 2, 3, 4, 5, 0, 1, \dots,$$

em que o estado de início de contagem era sempre o estado 0 e o estado a detectar (na primeira situação o 7, e na segunda o 5) não era o estado final do contador integrado. No caso do carregamento ser assíncrono fazia-se a detecção

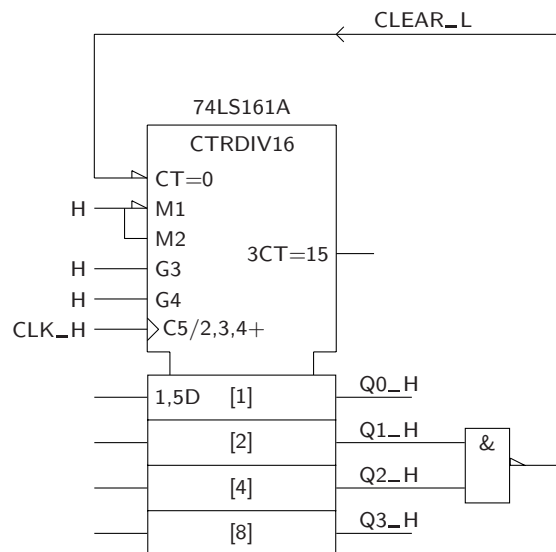


Figura 14.13: Logigramma do contador do Exercício 14.26, supondo que se utiliza um contador integrado do tipo 74LS161A

do estado que se seguia ao estado final pretendido, mas o princípio era o mesmo.

Nesses exercícios utilizou-se sempre um contador integrado que contaria segundo o CBN, desde LLLL até HHHH, se fosse deixado a contar livremente sem que houvesse quebra dessa sequência por carregamento em paralelo ou por Reset.

Neste exercício temos uma situação diferente: a sequência pretendida começa no estado 6 e prossegue até ao estado final do contador integrado ($HHHH = 15_{(10)}$), e só então se quebra a sequência de contagem do contador. Ou seja, queremos agora uma sequência que se obtém eliminando as primeiras palavras do CBN.

Naturalmente, nestes casos *somos forçados* a fazer o carregamento em paralelo, e está fora de questão utilizar o Reset do contador integrado.

Como o número de estados pretendido é igual a 10 temos uma **década de contagem** — ou, mais simplesmente, uma **década** — isto é, um divisor de frequência por 10. A designação cobre todos os divisores de frequência com módulo 10, independentemente da sequência de contagem utilizada (se uma década utilizar o código BCD, então dizemos que estamos em presença de uma **década BCD**).

Década (de contagem)

Divisor de frequência por 10

Década BCD

O diagrama de blocos da Figura 14.14 ilustra o esquema de princípio para o contador que se pretende implementar. Neste caso, a entrada de Reset vem permanentemente desactivada. O detector do estado 15 gera um sinal *LOAD_L* de carregamento em paralelo síncrono, o que faz com que o contador carregue o estado 6 assim que for detectado o estado 15.

Em termos de diagrama temporal, a situação pretendida é a que se indica na Figura 14.15. No instante t_1 , com *LOAD_L* activo e um flanco ascendente em *CLK_H*, faz-se o carregamento em paralelo do contador.

Para implementar o detector do estado 15 vamos atender ao símbolo IEC do contador 74LS163A. O 74LS163A possui uma saída *3CT=15* cujo significado, como sabemos, é o seguinte: a saída vem activada (a H) se a entrada G3 estiver

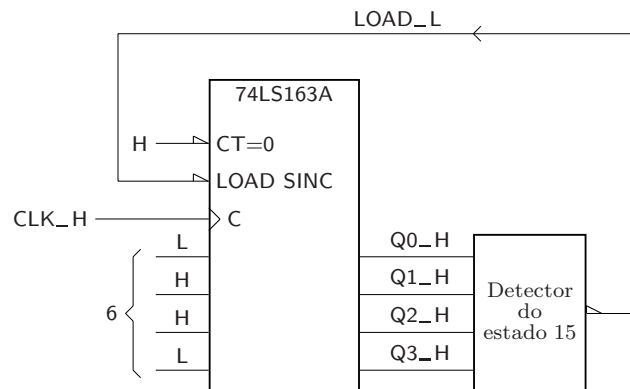
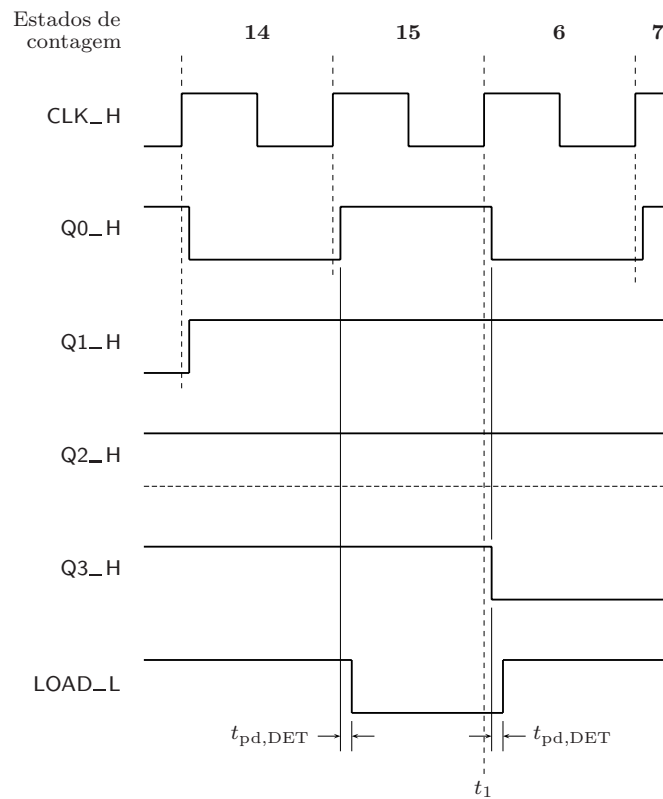


Figura 14.14: Diagrama de blocos da década do Exercício 14.27



Nota: o carregamento em paralelo é efectuado em t_1 , isto é, no único instante em que a função $LOAD$ está activa e ocorre num flanco ascendente em CLK

Figura 14.15: Diagrama temporal para a década do Exercício 14.27

activada (a H) e se o contador tiver atingido o estado 15. Então, a saída 3CT=15 vem activada durante *todo* o estado **15**, tal como pretendíamos para a saída $LOAD_L$ do detector.

Por outras palavras, o 74LS163A já possui internamente um detector do estado 15. Contudo, $LOAD_L$ é activa a L, enquanto que a saída $3CT=15$ é activa a H. Podemos, então, converter facilmente uma na outra à custa de um conversor de polaridade, como se indica no logograma final da Figura 14.16.

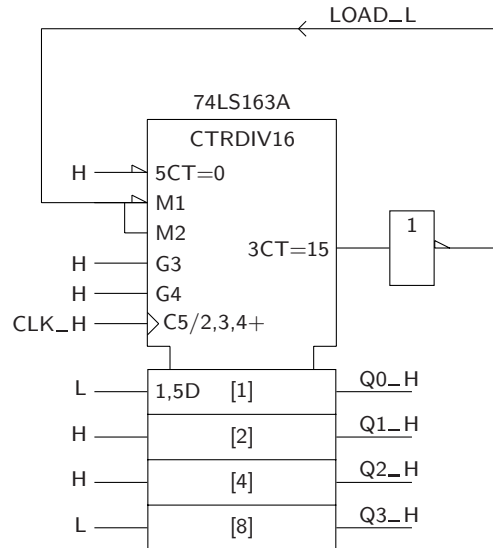


Figura 14.16: Logograma da década do Exercício 14.27

Antes de terminar, uma última observação: quando $LOAD_L$ está inactivo, o que acontece durante os estados 6 a 14, o contador conta (a função M2 vem activada o que, conjuntamente com G3 e G4 activos, permite a contagem ascendente do 74LS163A). Só durante o estado 15, em que $LOAD_L$ vem activo (mais precisamente quando ocorre um flanco ascendente em CLK_H) é que o contador integrado deixa de contar para passar a carregar em paralelo o estado 6.

14.28 Utilizar um contador integrado 74LS163A para implementar a sequência de contagem

$$\dots, 1, 2, 3, 6, 7, 8, 12, 13, 1, 2, \dots$$

Recorrer ao menor número possível de integrados para desenhar o esquema eléctrico do circuito que obtiver.

Resolução: Até agora temos implementado contadores com sequências de contagem que apenas possuem uma quebra de contagem, como sucede na transição do estado 15 para o estado 6 na sequência

$$\dots, 6, 7, 8, 9, \dots, 14, 15, 6, 7, \dots$$

No presente exercício, contudo, a sequência de contagem no CBN sofre *três* quebras (nas transições de 3 para 6, de 8 para 12 e de 13 para 1). Por isso, teremos de elaborar mais profundamente a solução apresentada no Exercício 14.20, que pertencia à classe mais simples de problemas em que não era necessária a lógica de carregamento em paralelo, ao contrário do que sucede aqui.

14.28

Tabela de transições

Para tanto, vamos utilizar agora uma **tabela de transições**, ou seja, uma tabela onde em cada linha se indicam as condições para uma determinada quebra da sequência de contagem e o estado para onde se pretende ir se essa condição se verificar. Cada linha da tabela contém, então:

1. um estado actual de contagem;
2. se for caso disso, as condições que possam existir em eventuais entradas externas do contador (para além do CLK_H);
3. o estado para o qual se pretende ir (estado seguinte) quando as condições anteriores se verificarem.

A cada linha de uma tabela de transições corresponde, então, o carregamento de um estado seguinte diferente do que resultaria do normal processo de contagem.

É de notar que as tabelas de transições podem ser utilizadas para qualquer sequência de contagem (podíamos, por exemplo, tê-las utilizado nos exercícios anteriores).

Tabela 14.3: Tabela de transições para o contador do Exercício 14.28, com identificação das quebras da sequência de contagem

	Estado actual (EA)				Estado seguinte (ES)				
	Q_3_H	Q_2_H	Q_1_H	Q_0_H	P_3_H	P_2_H	P_1_H	P_0_H	
3	L	L	H	H	L	H	H	L	6
8	H	L	L	L	H	H	L	L	12
13	H	H	L	H	L	L	L	H	1

Vamos começar por estabelecer uma tabela de transições com as condições em que se verifica uma quebra (truncagem) na sequência de contagem (Tabela 14.3). Por exemplo, quando o contador se encontrar no estado actual

$$(Q_3, Q_2, Q_1, Q_0) = (L, L, H, H),$$

queremos que ele passe para o estado seguinte

$$(Q_3, Q_2, Q_1, Q_0) = (L, H, H, L),$$

pelo que deveremos carregar em paralelo a quantidade booleana geral

$$(P_3, P_2, P_1, P_0) = (L, H, H, L).$$

Então, em termos de diagrama de blocos temos a situação descrita na Figura 14.5, ou, de forma mais específica para o nosso exercício, na Figura 14.17, com dois circuitos combinatórios já conhecidos do Exercício 14.20:

— o circuito CC1 (a *lógica de detecção de estados*), que gera a função *LOAD* que pretendemos activa a L, e que diz *quando* é que se deve fazer um carregamento em paralelo (obviamente, quando não carrega em paralelo, o 74LS163A conta); e

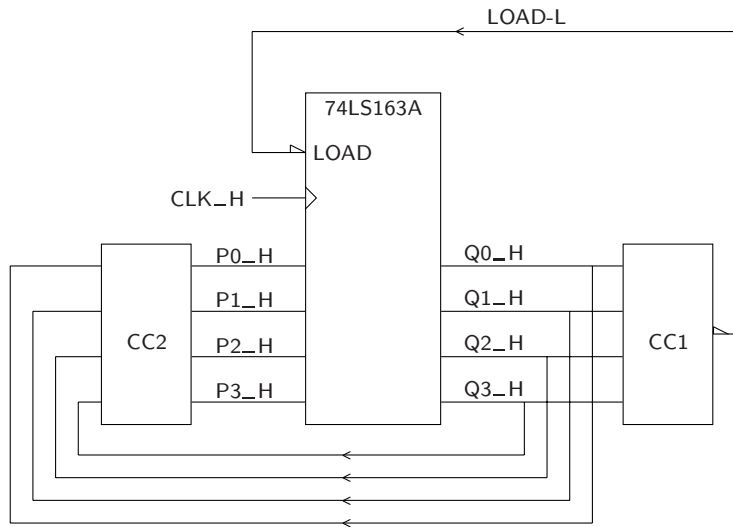


Figura 14.17: Diagrama de blocos do contador do Exercício 14.28

— o circuito CC2 (a *lógica de carregamento em paralelo*), que diz o que é que se deve carregar em paralelo.

De um modo geral, a lógica de detecção de estados pode ser implementada seguindo o seguinte raciocínio: queremos que a função *LOAD* venha activada quando detectarmos determinados conjuntos de estados, por exemplo um estado α , ou um estado β , ou um estado γ , etc.

Temos, portanto, que a lógica de detecção de estados realiza uma função *LOAD* em soma de produtos:

$$LOAD = \text{estado } \alpha + \text{estado } \beta + \text{estado } \gamma + \dots,$$

com o nível de actividade do OR final adequado ao nível de actividade do *LOAD*. Por exemplo, a saída do OR final no caso da Figura 14.17 deve vir activa a L, para que o carregamento em paralelo se possa efectivar.

Vamos agora ver como detectar os estados α , β , γ , etc. A forma correcta de o fazer recorre a um quadro de Karnaugh de n variáveis, em que n é o número de flip-flops do contador (no caso da Figura 14.17 temos $n = 4$, e as variáveis do quadro são $Q0$ a $Q3$).

Cada posição do quadro corresponde a um estado do contador, desde o estado 0 até ao estado $n - 1$. Basta colocar “1”s nos quadrados correspondentes aos estados que queremos detectar (os estados que provocam a quebra da sequência de contagem, e que correspondem às situações de carregamento em paralelo).

Por outro lado, devemos colocar “0”s nos estados em que queremos que o contador conte, em vez de carregar em paralelo.

E, finalmente, colocamos indiferenças nos estados pelos quais a sequência de contagem *não passa*.

Quanto à lógica de carregamento em paralelo, basta-nos estabelecer um quadro de Karnaugh para cada uma das funções P_i que queremos aplicar às entradas

de carregamento em paralelo, funções essas que dependem, naturalmente, dos estados actuais em que o contador se encontra (não esquecer que, no quadro de Karnaugh, cada quadrado corresponde a um estado actual).

Nesses quadros de Karnaugh, então, só vamos preencher os quadrados para os estados actuais em que queremos efectuar carregamentos em paralelo. Nos outros estados colocamos indiferenças. E, nos quadrados que vamos preencher, colocamos “1”s e “0”s de acordo com *estados seguintes* que tivermos obtido na tabela de transições de cada P_i (no nosso exemplo, a Tabela 14.3).

Vamos agora concretizar com o nosso exercício.

A lógica de CC1 é fácil de determinar: pretendem-se fazer carregamentos em paralelo quando se detectarem os estados actuais da Tabela 14.3, isto é, os estados 3, 8 e 13.

No quadro de Karnaugh da Figura 14.18 representam-se os valores para a função *LOAD*: valores 1 para os estados actuais 3, 8 e 13, em que queremos fazer carregamentos em paralelo; valores 0 para os estados actuais 1, 2, 6, 7 e 12, em que queremos deixar o contador contar; e indiferenças para os restantes estados actuais, pelos quais o contador, em princípio, não irá passar (sobre esta questão dos estados pelos quais o contador não irá passar ver, contudo, o Exercício 14.29).

$Q_3 \backslash Q_2$		$Q_1 Q_0$			
		00	01	11	10
00	—	0	1	0	
01	—	—	0	0	
11	0	1	—	—	
10	1	—	—	—	

$$LOAD = \overline{Q_2} Q_1 Q_0 + Q_3 \overline{Q_2} + Q_3 Q_0$$

Figura 14.18: Quadro de Karnaugh para a função *LOAD*

Ou seja, obtemos a soma de produtos mínima para *LOAD*:

$$LOAD = \overline{Q_2} Q_1 Q_0 + Q_3 \overline{Q_2} + Q_3 Q_0.$$

A lógica em CC2 é igualmente fácil de determinar se atendermos a que P_3 , P_2 , P_1 e P_0 são funções booleanas simples de Q_3 , Q_2 , Q_1 e Q_0 . Podemos, então, estabelecer quadros de Karnaugh para cada um dos P_i , como se indica na Figura 14.19.

Alternativamente, e de forma bastante mais simples, podemos analisar a Tabela 14.3 e, a partir dela, deduzir *directamente* as expressões lógicas dos P_i . Com efeito, a coluna P_0 da tabela vem igual à coluna Q_2 , pelo que podemos deduzir que $P_0 = Q_2$. Por outro lado, a coluna P_1 é o complemento da coluna Q_3 e é também igual à coluna Q_1 , ou seja, temos que $P_1 = \overline{Q_3} = Q_1$. De forma semelhante, a coluna P_2 é o complemento da coluna Q_2 , pelo que $P_2 = \overline{Q_2}$.

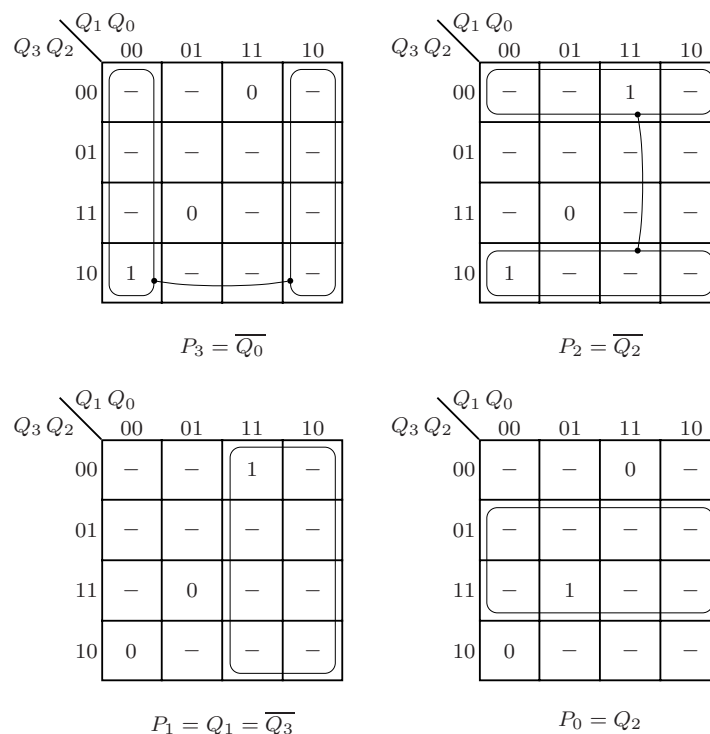


Figura 14.19: Quadros de Karnaugh para a função booleana geral $\mathbf{P} = (P_3, P_2, P_1, P_0)$

Finalmente, a coluna P_3 é o complemento da coluna Q_0 , ou seja, temos que $P_3 = \overline{Q_0}$.

Naturalmente, esta dedução simplificada das expressões mínimas para os P_i é possível porque os quadros de Karnaugh respectivos possuem 13 quadrados com indiferenças nas linhas que não são utilizadas na tabela de transições do contador.

Finalmente, estamos agora em posição de desenhar na Figura 14.20 o esquema eléctrico do contador. Em particular, é de notar como se utilizam apenas dois circuitos integrados para além do contador 74LS163A: um circuito integrado 74LS10 composto por três portas AND, cada uma com três entradas activas a H e a saída activa a L, e um circuito integrado 74LS00 formado por quatro portas AND, cada uma com duas entradas activas a H e a saída activa a L.

74LS10

74LS00

14.29 Desenhar o diagrama de estados completo do contador do Exercício 14.25, que utiliza o contador integrado 74LS163A. O que acontece ao contador se ele for inicializado no estado $12_{(10)}$ quando o circuito vem alimentado electricamente?

Resolução: No Exercício 14.25 admitiu-se que o contador nunca sai do seu **ciclo de contagem**

$$\dots, 0, 1, 2, 3, 4, 5, 0, 1, \dots,$$

o que pressupõe que o seu **estado inicial** é um dos que pertencem a este ciclo; ou seja, que o contador vem para um destes estados imediatamente após ter sido

14.29

Ciclo de contagem

Estado inicial

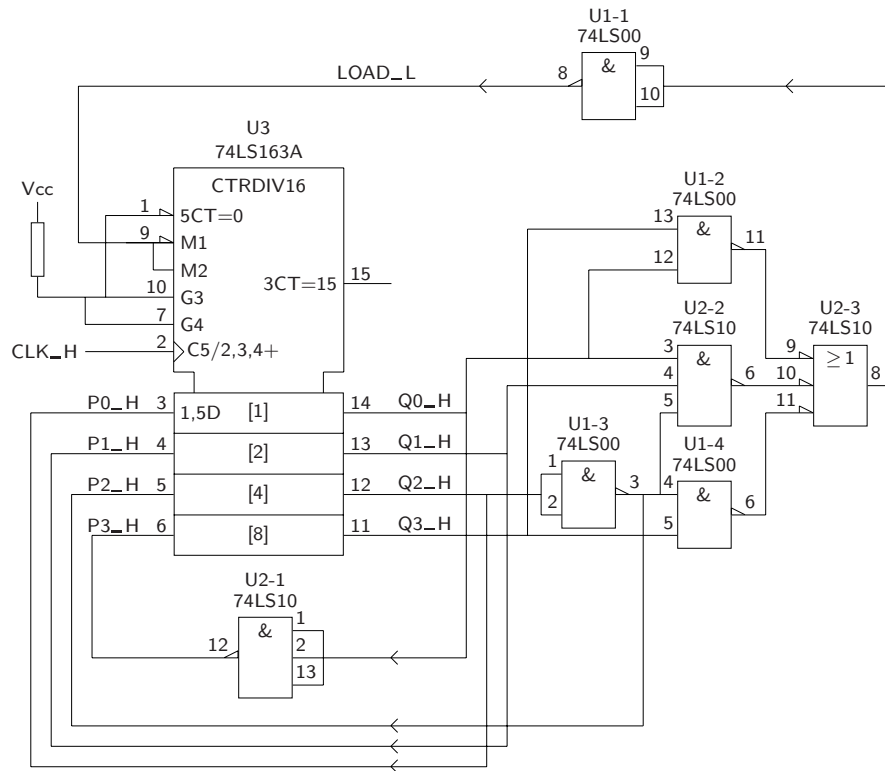


Figura 14.20: Esquema eléctrico do contador do Exercício 14.28

alimentado electricamente, isto é, depois da aplicação da tensão de alimentação V_{cc} ao circuito.

Contudo, como sabemos, não é possível prever o estado inicial de um latch ou de um flip-flop, qualquer que seja a sua estrutura. Logo, *é impossível prever o estado inicial do contador*. Pode perfeitamente acontecer que o estado inicial esteja fora do ciclo de contagem, por exemplo que seja o estado 12₍₁₀₎ do enunciado.

Sequência de (estados
de) contagem

Diagrama de estados
(de um contador)

Para resolvermos este problema temos de estabelecer a **sequência de estados de contagem** ou, mais simplesmente, a **sequência de contagem**, que inclui o ciclo de contagem do contador. Por outras palavras, queremos desenhar o **diagrama de estados** do contador.

Na Figura 14.21(b) apresenta-se o diagrama de estados do contador da Figura 14.9. Esse diagrama depende, como iremos ver a seguir, do modo como é detectado o estado que faz o *CLEAR* do contador integrado.

Na Figura 14.9 (página 134) a detecção do estado 5 foi feita de forma simplificada, aproveitando apenas dois níveis H do estado actual do 74LS163A, mais exactamente $Q2_H = H$ e $Q0_H = H$ [Figura 14.21(a), que reproduz o logigrama do contador].



A consequência de termos utilizado esta detecção simplificada é que, *para além de detectarmos o estado 5, também detectamos os estados 7, 13 e 15, já que todos eles possuem $Q2_H = H$ e $Q0_H = H$* , como podemos ver na Tabela 14.4.

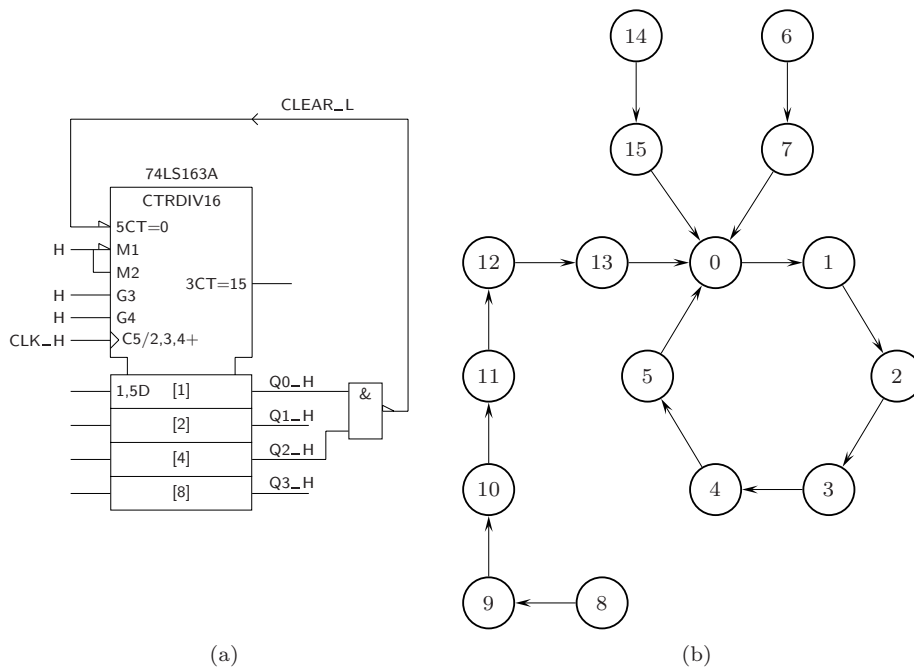


Figura 14.21: (a) Logigrama do contador da Figura 14.9; (b) sequência de contagem do contador, que inclui o ciclo de contagem $\dots, 0, 1, 2, 3, 4, 5, 0, 1, \dots$

Tabela 14.4: Tabela com a identificação dos estados descodificados no logigrama das Figuras 14.9 e 14.21(a)

Estado	Q3_H	Q2_H	Q1_H	Q0_H
5	L	H	L	H
7	L	H	H	H
13	H	H	L	H
15	H	H	H	H

Quer isso dizer que, se o contador atingir um destes estados (ou porque vem de um estado de contagem anterior, ou porque foi inicializado num desses estados, ou ainda porque existiu uma falha no circuito e ele foi, aleatoriamente, colocado num desses estados), é feito o *CLEAR* do 74LS163A e o contador recomeça do estado 0 — isto é, entra no ciclo de contagem, do qual não sairá a menos que ocorra uma (nova) falha.

Na Figura 14.21(b) mostra-se a sequência de contagem, incluindo o ciclo de contagem $\dots, 0, 1, 2, 3, 4, 5, 0, 1, \dots$. De notar as transições dos estados 7, 13, 15 e 5 para o estado 0. Por outro lado, notar como os restantes estados são de contagem, já que o detector está com a saída inactiva nesses casos.

Em resumo, tendo optado pela descodificação simplificada das Figuras 14.9 e 14.21(a), se o contador for inicializado, por exemplo, no estado 12, ele prosseguirá no estado 13 e a partir daí entra no ciclo de contagem pelo estado 0, mantendo-se nesse ciclo se não houver uma falha que o leve para um estado fora do ciclo.

Naturalmente, podíamos ter optado por construir um outro detector do estado 5, por exemplo como se ilustra na Figura 14.22(a). O AND agora utilizado tem 3 entradas e usa o facto de querermos $Q1_H = L$ na detecção desse estado.

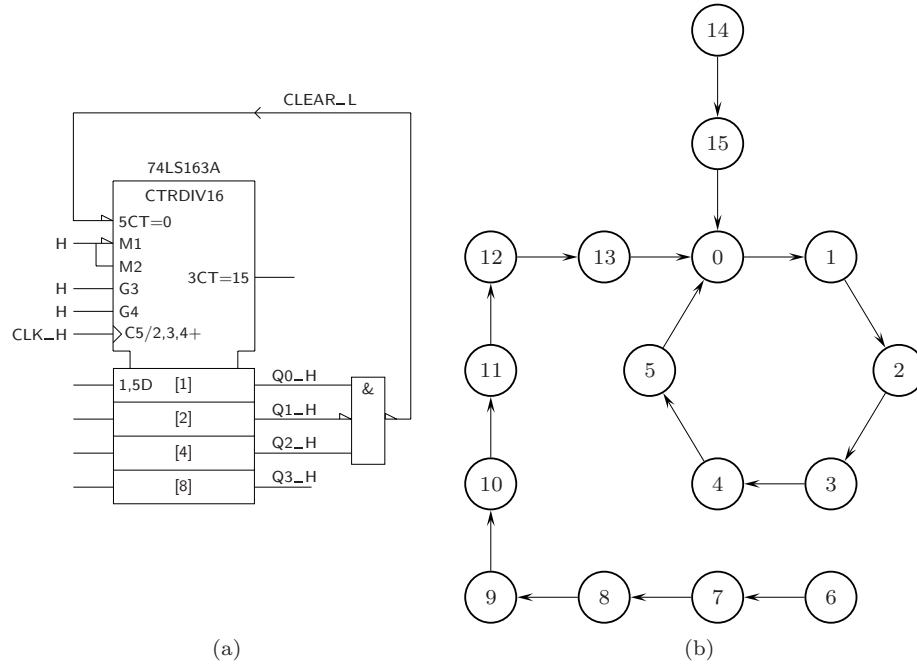


Figura 14.22: (a) Novo logigrama do contador da Figura 14.9, que altera o decodificador do estado 5; (b) correspondente sequência de contagem, que inclui o ciclo de contagem $\dots, 0, 1, 2, 3, 4, 5, 0, 1, \dots$

Agora obtemos uma sequência de contagem diferente da anterior, como se mostra na Figura 14.22(b), sequência essa que inclui o mesmo ciclo de contagem, como seria de esperar. A sequência diferente resulta do facto de o circuito de detecção detectar, para além do estado 5, também o estado 13, provocando o Reset do contador integrado em ambos os casos. Por outro lado, não esquecer que o estado que se segue ao estado 15 é, naturalmente, o estado 0, porque nada é feito para impedir essa transição no processo de contagem do contador integrado.

Finalmente, consideremos o detector do estado 5 da Figura 14.23(a), em que apenas esse estado vem, de facto, detectado.

Agora, a única forma de o contador entrar no ciclo de contagem é através do estado 15, por contagem. Ou seja, deixamos que o contador conte normalmente de 6 até 15 e daí prossiga para o estado 0, entrando então no ciclo de contagem, do qual não sairá a menos que ocorra uma falha.

74x169

14.34 O contador da Figura 14.33 usa um contador integrado do tipo 74x169. Qual é a sequência de contagem do contador?

14.34

Resolução: A direcção de contagem vem controlada por $Q3$: contagem ascendente se $Q3 = 1$ e descendente se $Q3 = 0$. Há carregamente em paralelo quando

o contador atinge um estado terminal: 1111 quando conta ascendentemente, e 0000 quando conta descendentemente. O bit mais significativo na saída do contador vem complementado e aplicado à entrada de carregamento em paralelo com maior peso. Os outros bits de saída vêm aplicados, sem alteração, às correspondentes entradas de carregamento em paralelo.

Admitamos que o contador se encontra inicialmente num dos estados 0000 – 0111. Nestas condições, O contador conta descendentemente porque $Q3 = 0$. Quando atingir o estado de contagem 0000, fica activo o carregamento em paralelo, o contador carrega 1000, e em seguida passa a contar ascendentemente (porque $Q3 = 1$). Quando atinge o estado 1111 o contador carrega em paralelo 0111 e, a partir daí, passa a contar descendentemente, repetindo o ciclo de contagem.

Se o contador estiver inicialmente num dos estados $1000 - 1111$, verifica-se o mesmo comportamento.

Segue-se que o ciclo de contagem é, em decimal,

$$\cdots, 8, 9, 10, 11, 12, 13, 14, 15, 7, 6, 5, 4, 3, 2, 1, 0, 8, \cdots.$$

Capítulo 15

Registos

15.1 Recorrendo a 4 flip-flops JK, implemente um registo com 4 andares que permita fazer deslocamento à direita, deslocamento à esquerda, deslocamento circular à direita, e memorizar em paralelo.

Resolução: Como queremos implementar um registo, o tipo de flip-flop mais adequado é o D. Como nos são dados flip-flops JK, podemos facilmente transformá-los em flip-flops D, como se mostra na Figura 15.1.

15.1

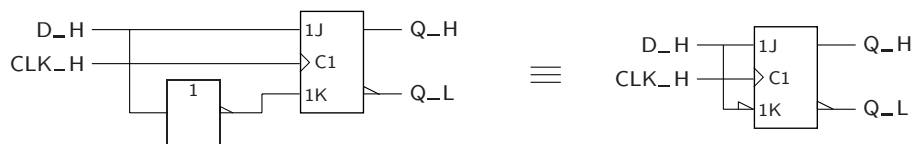


Figura 15.1: Um flip-flop JK transformado em flip-flop D

No nosso caso queremos implementar um registo multifunções. Como sugerido nas aulas teóricas, vai-se utilizar um multiplexer por cada flip-flop, com as suas entradas de dados ligadas adequadamente às saídas dos flip-flops e às entradas externas. Como se exigem quatro modos de funcionamento (ou funções) distintas para o registo, basta que os multiplexers possuam 4 entradas de dados e, portanto, 2 entradas de controlo, como se mostra na Figura 15.2(a).

Quanto aos 4 modos de funcionamento do registo, podemos afectá-los arbitrariamente às combinações de variáveis de selecção dos multiplexers. Por exemplo, podemos associar a variável booleana geral (*MODE1*, *MODE0*) aos modos do registo da forma expressa na tabela de verdade física da Figura 15.2(b).

Finalmente, uma observação quanto ao significado de **deslocamento circular** à direita (ou **rotação** à direita).

*Deslocamento circular
(rotação)*

A Figura 15.3(a) ilustra um deslocamento para a direita, enquanto que a Figura 15.3(c) ilustra um deslocamento para a esquerda. Por seu turno, as Figuras 15.3(b) e (d) ilustram, respectivamente, um deslocamento circular (rotação) para a direita e para a esquerda. Como podemos constatar, nas duas rotações o bit que sai

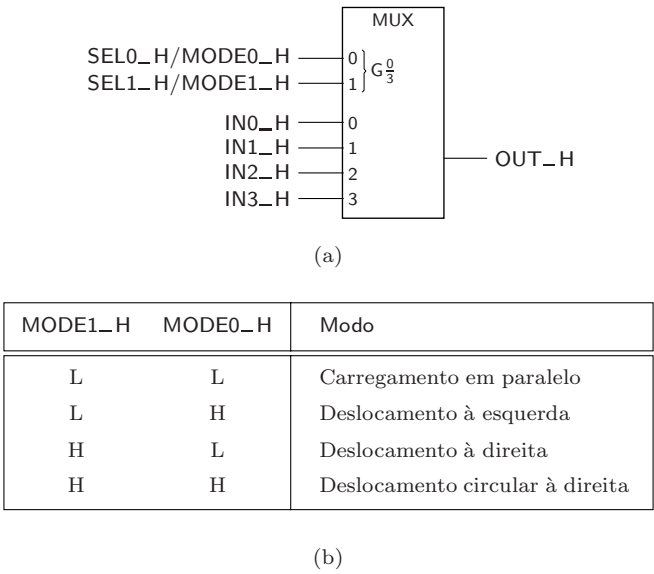


Figura 15.2: (a) Os multiplexers deste registo possuem 4 entradas de dados e 2 de controlo; (b) afectação (arbitrária) dos modos de funcionamento do registo às variáveis de selecção dos multiplexers

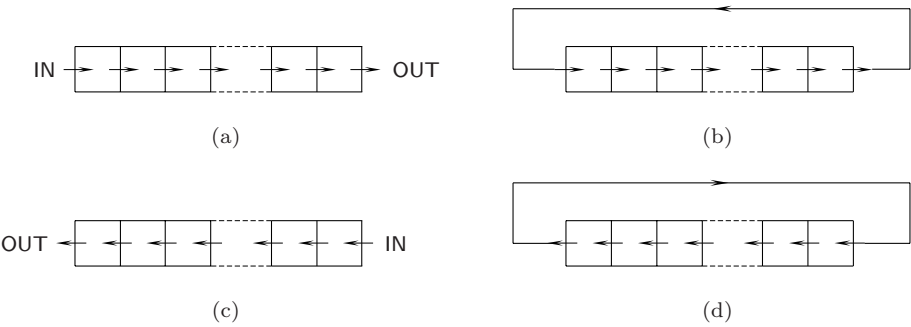


Figura 15.3: Esquematização: (a) de um deslocamento para a direita; e (b) de um deslocamento circular (ou rotação) para a direita. (c) e (d) Deslocamento e rotação para a esquerda

por uma das extremidades no deslocamento é reinserido pela outra extremidade (daí as designações “deslocamento circular” ou “rotação”).

Com estes elementos, podemos agora estabelecer as ligações adequadas, obtendo-se o logograma da Figura 15.4.

Comparando esta figura com a Figura 15.3, notemos que o flip-flop com saída $Q0_H$ é o menos significativo (mais à direita na Figura 15.3), e o flip-flop com saída $Q3_H$ é o mais significativo (mais à esquerda na Figura 15.3).

As entradas de carregamento em paralelo são designadas por $PAR.IN$. Quanto às entradas série, vêm designadas por $SER.IN.LEFT$ e $SER.IN.RIGHT$, respectivamente para o deslocamento para a esquerda e para o deslocamento para a direita. E as saídas correspondentes vêm designadas por $SER.OUT.LEFT$ e

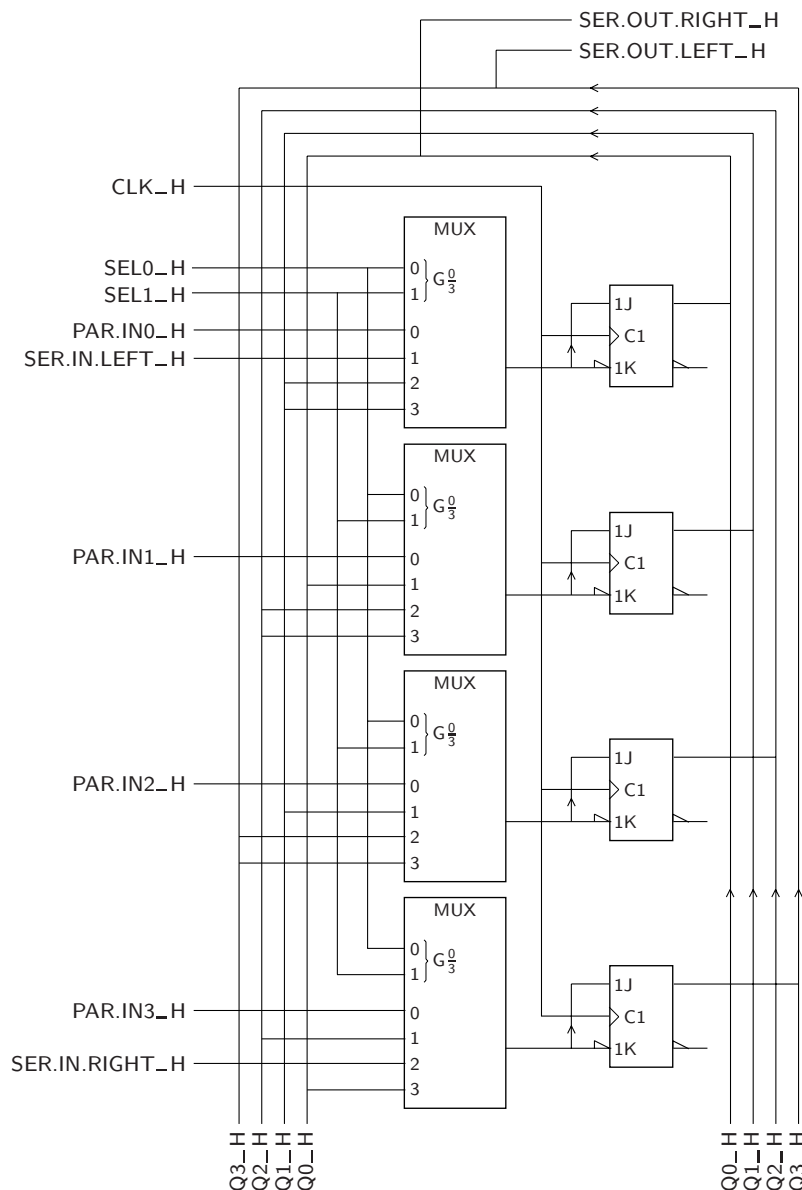


Figura 15.4: Logigramma do registo multifunções do Exercício 15.1

SER.OUT.RIGHT, como se esquematiza na Figura 15.5.

15.2 a) Construa um registo com 4 flip-flops do tipo D capaz de memorizar em paralelo do exterior, de efectuar a divisão do seu conteúdo por dois, e de duplicar o seu conteúdo (desde que o resultado da duplicação continue a poder ser representado com 4 bits).

b) Amplie o circuito anterior, de forma a ligar quatro registos idênticos aos pedidos na alínea anterior a um barramento comum.

Resolução: a) A ideia por detrás deste registo multifunções consiste em poder

15.2 a)

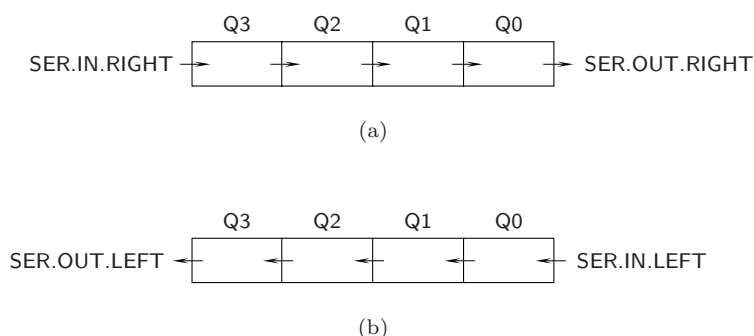


Figura 15.5: (a) Deslocamento para a direita; e (b) deslocamento para a esquerda

Operando guardar nele um determinado **operando**, um número sem sinal com 4 bits — e para isso precisamos de carregar em paralelo o operando no registo — e em seguida multiplicá-lo ou dividi-lo por 2, em alternativa, e deixar o resultado dessa operação guardada no registo para eventuais operações posteriores do mesmo tipo.

A estrutura do registo pretendido é idêntica à da do Exercício 15.1, pelo que recorreremos, mais uma vez, a um multiplexer por flip-flop para implementar os diversos modos de funcionamento. De mais simples temos o facto de, agora, se impor a utilização de flip-flops D, em vez dos flip-flops JK do exercício anterior.

Divisão inteira É necessário ter em conta que a divisão por 2 de um número pode facilmente ser obtida pelo deslocamento desse número no sentido dos pesos menores. E o número deve ser entendido como um número sem sinal e apenas com parte inteira (por essa razão, a divisão em questão também por vezes é designada por **divisão inteira**).



No nosso caso, se $Q3_H$ é a saída de maior peso e $Q0_H$ a de menor peso, esse deslocamento é no sentido de $Q3_H$ para $Q0_H$. *O quociente da divisão virá correcto se forem injectados “0”s pela entrada e se não saírem “1”s para o exterior.* Por exemplo, se

$$(Q3, Q2, Q1, Q0) = (1, 0, 1, 0),$$

a que corresponde o número $10_{(10)}$, um deslocamento para a direita dá origem a

$$(Q3, Q2, Q1, Q0) = (0, 1, 0, 1),$$

a que corresponde o número $5_{(10)} = 10_{(10)}/2$. Mas um novo deslocamento para a direita já produz

$$(Q3, Q2, Q1, Q0) = (0, 0, 1, 0),$$



a que corresponde o número $2_{(10)} \neq 5_{(10)}/2$. Então, e em rigor, o que se vai obtendo é a *parte inteira da divisão por 2, até se obter o resultado nulo*.

Por outro lado, a multiplicação por 2 de um número sem sinal e apenas com parte inteira pode facilmente ser obtida fazendo um deslocamento no sentido dos pesos maiores. No nosso caso, esse deslocamento deve ser no sentido de $Q0_H$ para $Q3_H$. De forma semelhante ao que acontecia com a divisão por 2,



também o produto virá correcto se forem injectados “0”s pela entrada e se não saírem “1”s para o exterior. Por exemplo, se

$$(Q3, Q2, Q1, Q0) = (0, 1, 0, 1),$$

a que corresponde o número $5_{(10)}$, um deslocamento para a esquerda dá origem a

$$(Q3, Q2, Q1, Q0) = (1, 0, 1, 0),$$

a que corresponde o número $10_{(10)} = 5_{(10)} \times 2$. Mas um novo deslocamento para a esquerda já produz

$$(Q3, Q2, Q1, Q0) = (0, 1, 0, 0),$$

a que corresponde o número $4_{(10)} \neq 10_{(10)} \times 2$. Então, e em rigor, o que se vai obtendo é o *módulo 16 do produto por 2, até se obter o resultado nulo* (módulo 16 no nosso caso, porque temos 4 andares no registo; com n andares seria módulo 2^n).



Vamos agora associar arbitrariamente a $(MODE1, MODE0)$ os 3 modos do registo, por exemplo na forma expressa na tabela de verdade física da Tabela 15.1.

Tabela 15.1: Afectação (arbitrária) dos modos de funcionamento do registo às variáveis de selecção dos multiplexers

MODE1_H	MODE0_H	Modo
L	L	Carregamento em paralelo
L	H	Multiplicação por 2
H	L	Divisão por 2
H	H	Não utilizado

Com estas considerações, e tendo em conta o que já foi desenvolvido no exercício anterior, obtemos o logograma da Figura 15.6.

b) Para resolver este exercício necessitamos de duas coisas:

15.2 b)

1. arranjar um símbolo IEC para o registo multifunções desenvolvido na alínea a); e
2. arranjar maneira de multiplexar as saídas dos 4 registos para um barramento comum formado por 4 linhas, num circuito vulgarmente designado por **banco de registos**.

Banco de registos

Quanto à questão do símbolo IEC para o registo multifunções, constatamos que ele deve ser muito parecido com o da Figura 15.12 de *SD:AAT*, que representa um registo de deslocamento universal com 4 andares do tipo 74x194 e a capacidade de carregar em paralelo, deslocar para a direita e deslocar para a esquerda — exactamente o que necessitamos aqui. A diferença é que o registo multifunções da alínea a) exige que nas entradas série sejam injectados níveis L. Mas esta questão tem a ver com o projecto do circuito que usa o registo multifunções, e não com o registo propriamente dito ou com o seu símbolo IEC — este aceita

74x194

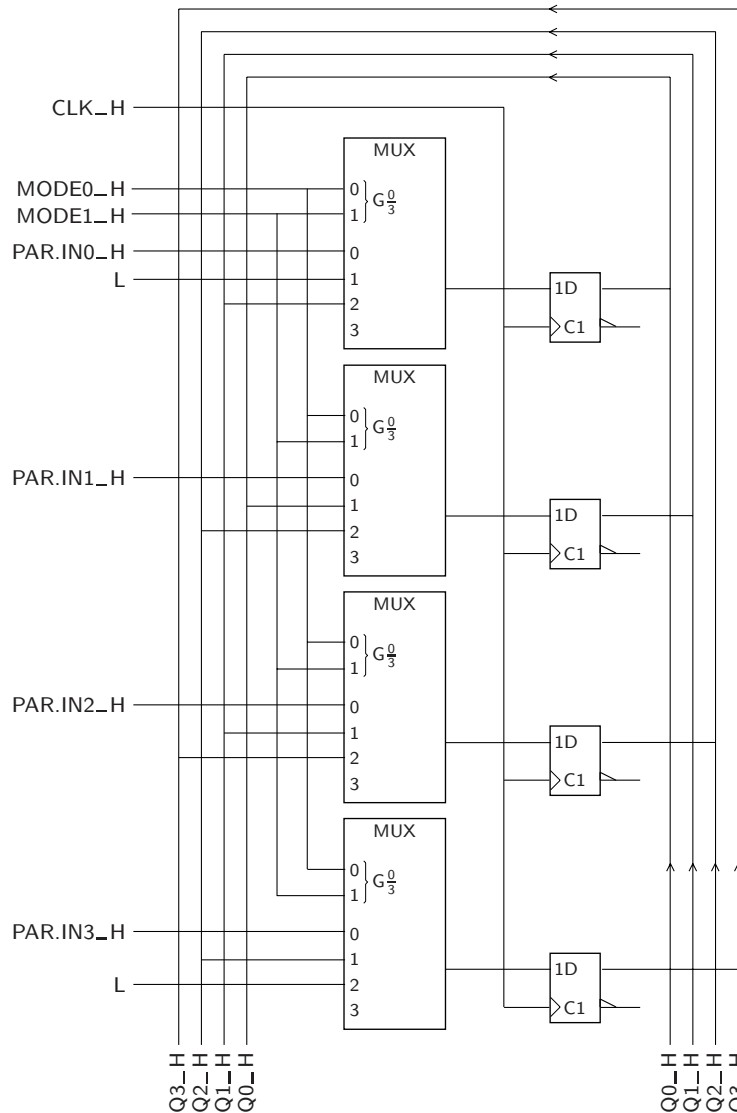


Figura 15.6: Logigramma do registo multifunções do Exercício 15.2

qualquer nível que lhe seja injectado pelas entradas de deslocamento série, e se esses níveis forem H não obtemos a multiplicação por 2 nem a divisão por 2 do número nele previamente guardado.

Segue-se que podemos de imediato desenhar o símbolo do registo anterior na Figura 15.7.

Quanto à questão da multiplexagem das saídas dos 4 registos multifunções para um barramento comum formado por 4 linhas, podemos fazê-lo também de duas maneiras:

- usar 4 multiplexers para encaminhar as saídas dos 4 registos multifunções

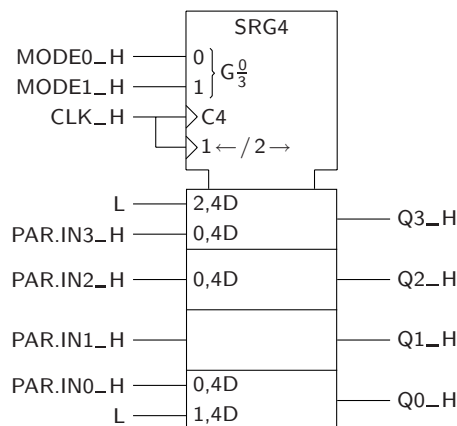


Figura 15.7: Símbolo IEC do registo multifunções da Figura 15.6

para o barramento comum; ou

- usar 4 Buffers tri-state para fazer exactamente a mesma operação.

Em qualquer dos casos precisamos de 2 variáveis que identifiquem o *número* do registo a seleccionar, variáveis essas que não devem ser confundidas com as variáveis de selecção dos multiplexers que utilizámos para construir os registos multifunções e que serviam para identificar a *função* (ou *modo*) por eles desempenhada — por isso, daremos a estas novas variáveis designações diferentes.

As duas soluções possuem vantagens e inconvenientes.

O uso de Buffers tri-state gera um circuito mais simples nas ligações entre as saídas dos registos e o barramento. Basta pensarmos que, se usarmos multiplexers, precisamos de encaminhar as 16 saídas dos registos para as 16 linhas de entradas de dados dos multiplexers, o que dá uma bela colecção de fios entrecruzados; se usarmos Buffers, ligamos as 4 saídas de cada registo às 4 entradas de cada Buffer, numa ligação muito mais “limpa”.

Por outro lado, a utilização de Buffers tri-state obriga à utilização de um decodificador para controlar as 4 entradas de Enable dos Buffers, Enables esses que, como facilmente percebemos, têm de ser gerados à custa das 2 variáveis de identificação do registo a seleccionar para o barramento.

Das duas alternativas, escolhemos a utilização dos Buffers porque permite gerar um logigrama mais legível (também um circuito impresso ou integrado com esta solução possui menos problemas de encaminhamento dos sinais).

A Figura 15.8 ilustra esta solução apenas para o Registo 3 e para o correspondente Buffer tri-state do banco de registos.

15.7 Um registo de deslocamento de comprimento variável é um registo SISO com um número variável de bits. Este tipo de registo é utilizado para provocar um número variável de ciclos (de relógio) de atraso da entrada para a saída. O comprimento vem especificado por uma variável booleana geral de controlo.

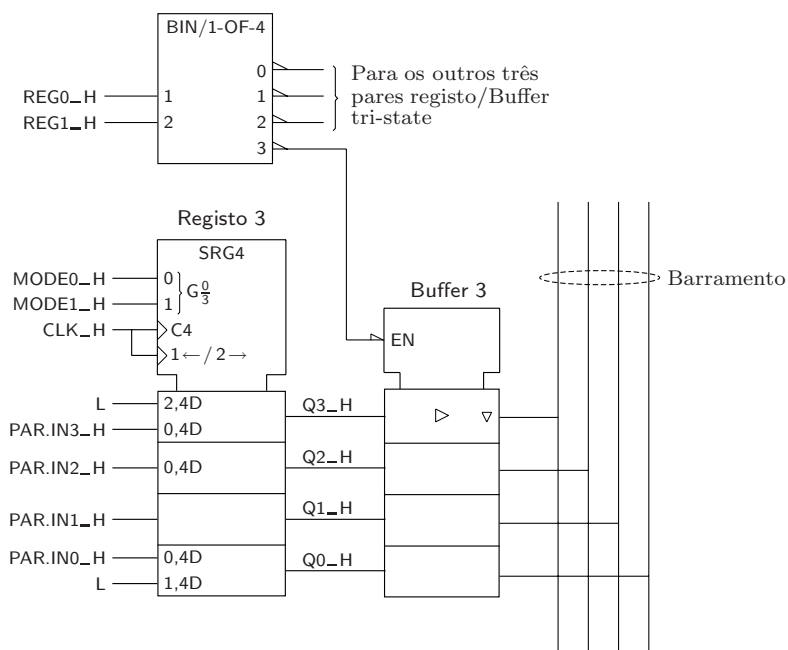


Figura 15.8: Logigrama de uma quarta parte do banco de registos multifunções do Exercício 15.2 (apenas para o Registo 3 e o correspondente Buffer tri-state)

Desenhe o logigrama de um registo de deslocamento com comprimento variável entre 1 e 8 bits. Use, para tanto, um registo SIPO e módulos combinatórios.

15.7

Resolução: A implementação do registo de comprimento variável encontra-se na Figura 15.9.

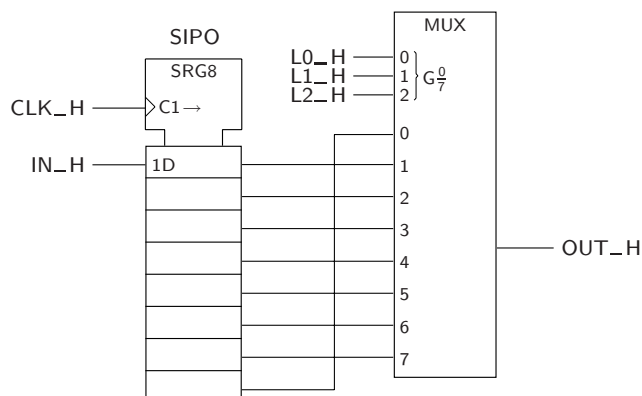


Figura 15.9: Logigrama de um registo de deslocamento com comprimento variável entre 1 e 8

Como o comprimento do registo varia entre 1 e 8, usamos uma variável booleana geral de controlo, $\mathbf{L} = (L2, L1, L0)$, que comanda as entradas de selecção do multiplexer.

O valor $L = 0$ corresponde a um deslocamento de 8 posições. Os restantes valores $L = i$ correspondem a deslocamentos de i posições.

O multiplexer selecciona a saída adequada do registo SIPO, o que irá provocar o atraso pretendido. Por exemplo, se $L = 3$, a saída OUT_H é obtida da saída 2 do registo de deslocamento, impondo um atraso de 3 impulsos de relógio aos dados que forem sendo injectados pela entrada IN_H do registo.

15.9 Como projectaria um andar (genérico) de um registo de deslocamento com carregamento em paralelo assíncrono? E se fosse com carregamento em paralelo síncrono?

Resolução: As Figuras 15.10(a) e (b) apresentam duas soluções possíveis para o caso do carregamento em paralelo assíncrono, quando se usam flip-flops do tipo D.

15.9

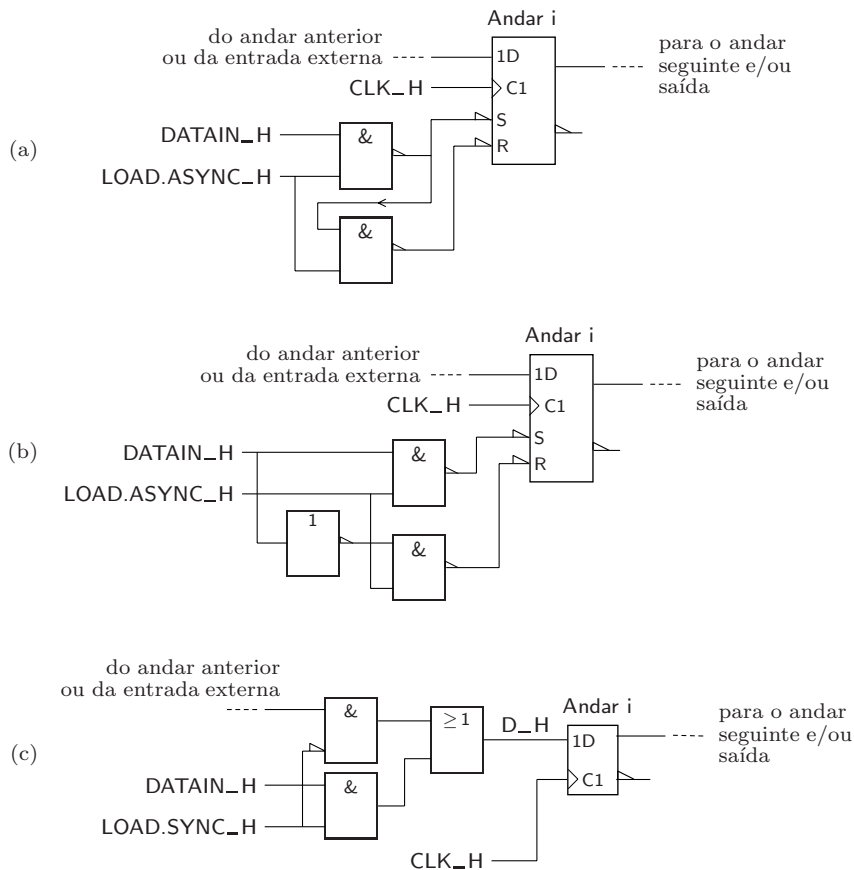


Figura 15.10: (a) e (b) Logigramas alternativos de um andar genérico de um registo de deslocamento com carregamento em paralelo assíncrono; e (c) com carregamento em paralelo síncrono

Naturalmente, neste caso temos de utilizar flip-flops com entradas assíncronas, porque só recorrendo a elas se pode fazer carregamento em paralelo assíncrono.

Quando não se quer fazer carregamento em paralelo — ou porque se quer manter

o estado do registo ou porque se quer fazer deslocamento, consoante o que for aplicado à entrada D_H do flip-flop— desactiva-se exteriormente a entrada de controlo $LOAD.ASYNC_H$, e as entradas de Set e de Reset dos flip-flops vêm ambas desactivadas.

Quando a entrada de controlo $LOAD.ASYNC_H$ vem activada, o nível de tensão presente na entrada de dados $DATAIN_H$ activa uma das entradas assíncronas de Set ou de Reset (e apenas uma delas) e o nível de tensão que estiver aplicado a $DATAIN_H$ aparece à saída do flip-flop após um certo tempo de propagação; diz-se que houve carregamento em paralelo assíncrono.

A Figura 15.10(c) apresenta um carregamento em paralelo síncrono onde, naturalmente, se usa a (única) entrada de dados síncrona do flip-flop, a entrada D_H .

Neste caso multiplexa-se essa entrada entre a saída do andar anterior (ou a entrada externa do registo, se se tratar do primeiro andar) e a entrada de dados $DATAIN_H$, sendo a multiplexagem controlada por $LOAD.SYNC_H$.

Se $LOAD.SYNC_H$ estiver activa, o nível de tensão na entrada de dados $DATAIN_H$ aparece à entrada D_H do flip-flop e, após ocorrência do próximo flanco de comutação, também à saída do flip-flop (decorrido que é o seu tempo de propagação). Houve, então, carregamento em paralelo síncrono.

Se $LOAD.SYNC_H$ estiver inactiva, a entrada $DATAIN_H$ não tem qualquer influência sobre a entrada síncrona do flip-flop, que recebe agora o que vier do andar anterior ou da entrada externa do registo.

Capítulo 16

Circuitos Sequenciais Síncronos

16.6 Para uma determinada máquina síncrona com entrada X_H e saída Z_H obteve-se o comportamento temporal descrito na Figura 16.50 (de *SD:AAT*). A máquina foi construída segundo o modelo de Moore ou de Mealy? Justifique.

Resolução: A Figura 16.1 analisa o comportamento temporal da máquina.

16.6

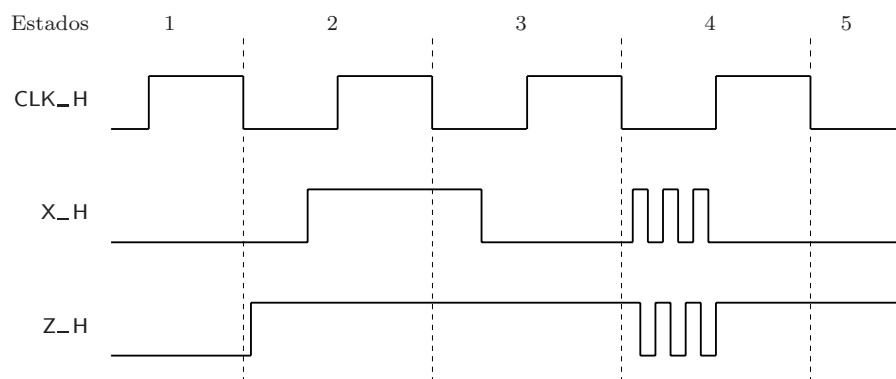


Figura 16.1: Análise do comportamento temporal da máquina síncrona do Exercício 16.6

Notemos que existe uma mudança na saída (na passagem do estado 1 para o estado 2) que ocorre imediatamente depois de um flanco descendente na entrada de relógio, sem que tenha ocorrido uma mudança na entrada. Isso quer dizer que essa mudança de estado dependeu exclusivamente do flanco de relógio. Podemos, assim, concluir que *a máquina sequencial foi construída com flip-flops que comutam nos flancos descendentes de CLK_H*.

Notemos ainda que, nos estados 1, 2, 3 e 5, a saída se mantém constante. Trata-se de uma **saída de Moore** porque, nesses estados, a saída do circuito não depende directamente da entrada.

Saída de Moore

Saída de Mealy

Pelo contrário, no estado 4 a saída muda com as variações da entrada (na realidade, é o seu complemento). Temos, neste caso, uma **saída de Mealy** porque, nesse estado, a saída depende directamente da entrada. Ora, para que tal aconteça, o circuito combinatório de saída deve depender do estado e da entrada (vd. a Figura 16.6 de *SD:AAT*), pelo que a máquina foi construída segundo o modelo de Mealy. Ou seja, é condição necessária e suficiente para uma máquina ser de Mealy que contenha pelo menos uma saída de Mealy.

16.9 Desenhe o diagrama de estados de uma máquina sequencial síncrona cuja função é gerar um bit de paridade para as palavras analisadas. As palavras têm comprimento 3, mas o circuito usa 4 impulsos de relógio para analisar cada palavra: os primeiros 3 impulsos são para os bits da palavra e o quarto impulso serve para a geração do bit de paridade. O bit de paridade deve vir igual a 1 se a paridade da palavra recebida for par, e a 0 se for ímpar. Enquanto o quarto impulso não ocorrer, o valor na saída é indiferente. Desenhe o circuito: (a) como uma máquina de Moore; e (b) como uma máquina de Mealy.

16.9 a)

Resolução: a) O diagrama da máquina de Moore encontra-se na Figura 16.2.

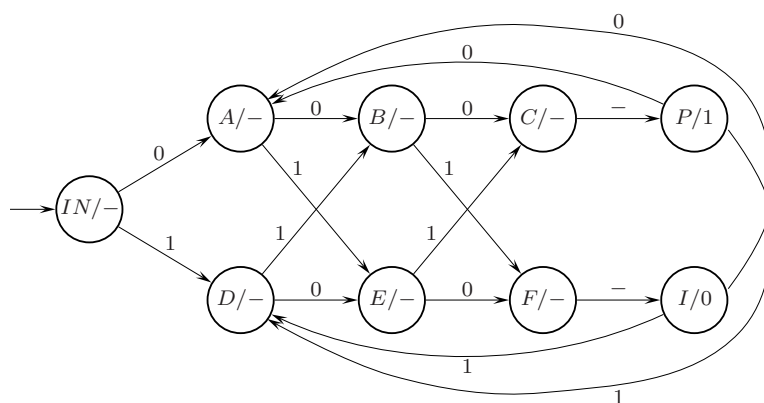


Figura 16.2: Diagrama de estados da máquina de Moore do Exercício 16.9 a)

Como podemos constatar, o estado P (com o significado de “par”) corresponde a uma paridade da palavra par, em que se pretende gerar saída a 1; e o estado I (com o significado de palavra “ímpar”) deve gerar saída a 0. Por outro lado, facilmente se pode verificar que o estado inicial, IN , é redundante, podendo o estado inicial ser qualquer um dos estados P ou I . Podemos, então, simplificar o diagrama de estados, como mostra a Figura 16.3, admitindo P como estado inicial.

16.9 b)

b) O diagrama da máquina de Mealy encontra-se na Figura 16.4.

Nesta máquina, o estado IN tem mesmo que ser o estado inicial.

16.10 Desenhe um diagrama de estados para uma máquina de Mealy com uma entrada série e uma saída que repete a sequência de entrada com dois períodos de relógio de desfasamento. Os dois primeiros valores na saída devem ser iguais a 0.

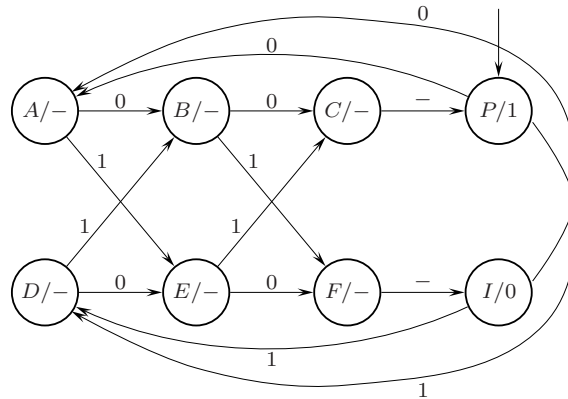


Figura 16.3: Diagrama de estados simplificado da máquina de Moore do Exercício 16.9 a)

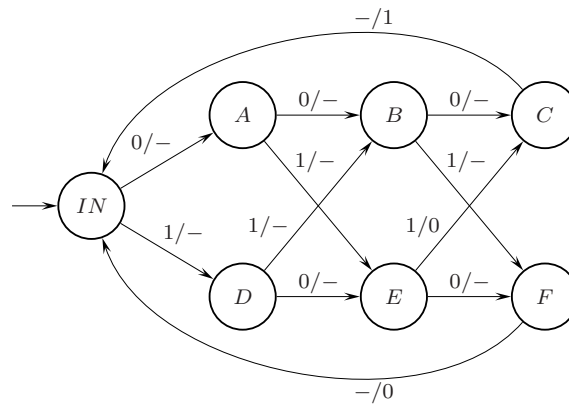


Figura 16.4: Diagrama de estados da máquina de Mealy do Exercício 16.9 a)

Resolução: Vamos gerar a saída da máquina coincidentemente com o bit de entrada que se segue ao par de bits anteriores.

16.10

Vamos começar por desenhar passo a passo o diagrama de estados, sem qualquer preocupação de obtenção de uma solução ótima, isto é, de uma solução com o menor número de estados.

Se a saída deve reproduzir os bits de entrada com 2 períodos de relógio de atraso, é fácil começar por desenhar a primeira parte do diagrama de estados: abrimos o diagrama a partir de um estado inicial, *A*, para todas as combinações de 2 bits na entrada nos 2 ciclos de relógio, como mostra a Figura 16.5.

Podemos, então, caracterizar estes estados da seguinte maneira:

1. estado *A*: estado inicial;
2. estado *B*: chegou o primeiro 0 na entrada;
3. estado *C*: chegou o primeiro 1 na entrada;
4. estado *D*: os últimos dois bits de entrada são (0, 0), por esta ordem;

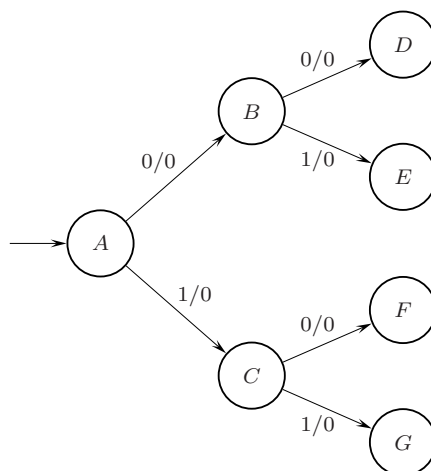


Figura 16.5: Parte inicial do diagrama de estados da máquina sequencial do Exercício 16.10

5. estado E : os últimos dois bits de entrada são $(0, 1)$, por esta ordem;
6. estado F : os últimos dois bits de entrada são $(1, 0)$, por esta ordem;
7. estado G : os últimos dois bits de entrada são $(1, 1)$, por esta ordem.

Por outras palavras, o estado D caracteriza-se por ter ocorrido a sequência $(X_{(t)}, X_{(t+1)}) = (0, 0)$ na entrada X , o estado E por ter ocorrido a sequência $(0, 1)$, o estado F por ter ocorrido a sequência $(1, 0)$ e o estado G por ter ocorrido a sequência $(1, 1)$.

Evidentemente, a saída deve vir a 0 nos estados D e E para coincidir com $X_{(t)} = 0$, e a 1 nos estados F e G para coincidir com $X_{(t)} = 1$.

Consideremos, então, o estado D , por exemplo, para o qual já ocorreu na entrada a sequência $(0, 0)$. Se ocorrer um novo 0 na entrada, devemos manter-nos no mesmo estado, já que os dois últimos bits recebidos foram $(0, 0)$. Se, pelo contrário, no estado D ocorrer um 1 na entrada, devemos passar ao estado E porque este se caracteriza por ter recebido, em último lugar, a sequência $(0, 1)$. Em ambos os casos com saída igual a 0, como acabámos de ver. Obtemos, então, o diagrama parcial da Figura 16.6.

De forma idêntica podemos concluir que as saídas no estado G , por exemplo, para o qual já ocorreu na entrada a sequência $(1, 1)$, deverão ser iguais a 1. Se ocorrer um 1 na entrada, devemos manter-nos no mesmo estado, já que os dois últimos bits recebidos foram $(1, 1)$. Se, pelo contrário, no estado G ocorrer um 0 na entrada, devemos passar ao estado F porque este se caracteriza por ter recebido, em último lugar, a sequência $(1, 0)$. Obtemos, então, o diagrama parcial da Figura 16.7.

Finalmente, podemos construir de forma semelhante o resto do diagrama, como mostra a Figura 16.8.

O diagrama de estados que se obteve não é mínimo. Com efeito, os estados A a C apenas lá estão para levar em linha de conta *os dois primeiros bits na*

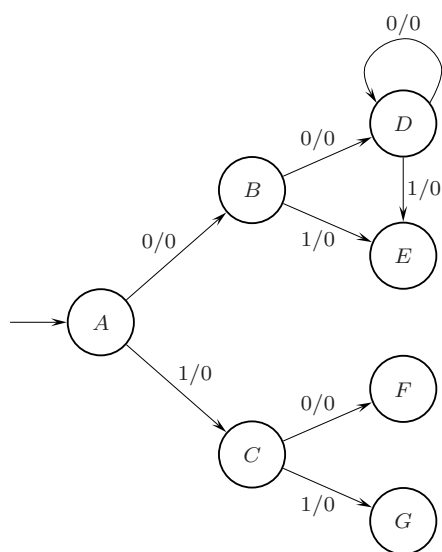


Figura 16.6: Continuação do diagrama de estados da máquina sequencial do Exercício 16.10

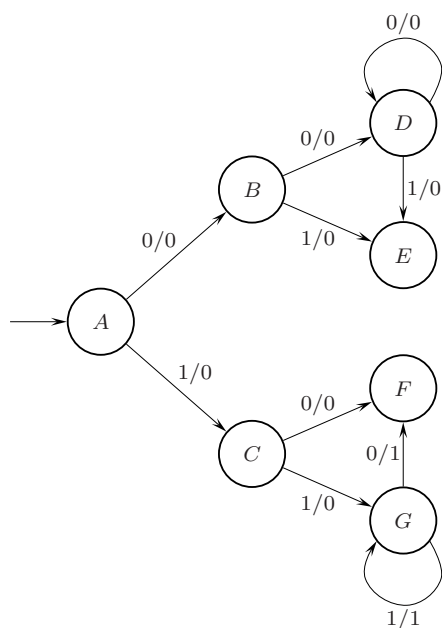


Figura 16.7: Continuação do diagrama de estados da máquina sequencial do Exercício 16.10

entrada. Pelo contrário, os estados D a G são genéricos, na medida em que a esses estados apenas interessam *os últimos dois bits de entrada*, sejam ou não os primeiros. Daí que possamos reduzir o diagrama da Figura 16.8 apenas aos estados D a G , tendo o cuidado de os redesignar e redefinir.

É o que se faz na Figura 16.9, com D a G redesignados por A a D , que passam

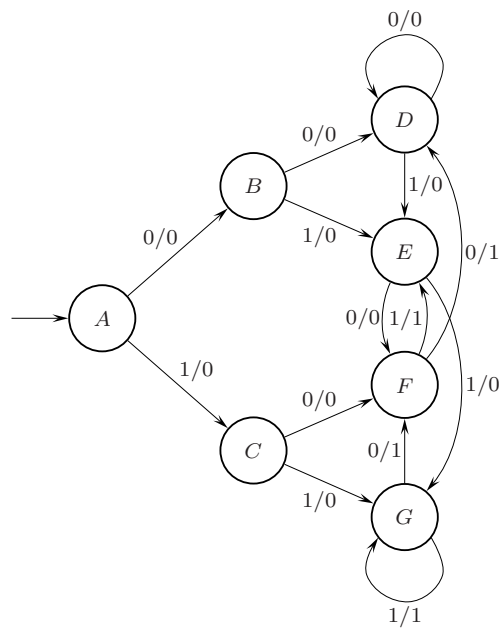


Figura 16.8: Diagrama de estados final para a máquina sequencial do Exercício 16.10

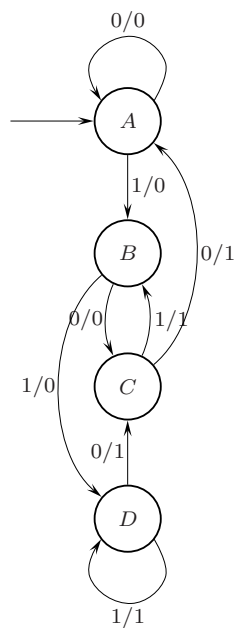


Figura 16.9: Diagrama de estados mínimo para a máquina sequencial do Exercício 16.10

a ter os seguintes significados:

1. estado A : ou não se recebeu nada, ou recebeu-se o primeiro 0, ou recebeu-se $(0, 0)$ em último lugar, por esta ordem;

2. estado B : recebeu-se $(0, 1)$ em último lugar, por esta ordem;
3. estado C : recebeu-se $(1, 0)$ em último lugar, por esta ordem;
4. estado D : recebeu-se $(1, 1)$ em último lugar, por esta ordem.

Notemos que, agora, apenas o estado A pode ser o estado inicial da máquina. Com efeito, esse estado é o único a partir do qual se geram dois zeros iniciais, situação essa imposta pela especificação do enunciado. Com efeito:

- começando pelo estado A , geramos inicialmente um 0 seguido de outro 0, porque a saída no estado A é 0 e em seguida vamos para A ou para B , em que a saída também é 0;
- se começássemos pelo estado B gerariamos inicialmente um 0 seguido de um 1, porque a saída no estado B é 0, mas em seguida iríamos para C ou para D , em que a saída é 1;
- se começássemos pelo estado C gerariamos inicialmente um 1 seguido de um 0, porque a saída no estado C é 1 e em seguida iríamos para A ou para B , em que a saída é 0; e
- se começássemos pelo estado D gerariamos inicialmente dois uns, porque a saída no estado D é 1 e em seguida iríamos para D ou para C , em que a saída também é 1.

Segue-se que apenas A pode ser o estado inicial.

Finalmente, notemos que, embora o enunciado nos obrigue a gerar diagramas de estados de Mealy, o que vamos obter realmente são máquinas de Moore. Notemos com efeito, como as saídas nos diagramas de estados das Figuras 16.8 e 16.9 se mantêm constantes em cada estado: no diagrama da Figura 16.8 os estados A e E têm sempre saída 0, e os estados F e G têm sempre saída 1, e no diagrama da Figura 16.9 os estados A e B têm sempre saída 0, e os estados C e D têm sempre saída 1.

16.12 Desenhe o diagrama de estados ou o fluxograma de uma máquina sequencial síncrona com uma entrada, X , e duas saídas, Z_0 e Z_1 , com o comportamento que se descreve em seguida: $Z_0 = 1$ apenas quando, na entrada, se verifica a sequência 1101, e $Z_0 = 0$ em todos os restantes casos; $Z_1 = 1$ apenas quando, na entrada, se verifica a sequência 1011, e $Z_1 = 0$ em todos os restantes casos. A máquina deve detectar sequências com sobreposição.

Resolução: Vamos gerar um diagrama de estados para a máquina (facilmente se pode converter esse diagrama num fluxograma, ou podemos começar por gerar o fluxograma em vez do diagrama). A máquina que vamos gerar é de Mealy (mas podia ser de Moore).

16.12

O diagrama de estados vai ser construído aos poucos. Como as duas sequências a detectar possuem um 1 inicial, que lhes é comum, começamos por considerar dois estados, A e B , indo-se de A para B apenas com esse 1 na entrada. É o que se ilustra na Figura 16.10.

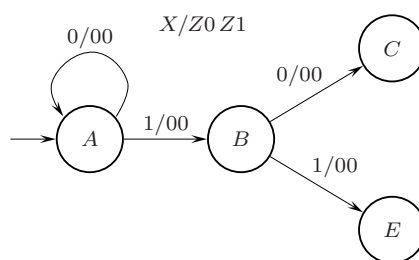


Figura 16.10: Início do diagrama de estados para a máquina de Mealy do Exercício 16.12

Consideremos agora o percurso no diagrama que vai do estado A ao estado C . Esse percurso corresponde aos dois primeiros bits da sequência 1011 na entrada X . Um terceiro bit a 1 na entrada leva-nos a um estado D . E, evidentemente, um 0 no estado D leva-nos de volta ao estado C . Podemos, então, caracterizar o estado A como sendo o estado inicial, o estado B como tendo-se recebido o primeiro 1 de uma das duas sequências, o estado C como tendo-se recebido os primeiros dois bits, 10, de uma das sequências, e o estado D como tendo-se recebido 101 dessa mesma sequência (Figura 16.11). No diagrama inclui-se ainda a transição de C para A , quando deixou de ser possível detectar a sequência 1011.

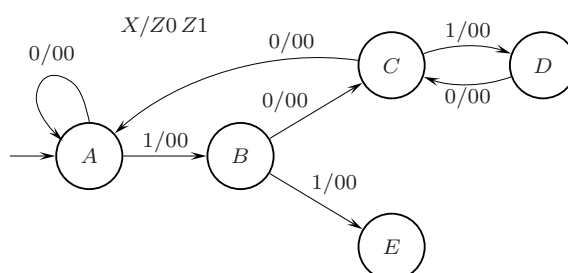


Figura 16.11: Continuação do diagrama de estados da máquina da figura anterior

Por outro lado, o estado E vem caracterizado por se terem recebido os dois primeiros bits, 11, da outra sequência. Voltemos, então, a concentrar a nossa atenção no estado D . Se, nesse estado, se receber um 1, então a sequência 1011 está formada e a saída $Z1$ deve vir a 1. Do estado D devemos, nessas circunstâncias, ir para o estado E , porque esse 1 pode ser o segundo 1 na sequência 1101. Segue-se que temos de redefinir D : ou se recebeu 101 da sequência 1011, ou se recebeu o primeiro 1 da outra sequência, 1101. Na Figura 16.12 ilustra-se a construção do diagrama de estados até este ponto.

No estado E , enquanto recebermos uns, devemos nele permanecer, já que os dois últimos uns podem ser o início da sequência 1101. E um 0 no estado E deve levar-nos para um novo estado, F , que se caracteriza por termos recebido os três primeiros bits, 110, da sequência 1101. Do estado F , com 1 na entrada, geraremos a saída $Z0$ a 1 porque essa sequência foi detectada. E de F deve ir-se para D , porque os três bits 101 que se receberam a partir do estado E estão no

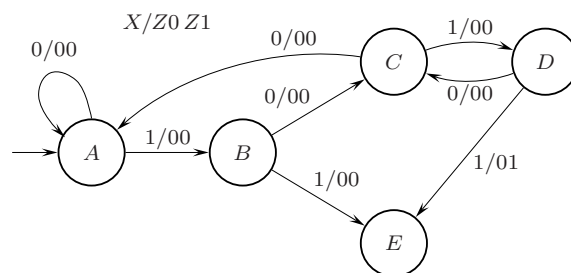


Figura 16.12: Continuação do diagrama de estados da máquina da figura anterior

bom caminho para a detecção da sequência 1011, que passa por D .

Na Figura 16.13 ilustra-se o diagrama de estados final da máquina de Mealy.

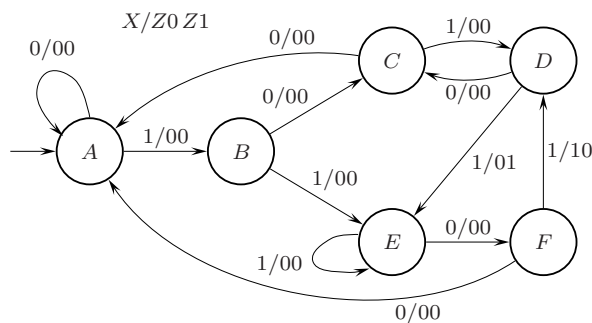


Figura 16.13: Diagrama de estados final para a máquina do Exercício 16.12

16.24 Elabore um diagrama de estados para uma máquina de Mealy que recebe em série, na sua única entrada, uma palavra qualquer do código BCD (entra em primeiro lugar o bit de maior peso), e cuja saída só vem a 1 se a palavra que entrou for inferior a $4_{(\text{BCD})}$ ou superior a $7_{(\text{BCD})}$. O valor na saída não deve vir especificado para os três primeiros bits da palavra.

Resolução: Neste exercício temos a garantia que apenas entram dígitos BCD na máquina sequencial, em série e com o bit de maior peso em primeiro lugar.

16.24

Vamos, então, considerar as 10 configurações possíveis representativas de dígitos BCD.

As sequências 0000, 0001, 0010 e 0011 (isto é, as sequências da forma 00 — —) designam dígitos cujos equivalentes decimais são inferiores a $4_{(\text{BCD})}$. Devemos, nesses casos, gerar uma saída 1 conjuntamente com o quarto bit das sequências. Outro tanto acontece com as sequências 1000 e 1001 (ou seja, as que começam por 1) que representam dígitos superiores a $7_{(\text{BCD})}$.

Pelo contrário, as sequências 0100, 0101, 0110 e 0111 (da forma 01 — —) representam dígitos compreendidos entre $4_{(\text{BCD})}$ e $7_{(\text{BCD})}$, e nesses casos devemos gerar saída 0.

Podemos, então, construir imediatamente o diagrama de estados de Mealy pretendido (Figura 16.14).

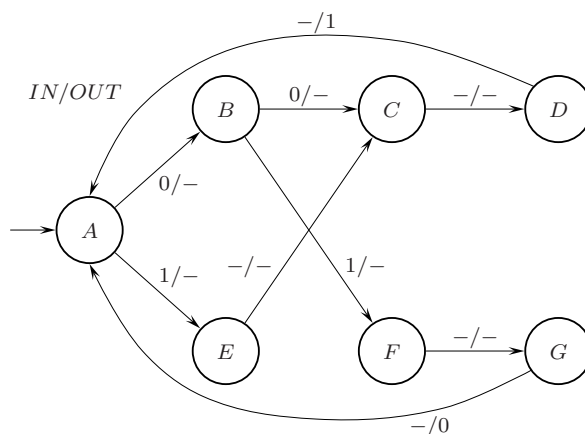


Figura 16.14: Diagrama de estados para o detector do Exercício 16.24

16.25 Desenhe o diagrama de estados de uma máquina de Mealy que detecte palavras do código BCD. As palavras, com 4 bits, entram em série por uma entrada única, começando pelo bit de maior peso. Ao fim de quatro impulsos de relógio o circuito deve vir reiniciado, preparado para detectar uma nova palavra. Para além da entrada, o circuito tem duas saídas, X e Y , tais que: (i) se $X = Y = 0$, então a palavra recebida não pertence ao código BCD; (ii) se $X = Y = 1$ a palavra pertence ao código; e (iii) se $X = 1$ e $Y = 0$ é porque não foi possível, até ao momento, saber se a palavra pertence ou não ao código.

16.25

Resolução: Consideremos as 16 configurações possíveis com 4 bits:

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
<hr/>			
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Estas combinações
correspondem a
números BCD

Estas combinações
não correspondem a
números BCD

Como o bit mais significativo entra em primeiro lugar, podemos desde logo concluir que todas as sequências binárias começadas por 0 (isto é, as sequências

0 ---) correspondem a dígitos BCD. Quanto às restantes, só as que começam por 100 (as sequências 100---) correspondem a dígitos BCD. Podemos, então começar por gerar a parte do diagrama de estados da Figura 16.15.

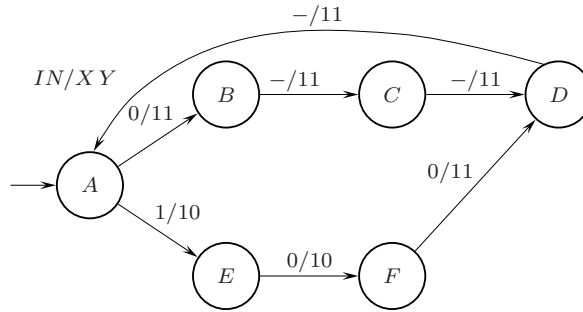


Figura 16.15: Parte inicial do diagrama de estados do detector de dígitos BCD do Exercício 16.25

Facilmente podemos agora completá-lo, como mostra a Figura 16.16.

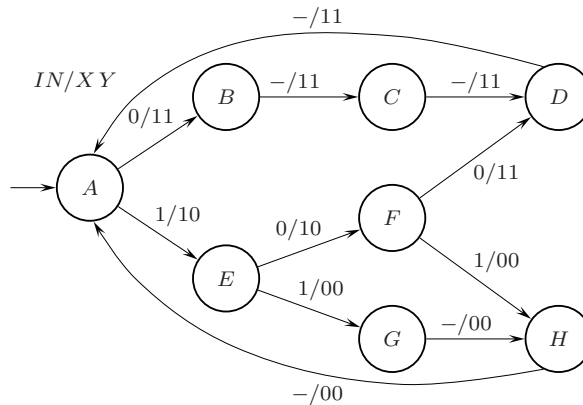


Figura 16.16: Diagrama de estados final para o detector de dígitos BCD do Exercício 16.25

16.27 Obtenha o diagrama de estados (ou o fluxograma) de uma máquina sequencial síncrona com duas entradas e duas saídas que compare dois dígitos BCD, A e B . Cada dígito é apresentado em série por uma das entradas, começando pelo bit de menor peso. Nas entradas são presentes sequências sucessivas de 4 bits. A saída deve indicar permanentemente se $A > B$, se $A < B$ ou se $A = B$.

Resolução: Para além das entradas A e B , a máquina sequencial deve ainda possuir duas saídas (digamos, S_1 e S_0) que codifiquem as três possibilidades de pares de dígitos BCD aplicados às entradas: $A > B$, $A < B$ ou $A = B$. Como não é exigido nenhum tipo de código em particular, podemos arbitrariamente escolher o seguinte:

16.27

	S1	S0
$A < B$	0	0
$A = B$	0	1
$A > B$	1	0

Para a geração do diagrama de estados desta máquina, consideremos um exemplo: à entrada A é aplicada a sequência de dígitos BCD (3, 5, 1, 9), e à entrada B a sequência (1, 5, 1, 8). Como, para cada dígito, começa por entrar o bit menos significativo, temos a seguinte situação:

A	1100	1010	1000	1001
B	1000	1010	1000	1000

\uparrow Bit mais significativo do primeiro dígito
 \uparrow Bit menos significativo do primeiro dígito

Como se pretende que a máquina vá respondendo imediatamente, usaremos o modelo de Mealy.

Inicialmente, a máquina estará num estado inicial, α . Poderá ser colocada neste estado por actuação de um Reset, por exemplo.

A partir deste estado, temos de considerar as quatro hipóteses de bits aplicáveis em A e em B . Se, nos bits que estão a entrar (os menos significativos dos dois dígitos BCD), se tiver $A = 1$ e $B = 0$, então, até ver, temos que $A > B$. Neste caso vamos para um estado, β , com o seguinte significado: “entrou o primeiro bit e $A > B$ ”. Entretanto, a máquina deverá gerar nas saídas os valores $(S1, S0) = (1, 0)$, como mostra a Figura 16.17.

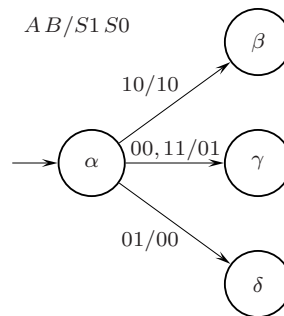


Figura 16.17: Parte inicial do diagrama de estados da máquina do Exercício 16.27

Se, no estado α , entrar $A = 0$ e $B = 0$, ou $A = 1$ e $B = 1$ temos que, até ver, $A = B$. Então, vamos de α para um estado γ , com o seguinte significado: “entrou o primeiro bit e $A = B$ ”. Naturalmente, a máquina deve gerar nas saídas os valores $(S1, S0) = (0, 1)$, como também mostra a Figura 16.17.

E se, no estado α , entrar $A = 0$ e $B = 1$, então temos que, para já, $A < B$, e vamos para um estado δ com o seguinte significado: “entrou o primeiro bit e

$A < B''$. Agora, a máquina deve gerar os valores $(S1, S0) = (0, 0)$, como ilustra a Figura 16.17.

Vamos agora estabelecer o diagrama correspondente à entrada dos segundos bits, mostrado na Figura 16.18.

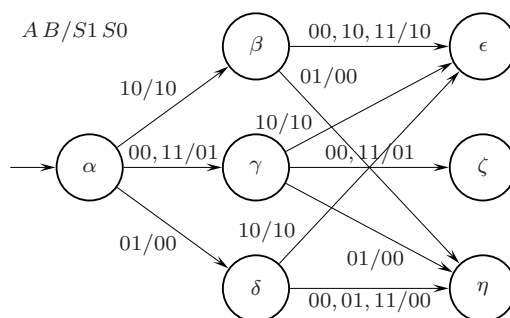


Figura 16.18: Continuação do diagrama de estados anterior

Repare-se que os estados ϵ , ζ e η representam quase o mesmo que, respectivamente, os estados β , γ e δ , com a diferença de se tratar agora dos segundos bits das sequências.

Como a entrada se faz pelo bit de menor peso, os segundos bits vão ser determinantes. Se esses bits forem iguais, tudo fica na mesma, isto é, se com o primeiro bit tínhamos $A < B$, então agora continua a ser $A < B$, e de igual modo para as outras opções. Mas se os segundos bits forem $A = 0$ e $B = 1$, então, haja o que houver do passado, temos $A < B$ a partir de agora. E de igual modo teremos $A > B$ a partir de agora se $A = 1$ e $B = 0$ nos segundos bits, independentemente do que ocorreu no passado.

O terceiro bit não é, fundamentalmente, diferente do segundo, limitando-se a introduzir uma nova camada de estados no diagrama, como mostra a Figura 16.19.

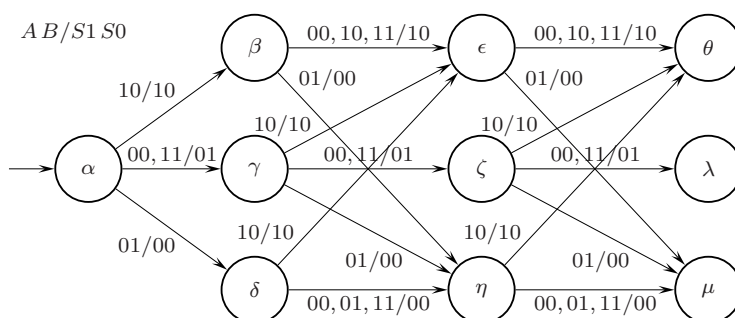


Figura 16.19: Continuação do diagrama de estados anterior

Os quartos bits vão gerar as saídas respectivas, de acordo com o esquema que se acabou de descrever. Como a máquina é de Mealy, e como por cada 4 bits (um dígito BCD) se recomeça tudo de novo, evolui-se sempre para o estado inicial, α . Obtemos, então, o diagrama de estados final da Figura 16.20.

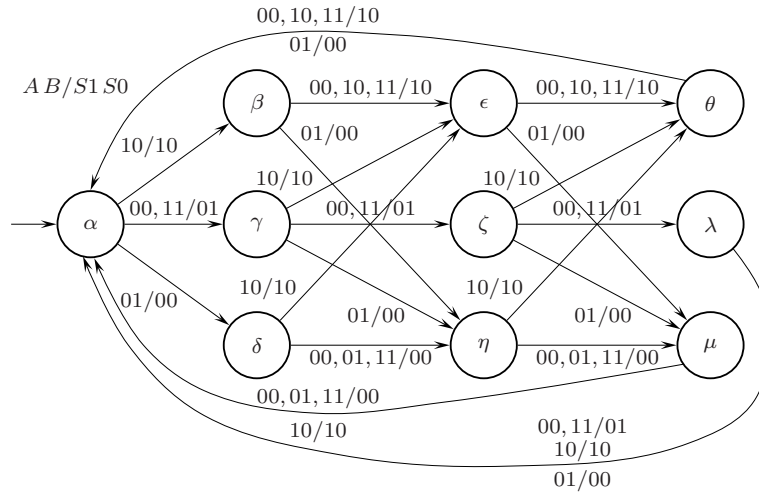


Figura 16.20: Diagrama de estados final para a máquina que compara dois dígitos BCD

16.29 Determinar um diagrama ou uma tabela de estados de uma máquina sequencial síncrona que recebe dígitos BCD a começar pelo bit menos significativo. A máquina dará saída 1 se o dígito for múltiplo de 4.

16.29

Resolução: Na Figura 16.21 apresenta-se o diagrama de estados da máquina de Mealy pretendida. Deve notar-se que apenas as sequências 0100 e 1000 (a começar pelo bit mais significativo) são dígitos BCD múltiplos de 4.

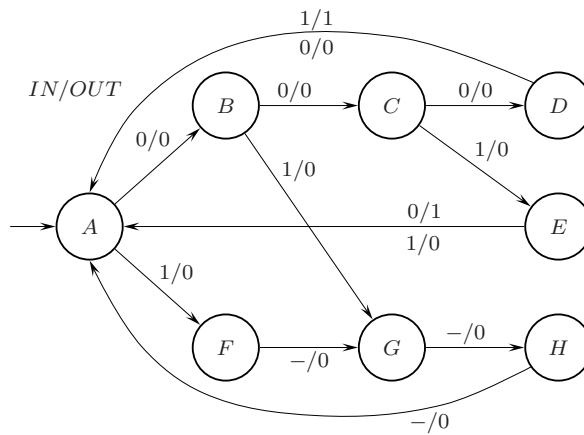


Figura 16.21: Diagrama de estados para o detector do Exercício 16.29

16.47 Utilize um contador integrado do tipo 74LS161A para implementar a máquina sequencial síncrona representada pelo diagrama de estados da Figura 16.56 (de SD:AAT).

16.47

Resolução: Para facilitar a resolução, reproduz-se na Figura 16.22 o diagrama

de estados que se pretende que o contador implemente.

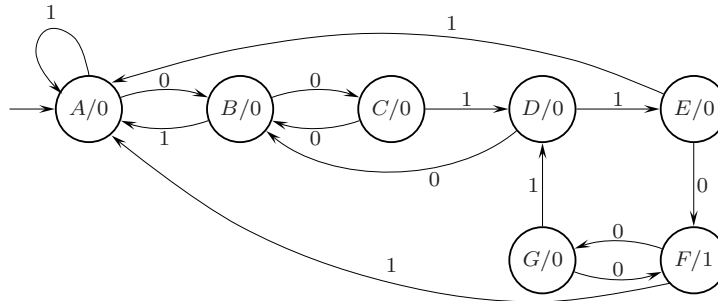


Figura 16.22: Diagrama de estados a implementar com um contador integrado do tipo 74LS161A

A resolução deste exercício passa por considerar duas lógicas combinatórias, uma lógica de detecção dos estados adequados para se fazer o carregamento em paralelo do contador, e uma outra lógica com os valores a carregar em paralelo nas entradas respectivas.

Temos, por conseguinte, uma situação em tudo idêntica à que foi utilizada no Capítulo 14 para fazer com que um contador integrado passasse a contar segundo uma sequência de estados arbitrária. Por essa razão, reproduzimos na Figura 16.23 o diagrama de blocos genérico que foi utilizado no Exercício 14.20, ou seja, o diagrama da Figura 14.5, e que serve para todas as sequências de estados que quisermos.

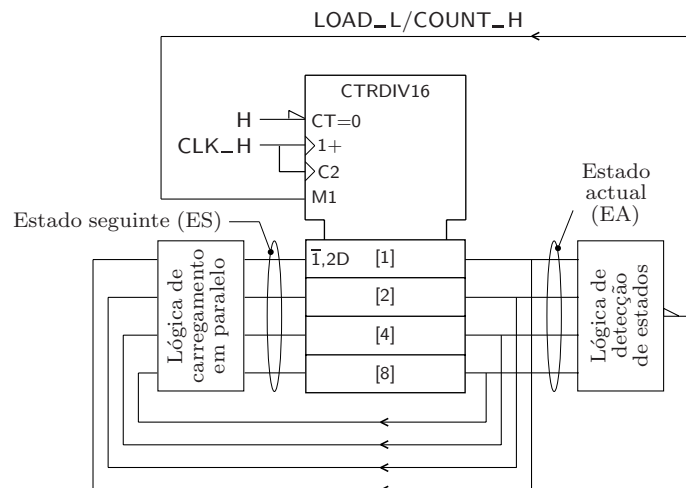
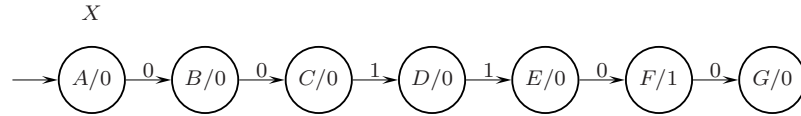


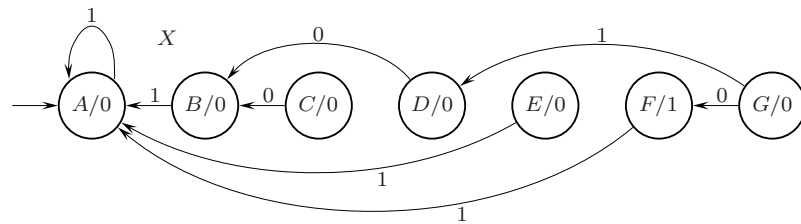
Figura 16.23: Diagrama de blocos que permite a implementação de um diagrama de estados arbitrário, feita à custa de um contador integrado que conta segundo o CBN

Com este diagrama de blocos vamos poder implementar a nossa máquina de estados. Basta que deixemos o contador contar (em CBN) os sucessivos estados da máquina, e fornecer-lhe os valores de carregamento em paralelo adequados quando não queremos que ele conte. Por exemplo, admitamos que o contador

conta em CBN a seguinte sequência de estados,



com os valores indicados para a entrada X , e que carrega em paralelo (quebra a sequência de contagem) para os outros valores de X , como se indica a seguir,



Facilmente podemos, agora, estabelecer uma tabela de transições de estados (semelhante à Tabela 14.3 do Exercício 14.28) e de saída, como se ilustra na Tabela 16.1.

Tabela 16.1: Tabela de transições e de saídas para o circuito com contador integrado que implementa o diagrama de estados da Figura 16.22. A tabela identifica as quebras da sequência de contagem e, por omissão, também a sequência de contagem, em CBN

Estado actual (EA)			Carregamento em paralelo?		Estado seguinte (ES)			Saída
Q ₂	Q ₁	Q ₀	X = 0	X = 1	P ₂	P ₁	P ₀	Z
A	0	0	0	Sim	0	0	0	0
B	0	0	1	Sim	0	0	0	0
C	0	1	0	Sim	0	0	1	0
D	0	1	1	Sim	0	0	1	0
E	1	0	0	Sim	0	0	0	0
F	1	0	1	Sim	0	0	0	1
G	1	1	0	Sim	1	0	1 se X = 0	0
G	1	1	0	Sim	0	1	1 se X = 1	0

Devemos notar que, dado que o diagrama de estados apenas tem 7 estados, nos limitamos a utilizar os 3 bits de menor peso do contador. Desta tabela podemos deduzir imediatamente a expressão da função da lógica de detecção de estados, que vai permitir o carregamento em paralelo do contador. Basta, para tanto, observar em que condições é que esse carregamento vai ser feito:

$$LOAD = A \cdot X + B \cdot X + C \cdot \overline{X} + D \cdot \overline{X} + E \cdot X + F \cdot X + G.$$

A implementação desta função pode ser feita, por exemplo, com um multiplexer.

Quanto à lógica de saída do circuito sequencial será:

$$Z = F = Q_2 \overline{Q_1} Q_0.$$

Finalmente, a lógica de carregamento em paralelo é a que se obtém pelo preenchimento dos quadros de Karnaugh da Figura 16.24, com as correspondentes minimizações. De notar que as entradas para essa lógica são X , a entrada do circuito, e as saídas Q_0 , Q_1 e Q_2 dos andares menos significativos do contador. De notar ainda que o primeiro quadro identifica o posicionamento dos estados A a G pelos quais o contador vai passar.

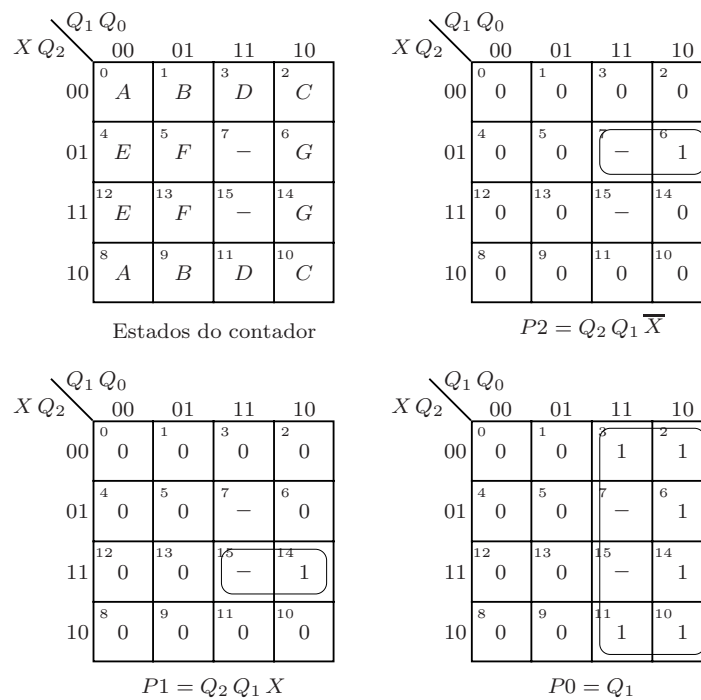


Figura 16.24: Quadros de Karnaugh que permitem obter as somas de produtos mínimos para a lógica de carregamento em paralelo

Obtemos, finalmente, o logigrama do circuito final na Figura 16.25.

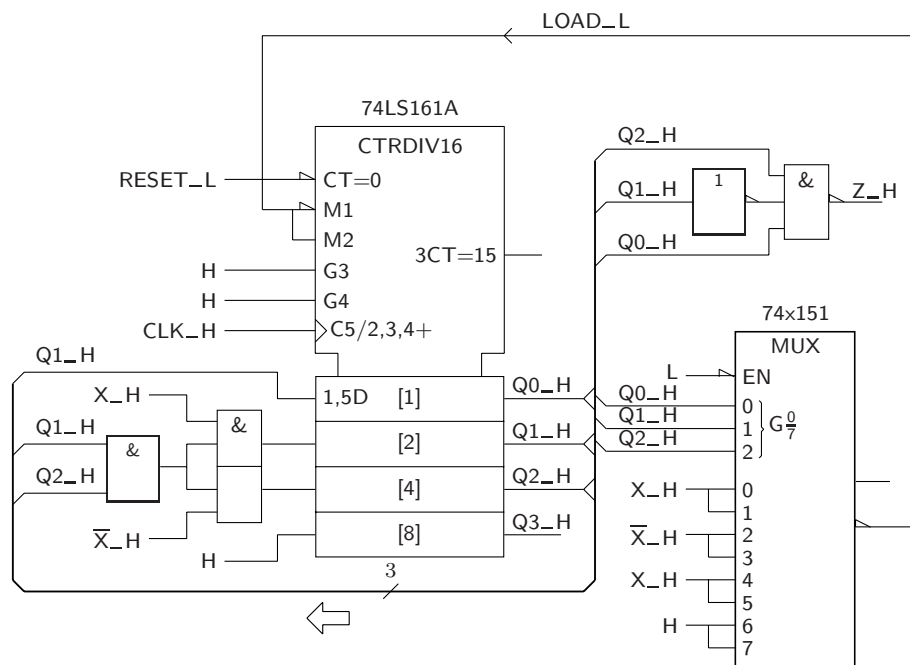


Figura 16.25: Logigrama do circuito que implementa a máquina de estados da Figura 16.22 e que utiliza um contador integrado do tipo 74LS161A

Índice Remissivo

- 74HC4024, *ver* Contador assíncrono 74HC4024
74HCT20, *ver* Portas NAND (lógica positiva) 74HCT20
74HCT21, *ver* Portas AND (lógica positiva) 74HCT21
74HCT238, *ver* Descodificador 74HCT238
74HCT32, *ver* Portas OR (lógica positiva) 74HCT32
74HCT393, *ver* Contador assíncrono 74HCT393
74HCT4002, *ver* Portas NAND (lógica positiva) 74HCT4002
74LS00, *ver* Portas NAND (lógica positiva) 74LS00
74LS10, *ver* Portas NAND (lógica positiva) 74LS10
74LS161A, *ver* Contador síncrono 74LS161A
74LS163A, *ver* Contador síncrono 74LS163A
74x00, *ver* Portas NAND (lógica positiva) 74x00
74x02, *ver* Portas NOR (lógica positiva) 74x02
74x04, *ver* Portas NOT 74x04
74x08, *ver* Portas AND (lógica positiva) 74x08
74x10, *ver* Portas NAND (lógica positiva) 74x10
74x11, *ver* Portas NAND (lógica positiva) 74x11
74x32, *ver* Portas OR (lógica positiva) 74x32
74x138, *ver* Descodificador 74x138
74x169, *ver* Contador síncrono 74x169
74x194, *ver* Registo de deslocamento universal 74x194
74x251, *ver* Multiplexer 74x251
- Adição, 4
 transporte na —, *ver* Transporte na adição
Aditivo, 6
“And-Or Invert”, 36
Arredondamento, 3
- Banco de registos, 153
Base, 2
Bit
 de sinal, 5, 6
 mais significativo, 7
 menos significativo, 3
- Ciclo
 de contagem, 143
- Circuito
 sequencial assíncrono, 115
 sequencial síncrono
 lógica de carregamento em paralelo num —, *ver* Lógica de carregamento em paralelo
 lógica de detecção de estados num —, *ver* Lógica de detecção de estados
- Código
 autocomplementar, 12
 binário, 9
 binário natural (CBN), 9, 10
 comprimento de uma palavra no —, 9
 comprimento mínimo de uma palavra no —, 9
 palavra do —, 9
 binário reflectido (CBR), 9, 10
 natural, 10
 reflectido, 9, 12
 ternário, 9
 valência de um —, *ver* Valência
- Complementação para 2, 5, 6
Complemento para 2, 5, 6
 intervalo na representação em —, *ver* Intervalo
- Contador assíncrono
 74HC4024, 129
 74HCT393, 129
 ascendente, 127
- Contador síncrono
 74LS161A, 134
 74LS163A, 132, 133, 135, 137, 139, 140, 143–145
 74x169, 146
- Contagem
 década de —, *ver* Década
- Corrida, 115

- Década, 137
 - BCD, 137
- Dependência
 - de Controlo, 119
- Descodificador
 - 74HCT238, 91
 - 74x138, 88, 89, 91, 92, 94
- Deslocamento circular, 149
- Diagrama
 - de estados
 - de um contador, 144
- Dígito
 - hexadecimal, 1
- Divisão inteira, 152
- Divisor
 - de frequência
 - por 10, 137
- Dízima, 2
- EA, *ver* Estado actual
- Entradas
 - de dados
 - de um multiplexer, 127
- Equação
 - de excitação de um flip-flop D, 122
- ES, *ver* Estado seguinte
- Estado
 - actual
 - de um contador, 130
 - inicial
 - de um contador, 143
 - seguinte
 - de um contador, 130
- Estados
 - de contagem, 127
- Flip-flop
 - SR master-slave
 - símbolo IEC de um —, 117
 - tempo de propagação de um —, 128
- Função
 - implicação, 43
- Implicação, 43
- Implicante, 44
 - não primo, *ver* Implicante primo, 44
 - primo, 44
 - essencial, 44
 - não essencial, *ver* Implicante primo
- Indicador
 - de polaridade, 56, 58–62
- Indução completa, 15
- Intervalo, 5, 6
- Latch
 - JK controlado, 112
 - comutação de um —, 114
- Linha
 - de atraso, 115
- Linhas
 - adjacentes, 11
 - simétricas, 11
- Linhas adjacentes, 22, 39
- Lógica
 - de carregamento em paralelo, 130, 141
 - de detecção de estados, 130, 140
- Matriz, 11
 - linhas adjacentes numa —, *ver* Linhas adjacentes
 - linhas de uma —, 11
 - linhas simétricas numa —, *ver* Linhas simétricas
 - ordem de uma —, 11
- Método
 - das multiplicações sucessivas, 2
- Multiplexer
 - 74x251, 106
- Notação de complemento para 2, *ver* Complemento para 2
- Notação de omplemento para 2, *ver* Complemento para 2
- Operando, 152
- Ordem
 - inversa, 10, 11
 - natural, 10, 11
- Parte
 - fraccionária, 2, 3
 - inteira, 2, 3, 152
- Pico, 117
- Porta
 - NOT (lógica positiva)
 - 74x04, 76
- Portas
 - AND (lógica positiva)
 - 74HCT21, 95
 - 74x08, 56, 76
 - 74x11, 62
 - NAND (lógica positiva)
 - 74HCT20, 92
 - 74HCT4002, 94
 - 74LS00, 143
 - 74LS10, 143
 - 74x00, 58
 - 74x10, 61, 76
 - NOR (lógica positiva)
 - 74x02, 58

- NOT
 - 74x04, 61
- OR (lógica positiva)
 - 74HCT32, 91
 - 74x32, 57
- Precisão, 2, 3
- Quadrado essencial, 46
- Qualificador
 - de entrada
 - $\bar{1}$, 2D, 130
 - 1R, 119
 - 1S, 119
 - C1, 119
 - R, 119
 - S, 119
 - de saída
 - \neg , 117
- Registo
 - de deslocamento universal, 153
 - 74x194, 153
- Registos
 - banco de —, *ver* Banco de registos
- Rotação, *ver* Deslocamento circular
- Saída
 - de Mealy, 160
 - de Moore, 159
- Sequência
 - de contagem, 144
 - de dígitos, 1
 - de estados de contagem, *ver* Sequência de contagem
- Significado
 - físico, 2
- Símbolo IEC
 - de atraso, 117
 - de um flip-flop SR master-slave, 117
 - do 74HC4024, 129
 - do 74HCT20, 92
 - do 74HCT21, 95
 - do 74HCT238, 91
 - do 74HCT32, 91
 - do 74HCT393, 129
 - do 74HCT4002, 94
 - do 74LS00, 143
 - do 74LS10, 143
 - do 74LS161A, 134
 - do 74LS163A, 132
 - do 74x00, 58
 - do 74x02, 58
 - do 74x04, 61
 - do 74x08, 56
 - do 74x10, 61
 - do 74x11, 62
 - do 74x138, 88
 - do 74x169, 146
 - do 74x194, 153
 - do 74x251, 106
 - do 74x32, 57
- Sistema
 - binário, 10
 - de numeração, 2, 4
 - base de um —, *ver* Base
 - binário, 10
 - posicional, 10
 - decimal, 3
 - hexadecimal, 1
 - ponderado, 2
 - posicional, 10
- Subtracção, 4
- Subtractivo, 6
- Tabela
 - de excitações
 - de um flip-flop A, 123
 - de transições, 140
 - de verdade física, 63
 - de verdade genérica
 - das portas XOR com 2 entradas, 62
 - $t_{pd,FF}$, 128
 - $t_{pd,MUX}$, 129
- Transporte
 - na adição, 4, 5
 - na subtracção, 7
- Valência, 9