

7. Conjunto de portas lógicas

Um sistema combinacional pode ser composto por uma rede interligada de diferentes tipos de portas lógicas. Como visto no Capítulo 4 há basicamente sete tipos de portas lógicas implementáveis em circuitos integrados: a porta NOT e as portas AND, OR e XOR e seus respectivos complementos. É possível se pensar em implementar sistemas digitais usando apenas um conjunto restrito dessas portas. Por exemplo, pode-se pensar em implementar um circuito digital usando apenas portas lógicas dos tipos NOT e AND ou apenas as do tipo XOR e AND e assim por diante. É possível então, formalizar o conceito de conjunto de portas lógicas.

Definição 7.0.1 (Conjunto de portas lógicas) Elenco (possivelmente restrito) de tipos de portas lógicas a ser considerado na implementação de um sistema digital.

A notação usada para conjuntos é utilizado aqui para se expressar os conjuntos de portas lógicas. A título de exemplo, pode-se pensar em três conjuntos arbitrários A , B e C , em que $A = \{\text{NOT}, \text{AND}\}$, $B = \{\text{NAND}\}$, $C = \{\text{OR}, \text{AND}, \text{NOT}, \text{XOR}\}$. Ao se considerar cada um dos conjuntos exemplos A , B e C na implementação de um sistema se está na verdade, forçando a implementação do referido sistema utilizando unicamente as portas constantes em cada conjunto. É possível que se queira implementar um sistema arbitrário S utilizando apenas as portas do conjunto A ou do conjunto C por decisão de projeto.

Decisões para a adoção de um conjunto restrito de portas lógicas para se implementar um sistema podem se basear em aspectos como: restrições de velocidade de resposta, custo, número de níveis de portas que sinal vai passar desde a entrada até a saída do sistema, etc.

7.1 Conjunto universal

Um questionamento surge a partir da formalização da ideia de conjunto de portas lógicas: quais são ou conjuntos de portas lógicas capazes de implementar qualquer função combinacional?

De fato, existem vários conjuntos capazes de implementar qualquer função combinacional. Esses conjuntos, recebem a denominação de conjunto universal formalizada na Definição 7.1.1.

Tabela 7.1: Implementação das portas OR e AND aplicando o teorema de De Morgan.

Porta	Implementação usando De Morgan
$x_0 + x_1 + \dots + x_{n-1}$	$\overline{\overline{x_0} \cdot \overline{x_1} \cdot \dots \cdot \overline{x_{n-1}}}$
$x_0 \cdot x_1 \cdot \dots \cdot x_{n-1}$	$\overline{\overline{x_0} + \overline{x_1} + \dots + \overline{x_{n-1}}}$

Definição 7.1.1 (Conjunto universal de portas lógicas) É um conjunto de portas lógicas capaz de implementar qualquer função combinacional.

A álgebra de boole usada para descrever funções de chaveamento e sistemas digitais tem como funções básicas AND, OR e NOT. Consequentemente, É possível se implementar qualquer função combinacional utilizando essas três portas. Ou seja, o conjunto

$$U_b = \{\text{NOT, AND, OR}\}$$

é um conjunto universal de acordo com a Definição 7.1.1. O conjunto U_b é o conjunto universal básico vindo diretamente dos operadores empregados na álgebra de boole usada em circuitos digitais e é referenciado algumas vezes neste texto.

Pode-se demonstrar que o conjunto arbitrário de portas X é um conjunto universal se for possível, utilizando apenas as portas do conjunto X , implementar todas as portas de um conjunto universal conhecido. A seguir, são elencados alguns conjuntos de portas lógicas, demonstrando sua universalidade utilizando como conjunto universal conhecido ou conjunto U_b .

Antes porém, vale salientar a aplicação do teorema de De Morgan para a conversão de portas lógicas. A Aplicação do teorema permite realizar conversão entre portas do tipo OR e AND e vice versa. Retomando as expressões do Teorema 2.2.10 vem:

$$\begin{aligned} \overline{x_0 + x_1 + \dots + x_{n-1}} &= \overline{x_0} \cdot \overline{x_1} \cdot \dots \cdot \overline{x_{n-1}} \\ \text{e} \\ \overline{x_0 \cdot x_1 \cdot \dots \cdot x_{n-1}} &= \overline{x_0} + \overline{x_1} + \dots + \overline{x_{n-1}}. \end{aligned} \tag{7.1}$$

É possível complementar ambos os lados (aplicando a unicidade dos complementos, Teorema 2.2.2)

e obter:

$$\begin{aligned} x_0 + x_1 + \dots + x_{n-1} &= \overline{\overline{x_0} \cdot \overline{x_1} \cdot \dots \cdot \overline{x_{n-1}}} \\ \text{e} \\ x_0 \cdot x_1 \cdot \dots \cdot x_{n-1} &= \overline{\overline{x_0} + \overline{x_1} + \dots + \overline{x_{n-1}}}. \end{aligned} \quad (7.2)$$

A conclusão dessas últimas expressões é de que é possível implementar uma porta OR usando uma porta AND e alguns complementos e é possível implementar uma porta AND usando portas OR e alguns complementos. A Tabela 7.1 mostra essa ideia relacionando as portas OR e AND (primeira coluna) às suas implementações utilizando, respectivamente, as portas AND e OR, por aplicação direta das expressões mostradas em (7.2).

A seguir são mostrados quatro conjuntos universais de portas lógicas e as respectivas demonstrações de suas universalidade usando como base o conjunto universal conhecido U_b e as transformações mostradas na Tabela 7.1.

7.1.1 Conjunto {NOT, AND}

Essas duas portas formam um conjunto universal. A implementação das portas do conjunto universal U_b usando apenas {NOT, AND} estão mostradas na Tabela 7.2 o que demonstra que esse é um conjunto universal.

Tabela 7.2: Implementação das portas do conjunto universal conhecido U_b usando apenas as portas do conjunto de portas {NOT, AND}

Conjunto universal conhecido U_b	Implementação apenas com portas de {NOT, AND}
$x \rightarrow \overline{x}$	$x \rightarrow \overline{x}$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$

7.1.2 Conjunto {NOT, OR}

Essas duas portas formam um conjunto universal. A implementação das portas do conjunto universal U_b usando apenas {NOT, OR} estão mostradas na Tabela 7.3 o que demonstra que esse é um conjunto universal.

Tabela 7.3: Implementação das portas do conjunto universal conhecido U_b usando apenas as portas do conjunto de portas $\{\text{NOT, OR}\}$

Conjunto universal conhecido U_b	Implementação apenas com portas de $\{\text{NOT, OR}\}$
$x \rightarrow \bar{x}$	$x \rightarrow \bar{x}$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$

7.1.3 Conjunto {NAND}

A porta NAND sozinha forma um conjunto universal. A implementação das portas do conjunto universal U_b usando apenas $\{\text{NAND}\}$ estão mostradas na Tabela 7.4 o que demonstra que esse é um conjunto universal. Perceba que é possível implementar uma porta de complementação (NOT) usando apenas NAND usando a lei da idempotência da álgebra de boole (Teorema 2.2.5) como mostrado na primeira linha da Tabela 7.4.

Tabela 7.4: Implementação das portas do conjunto universal U_b usando apenas portas NAND.

Conjunto universal conhecido U_b	Implementação apenas com portas de $\{\text{NAND}\}$
$x \rightarrow \bar{x}$	$x \rightarrow \bar{x}$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$

7.1.4 Conjunto {NOR}

A porta NOR sozinha forma um conjunto universal. A implementação das portas do conjunto universal U_b usando apenas {NOR} estão mostradas na Tabela 7.5 o que demonstra que esse é um conjunto universal. Perceba que é possível implementar uma porta de complementação (NOT) usando apenas NAND usando a lei da idempotência da álgebra de boole (Teorema 2.2.5) como mostrado na primeira linha da Tabela 7.5

Tabela 7.5: Implementação das portas do conjunto universal conhecido U_b usando apenas as portas do conjunto de portas {NOR}

Conjunto universal conhecido U_b	Implementação apenas com portas de {NOR}
$x \rightarrow \overline{x}$	$x \rightarrow \overline{\overline{x}}$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$
$x_0, x_1, \dots, x_{n-1} \rightarrow z$	$x_0, x_1, \dots, x_{n-1} \rightarrow z$

7.2 Implementação de circuitos com conjuntos universais {NAND} e {NOR}

Como visto nas seções anteriores, é possível implementar qualquer sistema combinacional usando apenas porta NAND ou apenas portas NOR. A questão que surge agora é como se obter algebricamente essa implementações particulares. Se faz necessário, portanto, o estabelecimento de um procedimento que seja capaz de transformar uma função de chaveamento qualquer em uma expressão contendo apenas portas do tipo NAND ou apenas portas do tipo NOR. Isso pode ser feito usando de forma sistemática os seguintes teoremas da álgebra de boole:

- Lei da involução (Teorema 2.2.6);
- Teorema de De Morgan (Teorema 2.2.10).

Para formalizar da procedimento, considere uma função $f(x_{n-1}, \dots, x_1, x_0)$ arbitrária de n variáveis que se quer descrever apenas em função de operações NAND ou OR. Primeiramente, aplica-se a lei da involução da álgebra de boole à função de chaveamento da seguinte forma:

$$f(x_{n-1}, \dots, x_1, x_0) = \overline{\overline{f(x_{n-1}, \dots, x_1, x_0)}} \quad (7.3)$$

As complementações que aparecem no lado direito de (7.3) pode ser usadas sucesivamente para a aplicação do teorema de De Morgan até que a expressão esteja de uma forma que só dependa

de operações NAND ou só de operações NOR. O Exemplo 7.1 mostra a técnica sendo aplicada a uma função de chaveamento específica, escrevendo-a usando apenas portas do tipo NAND e o Exemplo 7.2 mostra a técnica sendo aplicada a uma função de chaveamento específica, escrevendo-a usando apenas portas do tipo NOR

■ **Exemplo 7.1** Escreva a função $w = f(x, y, z) = \bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ usando apenas portas do tipo NAND. Aplicando a técnica descrita, o primeiro passo a ser feito é a aplicação do teorema da involução à expressão que se quer transformar, ou seja, fazer $w = \overline{\overline{w}}$. Ao se fazer isso tem-se:

$$w = \overline{\overline{w}} = \overline{\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z}}. \quad (7.4)$$

Feito isso, aplica-se o teorema de De Morgan ao lado direito de (7.4) obtendo-se:

$$\overline{\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z}} = \overline{(\bar{x} \cdot y \cdot z) \cdot (\bar{x} \cdot y) \cdot (\bar{y} \cdot z)}. \quad (7.5)$$

Perceba que a barra mais de baixo do lado esquerdo de (7.5) foi usada para se aplicar o teorema de De Morgan à expressão $\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ pois, pelo referido teorema:

$$\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z} = (\bar{x} \cdot y \cdot z) \cdot (\bar{x} \cdot y) \cdot (\bar{y} \cdot z). \quad (7.6)$$

Para finalizar, perceba que o lado direito de (7.5) está escrita de tal forma que apenas são necessárias portas do tipo NAND para sua implementação. ■

■ **Exemplo 7.2** Escreva a função $w = f(x, y, z) = \bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ usando apenas portas do tipo NOR.

Aplicando a técnica descrita, o primeiro passo a ser feito é a aplicação do teorema da involução à expressão que se quer transformar, ou seja, fazer $w = \overline{\overline{w}}$. Ao se fazer isso tem-se:

$$w = \overline{\overline{w}} = \overline{\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z}}. \quad (7.7)$$

Feito isso, aplica-se o teorema de De Morgan ao lado direito de (7.7) obtendo-se:

$$\overline{\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z}} = \overline{(\bar{x} \cdot y \cdot z) \cdot (\bar{x} \cdot y) \cdot (\bar{y} \cdot z)}. \quad (7.8)$$

Perceba que a barra mais de baixo do lado esquerdo de (7.8) foi usada para se aplicar o teorema de De Morgan à expressão $\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ pois, pelo referido teorema:

$$\overline{\bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z} = (\bar{x} \cdot y \cdot z) \cdot (\bar{x} \cdot y) \cdot (\bar{y} \cdot z). \quad (7.9)$$

Aplicações sucessivas do teorema de De Morgan são necessárias para transformar o lado direito de (7.8) em um formato em que apenas portas NOR sejam necessárias na implementação:

$$\begin{aligned} \overline{(\bar{x} \cdot y \cdot z) \cdot (\bar{x} \cdot y) \cdot (\bar{y} \cdot z)} &= \overline{(x + \bar{y} + \bar{z}) \cdot (x + \bar{y}) \cdot (y + \bar{z})} \\ &= \overline{(x + \bar{y} + \bar{z})} + \overline{(x + \bar{y})} + \overline{(y + \bar{z})}. \end{aligned} \quad (7.10)$$

O lado direito da primeira linha é resultado da aplicação do teorema de De Morgan à cada uma das expressões $\bar{x} \cdot y \cdot z$, $\bar{x} \cdot y$ e $\bar{y} \cdot z$, enquanto que o lado direito da segunda linha é resultado da aplicação do teorema de De Morgan à expressão $(x + \bar{y} + \bar{z}) \cdot (x + \bar{y}) \cdot (y + \bar{z})$. Para finalizar, perceba que o lado direito de (7.10) está escrita de tal forma que apenas portas do tipo NOR precisam ser usada para sua implementação. ■

7.2.1 Notação de funções para implementação NAND e NOR

A implementação de funções de chaveamento em sistemas digitais usando apenas portas do tipo NAND ou NOR é bastante importante. As portas desse tipo são implementadas de forma mais simples que suas contrapartes não complementadas AND e OR nos circuitos integrados. Assim, as portas NAND/NOR, em geral, mais rápidas e mais econômicas em termos de gasto de energia do que suas contrapartes. Além disso, é bastante útil usar implementações apenas as portas NAND/NOR pois com apenas um único tipo de circuitos integrado é possível se implementar qualquer função.

Para evidenciar a característica de se realizar a implementação de um circuito usando apenas portas do tipo NAND ou NOR podem se estabelecer uma notação especial para esse propósito. Define-se aqui a função $\text{NAND}(x_{n-1}, x_{n-2}, \dots, x_0)$, para representar uma operação NAND de até n variáveis e a função $\text{NOR}(x_{n-1}, x_{n-2}, \dots, x_0)$ para representar uma operação NOR de até n variáveis. Definindo formalmente:

$$\text{NAND}(x_{n-1}, x_{n-2}, \dots, x_0) = \overline{x_{n-1} \cdot x_{n-2} \cdot \dots \cdot x_0}, \quad (7.11)$$

e

$$\text{NOR}(x_{n-1}, x_{n-2}, \dots, x_0) = \overline{x_{n-1} + x_{n-2} + \dots + x_0}. \quad (7.12)$$

Perceba que essa as funções podem ser usadas para representar um número de variável de variáveis independentes. Por exemplo, pode-se escrever $\text{NAND}(a, b)$ para representar a operação $\overline{a \cdot b}$ ou se escrever $\text{NOR}(x_3, x_2, x_1, x_0)$ para se representar $\overline{x_3 + x_2 + x_1 + x_0}$.

As Tabelas 7.6 e 7.7 mostram, respectivamente, a escrita dos três operadores básicos de álgebra de boole, NOT, AND e OR, usando de funções NAND(\cdot) e NOR(\cdot). Os resultado mostrados nas tabelas decorrem diretamente da aplicação dos teoremas da involução e de De Morgan como descrito no início da Seção 7.2.

Tabela 7.6: Escrita dos três operadores básicos de álgebra de boole, NOT, AND e OR, usando a função NAND(\cdot).

Função lógica	Função lógica escrita apenas com NAND(\cdot)
\overline{x}	NAND(x, x)
$x_{n-1} \cdot x_{n-2} \cdot \dots \cdot x_0$	NAND(NAND($x_{n-1}, x_{n-2}, \dots, x_0$), NAND($x_{n-1}, x_{n-2}, \dots, x_0$))
$x_{n-1} + x_{n-2} + \dots + x_0$	NAND(NAND(x_{n-1}, x_{n-1}), NAND(x_{n-2}, x_{n-2}), ..., NAND(x_0, x_0))

Tabela 7.7: Escrita dos três operadores básicos de álgebra de boole, NOT, AND e OR, usando a função NOR(\cdot)).

Função lógica	Função lógica escrita apenas com NOR(\cdot)
\overline{x}	NOR(x, x)
$x_{n-1} \cdot x_{n-2} \cdot \dots \cdot x_0$	NOR(NOR(x_{n-1}, x_{n-1}), NOR(x_{n-2}, x_{n-2}), ..., NOR(x_0, x_0))
$x_{n-1} + x_{n-2} + \dots + x_0$	NOR(NOR($x_{n-1}, x_{n-2}, \dots, x_0$), NOR($x_{n-1}, x_{n-2}, \dots, x_0$))

É possível perceber que as expressões escritas nas segundas colunas das Tabelas 7.6 e 7.7 dependem exclusivamente das funções NAND(\cdot) NOR(\cdot) e das variáveis x_n , evidenciando a implementação do sistema usando exclusivamente portas do tipo NAND ou NOR.

■ **Exemplo 7.3** Escreva a função $w = f(x,y,z) = \bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ usando as variáveis x, y e z e a função NAND(\cdot).

Do Exemplo 7.1 é possível se concluir que w pode ser assim escrito:

$$w = \overline{(\bar{x} \cdot y \cdot z)} \cdot \overline{(\bar{x} \cdot y)} \cdot \overline{(\bar{y} \cdot z)}. \quad (7.13)$$

Primeiramente, observe que algumas variáveis no lado direito de (7.13) precisam passar por uma operação NOT individualmente, como são os casos das variáveis x e y . Essa operações NOT em variáveis isoladas podem ser feitas apenas com NAND(\cdot) como mostrado na primeira linha da Tabela 7.6. Substituindo as operações NAND (e as NOTs isoladas) do lado direito de (7.13) pela função NAND(\cdot) chega-se a:

$$\begin{aligned} w &= \overline{(\bar{x} \cdot y \cdot z)} \cdot \overline{(\bar{x} \cdot y)} \cdot \overline{(\bar{y} \cdot z)} \\ &= \text{NAND}(\text{NAND}(\text{NAND}(x,x),y,z),\text{NAND}(\text{NAND}(x,x),y),\text{NAND}(\text{NAND}(y,y),z)) \end{aligned} \quad (7.14)$$

■

■ **Exemplo 7.4** Escreva a função $w = f(x,y,z) = \bar{x} \cdot y \cdot z + \bar{x} \cdot y + \bar{y} \cdot z$ usando as variáveis x, y e z e a função NOR(\cdot).

Do Exemplo 7.2 é possível se concluir que w pode ser assim escrito:

$$w = \overline{(x+\bar{y}+\bar{z})} + \overline{(x+\bar{y})} + \overline{(y+\bar{z})}. \quad (7.15)$$

Primeiramente, observe que algumas variáveis no lado direito de (7.15) precisam passar por uma operação NOT individualmente, como são os casos das variáveis y e z . Essa operações NOT em variáveis isoladas podem ser feitas apenas com NOR(\cdot) como mostrado na primeira linha da Tabela 7.7. Substituindo as operações NOR (e as NOTs isoladas) do lado direito de (7.15) pela função NOR(\cdot) pode-se escrever \bar{w} assim:

$$\bar{w} = \text{NOR}(\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(z,\text{NOR}(z,z)),\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(y,\text{NOR}(z,z))). \quad (7.16)$$

Como $w = \text{NOR}(\bar{w},\bar{w})$ pode-se escrever:

$$\begin{aligned} w &= \text{NOR}(\text{NOR}(\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(z,\text{NOR}(z,z))),\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(y,\text{NOR}(z,z))), \\ &\quad \text{NOR}(\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(z,\text{NOR}(z,z)),\text{NOR}(x,\text{NOR}(y,y)),\text{NOR}(y,\text{NOR}(z,z)))) \end{aligned} \quad (7.17)$$

■

7.3 Problemas propostos

Problema 7.1 Determine se cada um dos conjuntos de portas lógicas a seguir são ou não conjuntos universais.

- a) {XOR}
- b) {XNOR}
- c) {XOR,AND}
- d) {XOR,OR}

Problema 7.2 Aplique a técnica mostrada na Seção 7.2 e encontre uma expressão algébrica equivalente às funções mostradas a seguir que podem ser implementadas apenas com portas do tipo NAND.

- a) $f(x, y, z) = \bar{x} \cdot y \cdot z + x \cdot \bar{y} \cdot z + y \cdot z.$
 b) $f(x, y, z, w) = (x+y+z+\bar{w}) \cdot (\bar{x}+y+w) \cdot (\bar{y}+\bar{z}).$
 c) $f(x, y, z) = x \oplus y \oplus z$
 d) $f(x_3, x_2, x_1, x_0) = \text{Conjunto-UM}(1, 5, 10).$
 e) $f(x_2, x_1, x_0) = \text{Conjunto-ZERO}(3, 5, 6).$

Problema 7.3 Aplique a técnica mostrada na Seção 7.2 e encontre uma expressão algébrica equivalente às funções mostradas a seguir que podem ser implementadas apenas com portas do tipo NOR.

- a) $f(x, y, z) = \bar{x} \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z} + y \cdot z.$
 b) $f(x, y, z, w) = (x+y+\bar{z}+w) \cdot (\bar{x}+\bar{y}+w) \cdot (\bar{y}+z).$
 c) $f(x, y, z) = \overline{x \oplus y \oplus z}$
 d) $f(x_3, x_2, x_1, x_0) = \text{Conjunto-ZERO}(2, 7, 11).$
 e) $f(x_2, x_1, x_0) = \text{Conjunto-UM}(1, 2, 3).$

Problema 7.4 Encontre as expressões algébricas e as implementações com portas lógicas solicitadas em cada ítem:

- a) Tabela verdade mostrada na Tabela 7.8a usando apenas portas do tipo NAND.
 b) Tabela verdade mostrada na Tabela 7.8a usando apenas portas do tipo NOR.
 c) Tabela verdade mostrada na Tabela 7.8b usando apenas portas do tipo NAND.
 d) Tabela verdade mostrada na Tabela 7.8b usando apenas portas do tipo NOR.

Tabela 7.8: Tabelas verdades consideradas no exercício 7.4

i	x_3	x_2	x_1	x_0	$f(x_3, x_2, x_1, x_0)$
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

(a)

i	x_3	x_2	x_1	x_0	$f(x_3, x_2, x_1, x_0)$
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

(b)

Problema 7.5 Aplique a técnica mostrada na Seção 7.2.1 e, usando apenas a função $\text{NAND}(\cdot)$ e as variáveis independentes, escreva uma expressão para:

- a) $f(x, y, z) = \bar{x} \cdot y \cdot z + x \cdot \bar{y} \cdot z + y \cdot z.$
- b) $f(x, y, z, w) = (x+y+z+\bar{w}) \cdot (\bar{x}+y+w) \cdot (\bar{y}+\bar{z}).$

Problema 7.6 Aplique a técnica mostrada na Seção 7.2.1 e, usando apenas a função $\text{NOR}(\cdot)$ e as variáveis independentes, escreva uma expressão para:

- a) $f(x, y, z) = \bar{x} \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z} + y \cdot z.$
- b) $f(x, y, z, w) = (x+y+\bar{z}+w) \cdot (\bar{x}+\bar{y}+w) \cdot (\bar{y}+z).$

8. Introdução à simplificação e mapas-K

8.1 Redes de dois níveis

Como visto no Capítulo 6 qualquer função de chaveamento pode ser implementada por expressões algébricas na forma SPD ou PDS. É possível se usar o conceito de funções de chaveamento para modelar o funcionamento de sistemas combinacionais digitais. Essa modelagem pode expressar um rede de portas lógicas fisicamente interconectadas em um *hardware* de um sistema digital.

Mais especificamente, funções/expressões de chaveamento tipo SPD (PDS) são implementadas em *hardware* por redes de portas lógicas com um nível de portas NOT, um segundo nível de portas AND (OR) e um terceiro nível de portas OR (AND). Partindo da premissa que as variáveis complementadas e não complementadas estão disponíveis na entrada, o primeiro nível de portas NOT não formalmente contabilizado e se define que redes nesse formato são redes de portas de dois níveis. Esquematizando os dois tipos possíveis de topologia para redes de dois níveis tem-se:

Soma de produtos	
Primeiro nível	Segundo nível
AND	OR

Produto de somas	
Primeiro nível	Segundo nível
OR	AND

Neste capítulo são desenvolvidos os principais conceitos usados para obtenção de redes (sistemas digitais) mínimas de portas lógicas de dois níveis (*i.e.* na forma soma da produtos ou na forma produto de somas).

8.2 Redes mínimas de dois níveis

A técnica para obtenção de sistemas combinacionais mínimos de dois níveis é baseada em algumas premissas que são discutidas nesta seção.

Dois procedimentos práticos para, de fato, se obter implementações mínimas de circuitos digitais são tratados no capítulo seguinte (Capítulo ??).

As premissas assumidas para se obter redes mínimas são:

- A topologia da rede é fixa, ou na forma SDP ou na forma PDS.

- As entradas do sistema estão disponíveis nas formas complementadas e não complementadas (*i.e.* o número de portas inversoras na rede não é um objetivo de minimização).
- As redes têm apenas uma saída.
- O objetivo é minimizar o número total de portas a AND ou OR e o número de entradas dessas portas.

No próximo capítulo (Capítulo ??) são desenvolvidas e mostradas técnicas para se encontrar redes mínimas de dois níveis. Elas são baseadas na aplicação sistemática do seguinte teorema da álgebra de boole:

$$\begin{aligned} A \cdot B + A \cdot \bar{B} &= A \quad (\text{soma de produtos}) \\ \text{e} \\ (A + B) \cdot (A + \bar{B}) &= A \quad (\text{produto de somas}). \end{aligned} \tag{8.1}$$

Note que se A for um termo produto de n literais na primeira expressão, por exemplo $A = x_{n-1} \cdot \dots \cdot x_1 \cdot x_0$, ou um termo soma de n literais na segunda expressão, por exemplo $A = x_{n-1} + \dots + x_1 + x_0$, as expressões continuam válidas:

$$\begin{aligned} x_{n-1} \cdot \dots \cdot x_1 \cdot x_0 \cdot B + x_{n-1} \cdot \dots \cdot x_1 \cdot x_0 \cdot \bar{B} &= x_{n-1} \cdot \dots \cdot x_1 \cdot x_0 \quad (\text{soma de produtos}) \\ \text{e} \\ (x_{n-1} + \dots + x_1 + x_0 + B) \cdot (x_{n-1} + \dots + x_1 + x_0 + \bar{B}) &= x_{n-1} + \dots + x_1 + x_0 \quad (\text{produto de somas}). \end{aligned} \tag{8.2}$$

A conclusão que se chega é de que a soma de dois termos produtos p_1 e p_2 que possuem exatamente as mesmas literais excetuando-se uma única literal que aparece complementada em p_1 e não complementada em p_2 (ou vice-versa) é igual a p_1 (ou p_2) com a variável que “mudou” de complementada para não complementada removida de p_1 (ou p_2).

Conclusão semelhante se chega no produto de somas: o soma de dois termos soma s_1 e s_2 que possuem exatamente as mesmas literais excetuando-se uma única literal que aparece complementada em s_1 e não complementada em s_2 (ou vice-versa) é igual a s_1 (ou s_2) com a variável que “mudou” de complementada para não complementada removida de s_1 (ou s_2). Para ficar mais claro alguns exemplos da aplicação desse teorema são:

$$\begin{aligned} x \cdot y \cdot \bar{z} + x \cdot y \cdot z &= x \cdot y \quad (\text{apenas a variável } z \text{ mudou, foi eliminada}) \\ x \cdot \bar{y} \cdot z \cdot \bar{w} + x \cdot y \cdot z \cdot \bar{w} &= x \cdot z \cdot \bar{w} \quad (\text{apenas a variável } y \text{ mudou, foi eliminada}) \\ (x + y + \bar{z}) \cdot (x + y + z) &= x + y \quad (\text{apenas a variável } z \text{ mudou, foi eliminada}) \\ (x + \bar{y} + z + \bar{w}) \cdot (x + y + z + \bar{w}) &= x + z + \bar{w} \quad (\text{apenas a variável } y \text{ mudou, foi eliminada}). \end{aligned} \tag{8.3}$$

Note que em todos os casos mostrados em (8.3) os dois termos que aparecem nos lados esquerdos das igualdades são exatamente iguais exceto que em um dos termos uma única variável aparece complementada e no outro não. A regra do teorema só vale para exatamente uma única mudança desse tipo. Se houver duas mudanças como em:

$$x \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z = ?, \tag{8.4}$$

(ambas as variáveis x e z mudaram entre os dois termos produtos) a regra não se aplica.

As Definições 8.2.1 e 8.2.2 introduzem o conceito de distância que pode ser usado para formalizar o que foi discutido até aqui.

Definição 8.2.1 (Distância entre dois termos produtos) Considere dois termos produtos p_1 e p_2 . Encontre a representação binária B_1 de p_1 associando com cada variável complementada de p_1 o bit 0 e com cada variável não complementada de p_1 o bit 1. Faça o mesmo com p_2 e obtenha B_2 . A distância entre os termos produtos p_1 e p_2 é definida como o número de bits diferentes quando uma comparação bit a bit entre B_1 e B_2 é realizada.

Definição 8.2.2 (Distância entre dois termos soma) Considere dois termos soma s_1 e s_2 . Encontre a representação binária B_1 de s_1 associando com cada variável complementada de s_1 o bit 1 e com cada variável não complementada de s_1 o bit 0. Faça o mesmo com s_2 e obtenha B_2 . A distância entre os termos soma s_1 e s_2 é definida como o número de bits diferentes quando uma comparação bit a bit entre B_1 e B_2 é realizada.

Ou seja, a identidade mostrada em (8.1) pode ser aplicada a uma soma de dois termos produtos que possuem distância exatamente igual a 1 ou ao produto de dois termos soma que possuem distância exatamente igual a 1.

Perceba que a aplicação de (8.1) resulta em minimização do sistema a ser implementado pois dois termos de n literais são transformados em um único termo de $n - 1$ literais.

É bem intuitivo entender que o conceito de distância mostrado nas Definições 8.2.1 e 8.2.2 também se aplica para *strings* binárias, o qual será usado nas seções seguintes.

8.3 Mapas de Karnaugh

É possível se organizar a tabela verdade de um sistema de uma forma especial que seja possível se identificar exatamente em que situações de uma função de chaveamento é permitido aplicar a identidade (8.1). Essa identificação se dá por inspeção, ou seja, apenas olhando para a tabela. Essa nova forma de organizar a tabela verdade é chamada de mapa de Karnaugh ou mapa-K.

Explica-se aqui a elaboração do mapa-K a partir de uma tabela verdade arbitrariamente escondida e mostrada na Tabela 8.2a. Perceba que é possível rescrever a Tabela 8.2a usando a Tabela 8.2b apenas fazendo mudanças nas disposições das variáveis, dispondo duas delas nas linhas (x_3 e x_2) e duas nas colunas (x_1 e x_0). O valor contido em cada célula da Tabela 8.2b é o valor da função quando as variáveis independentes tem valores definidos nas linhas e colunas da tabela. Em suma, a Tabela 8.2a de 16 linhas é transformada na Tabela 8.2b de 4 linhas e 4 colunas.

Do ponto de vista de sistemas, cada célula da Tabela 8.2b corresponde à saída do sistema quando a ele são apresentados, nas entradas, os bits constantes nas linha e colunas da tabela. Por exemplo, a célula da primeira linha e primeira coluna da Tabela 8.2b tem valor igual a 1. Isso corresponde à resposta do sistema quando a entrada $x_3x_2x_1x_0 = 0000$ é a ele apresentada.

Para facilitar o uso de (8.1) e aplica-la diretamente à tabela, por inspeção, é interessante que a Tabela 8.2b possua a seguinte propriedade: duas células vizinhas (*i.e.* adjacentes nas linha ou nas colunas) representem padrões de atribuição de variáveis independentes (*strings* binárias) da função com distância igual a 1. Além disso, que essa propriedade se repita para qualquer célula c_1 na tabela com relação a qualquer célula vizinha a c_1 escolhida na tabela.

Infelizmente, a Tabela 8.2b não possui tal propriedade. Por exemplo, as células $a_{2,1}$ (linha dois coluna um) e $a_{3,1}$ (linha três coluna um) são vizinhas (mesma coluna e linhas sucessivas). No entanto, à célula $a_{2,1}$ corresponde a atribuição de variáveis $x_3x_2x_1x_0 = 0100$ e à célula $a_{3,1}$,

Tabela 8.2: Construção de um mapa-K de 4 variáveis.

i	x_3	x_2	x_1	x_0	$f(x_3, x_2, x_1, x_0)$		
0	0	0	0	0	1		
1	0	0	0	1	0		
2	0	0	1	0	1		
3	0	0	1	1	0		
4	0	1	0	0	1		
5	0	1	0	1	0		
6	0	1	1	0	1		
7	0	1	1	1	0		
8	1	0	0	0	1		
9	1	0	0	1	0		
10	1	0	1	0	1		
11	1	0	1	1	0		
12	1	1	0	0	0		
13	1	1	0	1	0		
14	1	1	1	0	0		
15	1	1	1	1	0		

x_3	x_2	x_1	x_0
00	00	01	10
01	01	10	11
10	10	10	10
11	11	00	00

(b)

x_3	x_2	x_1	x_0
00	00	01	11
01	01	10	01
11	11	00	00
10	10	10	01

(c)

(a)

corresponde a atribuição de variáveis $x_3x_2x_1x_0 = 1000$. Ou seja, a distância entre 0100 e 1000 é dois e não um como desejado para células vizinhas.

Ao se trocar de posições entre a quarta e terceira linhas e entre a quarta e terceira coluna na Tabela 8.2b obtém-se a Tabela 8.2c. Observe na Tabela 8.2c que quaisquer células vizinhas (nas colunas ou nas linhas) possuem atribuição de variáveis independentes com distância igual a 1 e gozam da propriedade necessária para a aplicação de (8.1) diretamente na tabela, como desejado inicialmente. O mapa assim disposto é chamado de mapa-K.

O mapa-K mostrado na Tabela 8.2c representa um mapa-K para 4 variáveis. A notação para esse mapa-K pode ser simplificada destacando-se as linhas ou colunas nas quais uma determinada variável tem valor igual a 1. No mapa-K mostrado na Fig. 8.1a estão destacadas as linhas nas quais as variáveis x_3 e x_2 são iguais a 1 e as colunas nas quais as variáveis x_1 e x_2 são iguais a 1 posicionando essas variáveis nas bordas do mapa. Por exemplo, as duas últimas linhas estão marcadas com x_3 , indicando que a variável x_3 tem valor igual a 1 nessas linhas, enquanto que nas duas primeiras linhas não há marcação da variável x_3 o que indica que, para essas linhas, x_3 vale 0. A mesma análise pode ser feita para as demais variáveis. A Fig. 8.1b simplifica ainda mais a notação mostrada na Fig. 8.1a ao apenas indicar, por meio de barras, as linhas e colunas nas quais as variáveis x_n têm valor igual a 1 (e, consequentemente, têm valor 0 nas linhas/colunas não indicadas).

Por meio do exemplo anterior foi mostrado como construir um mapa-K capaz de representar uma função de 4 variáveis. No entanto, podem ser montados mapas-K de múltiplas variáveis como mostrado na Seção 8.4.

Uma definição geral para o mapa-K é que ele constitui uma forma de se expressar a tabela verdade de um sistema que torna mais simples a identificação de células cujas atribuições de variáveis têm distância igual a 1.

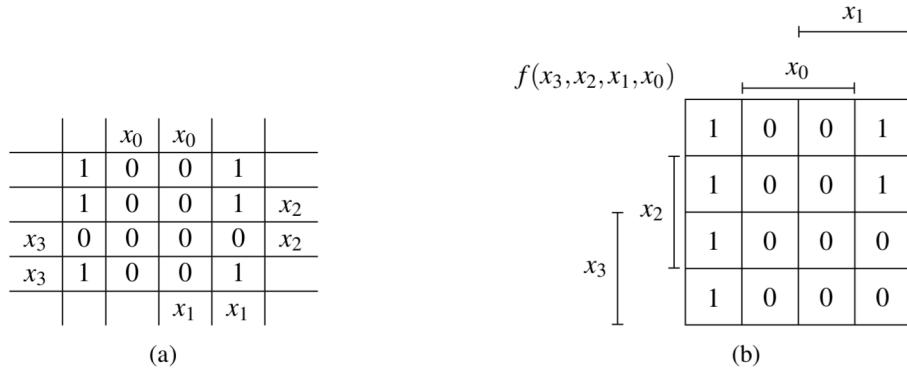


Figura 8.1: Versões simplificadas do mapa de Karnaugh de 4 variáveis.

- **Exemplo 8.1** (Preenchimento do mapa-K a partir do conjunto um de uma função) Preencha a o mapa de Karnaugh para a função $f(x_3, x_2, x_1, x_0) = \text{Conjunto-UM}(1, 5, 9, 14)$

Para resolver esse problema é preciso identificar a ordem das variáveis e os índices indicados no conjunto um. A ordem indicada das variáveis é $x_3x_2x_1x_0$. A Fig. 8.2a mostra a tabela de indexação decimal para cada célula do mapa-K considerando essa ordem de variáveis. As atribuições de variáveis independentes que resultam em 1 são os números {1,5,9,14} que, transformados para binário, resultam nos números {0001,0101,1001,1110}. A Fig. 8.2b mostra o preenchimento do mapa-K com 0s e 1s. É possível entender esse preenchimento de duas maneiras:

1. Os 1s são colocados na tabela mostradas na Fig. 8.2b nos respectivos índices (*i.e.* {1, 5, 9, 14}) indicados no conjunto um da função (e mostrados na Fig. 8.2a).
 2. Os 1s são colocados na tabela mostrada na Fig. 8.2b nas linhas/colunas respectivas às palavras binárias {0001, 0101, 1001, 1110} (conjunto um da função) obtidas se compondo os bits indicados nas linhas com os indicados nas colunas da tabela mostrada na Fig. 8.2b.

		x_1x_0						x_1x_0			
		00	01	11	10			00	01	11	10
x_3x_2	00	0	1	3	2	x_3x_2	00	0	1	0	0
	01	4	5	7	6		01	0	1	0	0
	11	12	13	15	14		11	0	0	0	1
	10	8	9	11	10		10	0	1	0	0

(a) Indexação do mapa-K.

(b) Mapa-K da função $f(x_3, x_2, x_1, x_0)$.

Figura 8.2: Mapas-K usados no Exemplo 8.1.

■ **Exemplo 8.2** (Preenchimento do mapa-K a partir do conjunto zero de uma função) Preencha a o mapa de Karnaugh para a função $f(x_3, x_2, x_1, x_0) = \text{Conjunto-ZERO}(0, 2, 3, 6)$

Para resolver esse problema é preciso identificar a ordem das variáveis e os índices indicados no conjunto um. A ordem indicada das variáveis é $x_3x_2x_1x_0$. A Fig. 8.3a mostra a tabela de indexação decimal para cada célula do mapa-K considerando essa ordem de variáveis. As atribuições de variáveis independentes que resultam em 0 são os números $\{0, 2, 3, 6\}$ transformados para binário, o que resulta nos padrões binários $\{0000, 0010, 0011, 0110\}$. A Fig. 8.3b mostra o preenchimento do mapa-K com 0s e 1s. É possível entender esse preenchimento de duas maneiras:

1. Os 0s são colocados na tabela mostradas na Fig. 8.3b nos respectivos índices (*i.e.* $\{0, 2, 3, 6\}$) indicados no conjunto um da função (e mostrados na Fig. 8.3a)
2. Os 1s são colocados na tabela mostrada na Fig. 8.3b nas linhas/colunas respectivas às palavras binárias $\{0000, 0010, 0011, 0110\}$ (que formam o conjunto um da função) obtidas se compondo os *bits* indicados nas linhas com os indicados nas colunas da tabela mostrada na Fig. 8.3b.

x_1x_0				x_1x_0									
		00	01	11	10			00	01	11	10		
x_3x_2		00	0	1	3	2	x_3x_2		00	0	1	0	0
		01	4	5	7	6			01	1	1	1	0
11	12	13	15	14	11	1	1	1	1				
10	8	9	11	10	10	1	1	1	1				

(a) Indexação do mapa-K.

(b) Mapa-K da função $f(x_3, x_2, x_1, x_0)$.

Figura 8.3: Mapas-K usados no Exemplo 8.2.

8.4 Exemplos de mapas de Karnaugh de duas a seis variáveis

Na seção anterior foi construído um mapa-K representativo para uma função de 4 variáveis. No entanto, pode-se construir mapas-K para mais ou menos variáveis. Exemplos desses mapa-K estão mostrados na figura 8.4 a 8.7.

Os exemplos das figuras mostram mapas-K de 2 variáveis (Fig. 8.4a), de 3 variáveis (Fig. 8.4b), de 4 variáveis (Fig. 8.5), de 5 variáveis (Fig. 8.6), de 6 variáveis (Fig. 8.7). Nas células de cada mapa encontra-se indicada a linha a qual a célula está ligada na tabela verdade da função (*i.e.* indexação da linha).

Note que o mapa-K para 5 variáveis é formado por dois mapas de 4 variáveis. As variáveis x_3, x_2, x_1, x_0 ocupam as mesmas posições nos dois mapas. O que diferencia um mapa do outro é o valor da quinta variável x_4 , que no mapa da esquerda vale 0 e no mapa da direita vale 1.

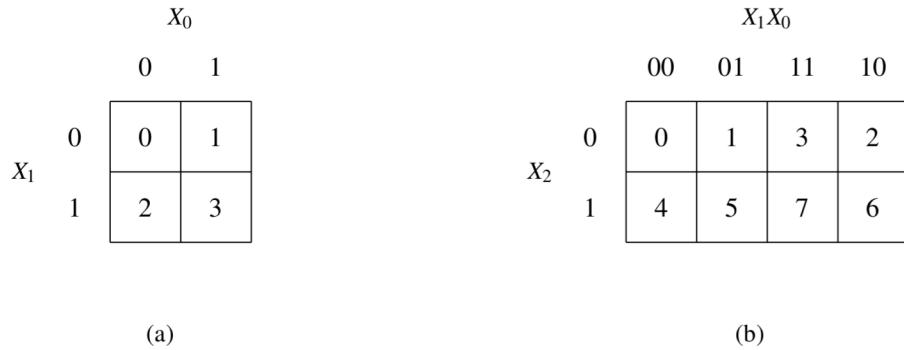


Figura 8.4: Mapa de Karnaugh de: (a) 2 variáveis e (b) 3 variáveis.

		X_1X_0				
		00	01	11	10	
		00	0	1	3	2
		01	4	5	7	6
X_3X_2		11	12	13	15	14
		10	8	9	11	10

Figura 8.5: Mapa de Karnaugh de 4 variáveis.

		X_1X_0				
		00	01	11	10	
		00	0	1	3	2
		01	4	5	7	6
X_3X_2		11	12	13	15	14
		10	8	9	11	10

		X_1X_0			
		00	01	11	10
		16	17	19	18
		20	21	23	22
$X_4=0$		28	29	31	30
		24	25	27	26

		X_1X_0			
		00	01	11	10
		16	17	19	18
$X_4=1$		20	21	23	22
		28	29	31	30
		24	25	27	26

Figura 8.6: Mapa de Karnaugh de 5 variáveis.

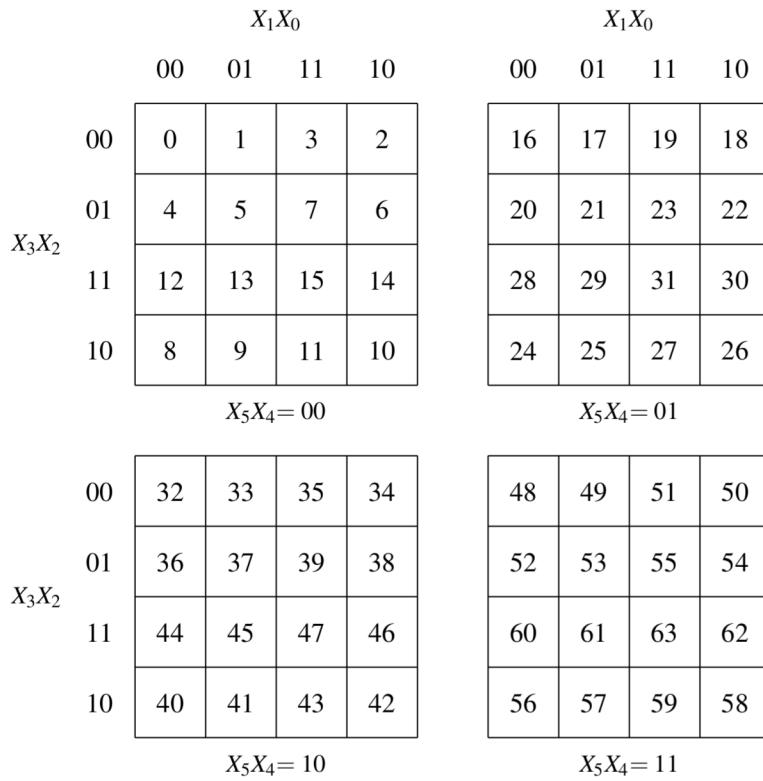


Figura 8.7: Mapa de Karnaugh de 6 variáveis.

O mapa-K para 6 variáveis segue ideia similar, mas com quatro mapas-K de 4 variáveis. Novamente as variáveis x_3, x_2, x_1, x_0 ocupam as mesmas posições nos quatro mapas. O que diferencia um mapa dos outros são os valores das variáveis x_5 e x_4 , que valem 00 (alto esquerda), 01 (alto direita), 10 (baixo esquerda), 11 (baixo direita).

8.5 Agrupamentos de 1s em mapas de Karnaugh

Os mapas-K são organizados de forma que as atribuições de variáveis referentes a células vizinhas no mapa-K têm distância igual a 1. Ou seja, se houver dois 1s lado a lado no mapa-K esses 1s correspondem a termos produtos (mintermos) com distância igual a um, cabendo portanto a aplicação de (8.1) à soma desses termos.

Essa ideia é desenvolvida aqui por meio de um exemplo prático. Considere a função $f(x_3, x_2, x_1, x_0) = \text{Conjunto-UM}(5, 7, 13, 15)$, cujo mapa-K está mostrado na Fig. 8.8a. Como há quatro mintermos na composição dessa função, ela pode ser escrita na forma canônica SDP assim:

$$f(x_3, x_2, x_1, x_0) = \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 + \overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0 + x_3 \cdot x_2 \cdot x_1 \cdot x_0. \quad (8.5)$$

Observe que na Fig. 8.8a cada mintermo está destacado com uma cor: $m_5 = \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0$ (em vermelho), $m_7 = \overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0$ (em verde), $m_{13} = x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0$ (em amarelo), $m_{15} = x_3 \cdot x_2 \cdot x_1 \cdot x_0$ (em azul).

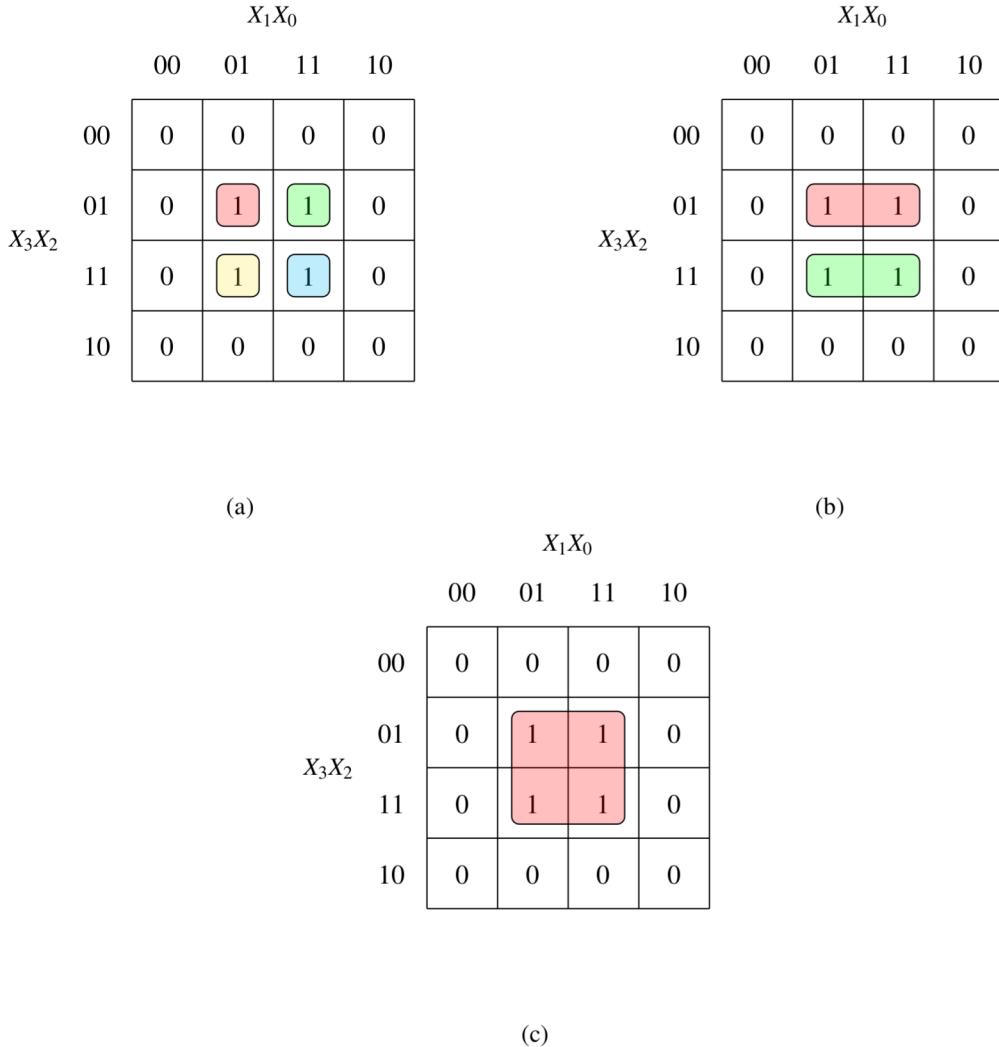


Figura 8.8: Exemplo de agrupamentos de 1s em mapas-K.

azul). Os mintermos m_5 e m_7 são termos produtos de distância 1 assim como os mintermos m_{13} e m_{15} . Aplicando (8.1) em cada par se chega a:

$$f(x_3, x_2, x_1, x_0) = \bar{x}_3 \cdot x_2 \cdot x_0 + x_3 \cdot x_2 \cdot x_0. \quad (8.6)$$

O termo $\bar{x}_3 \cdot x_2 \cdot x_0$ de (8.6) se originou da aplicação de (8.1) à soma dos mintermos m_5 e m_7 , enquanto que o termo $x_3 \cdot x_2 \cdot x_0$ se originou da aplicação de (8.1) à soma dos mintermos m_{13} e m_{15} . Note que o termo $\bar{x}_3 \cdot x_2 \cdot x_0$ corresponde ao agrupamento destacado em vermelho na Fig. 8.8b, enquanto que o termo $x_3 \cdot x_2 \cdot x_0$ corresponde ao agrupamento destacado em verde na Fig. 8.8b. Fica claro, portanto, que as simplificações decorrentes de (8.1) podem ser obtidas diretamente do mapa-K por meio de agrupamentos de 1s.

Todavia, um agrupamento ainda maior pode ser feito. Perceba que (8.6) contém a soma de dois

termos produtos de distância 1. Portanto, (8.1) pode ser novamente aplicada a (8.6) resultando em:

$$f(x_3, x_2, x_1, x_0) = x_2 \cdot x_0. \quad (8.7)$$

Ou seja, o termo $x_2 \cdot x_0$ corresponde ao agrupamento mostrado em vermelho na Fig. 8.8c, que pode ser visto como o agrupamento das duplas mostradas em vermelho e verde na Fig. 8.8b.

Os agrupamentos propostos por (8.1) são feitos sempre de dois em dois termos produtos como mostrado no exemplo. Assim, a identidade pode ser aplicada a qualquer agrupamento contendo 2^k 1s vizinhos no mapa. Ou seja, podem ser agrupados 1s sozinhos, duplas de 1s vizinhos, quartetos de 1s vizinhos, octetos de 1s vizinhos e assim por diante.

Um agrupamento de 2^k 1s vizinhos gera a eliminação de k variáveis no termo produto correspondente. Se o mapa-K representa uma função de n variáveis, o termos produto correspondente ao agrupamento contém $n - k$ variáveis. No exemplo discutido, a função considerada é de 4 variáveis ($n = 4$), os agrupamentos de $2^0 = 1$ 1s vizinhos na Fig. 8.8a geram 4 termos produtos de 4 variáveis (sem eliminação), os agrupamentos de $2^1 = 2$ 1s vizinhos na Fig. 8.8b geram 2 termos produtos de 3 variáveis (eliminação de uma variável) e o agrupamento de $2^2 = 4$ 1s vizinhos na Fig. 8.8c gera 1 termo produto de 2 variáveis (eliminação de duas variáveis).

8.5.1 Extração da expressão referente a um agrupamento de 1s no mapa-K

A extração da expressão correspondente a um agrupamento de 1s identificado em um mapa-K se dá pela análise das variáveis envolvidas no agrupamento. O termo produto referente a um agrupamento de 1s tem as seguintes características:

- As variáveis que permanecem iguais a 1, para todos os 1s do agrupamento, entram no termo produto correspondente ao agrupamento sem barras.
- As variáveis que permanecem iguais a 0, para todos os 1s do agrupamento, entram no termo produto correspondente ao agrupamento com barras.
- As variáveis que não tem valor fixo para todos os 1s do agrupamento são eliminadas do termo produto.

Por exemplo, o agrupamento em vermelho mostrado na Fig. 8.8b tem $x_3 = 0$ para todos os 1s do agrupamento (x_3 entra com barra), $x_2 = 1$ para todos os 1s do agrupamento (x_2 entra sem barra), $x_0 = 1$ para todos os 1s do agrupamento (x_0 entra sem barra) e $x_1 = 0$ para um 1 do agrupamento e $\underline{x_1} = 1$ para o outro sendo portanto eliminada. Aplicando esses achados, chega-se ao termo produto $\overline{x_3} \cdot x_2 \cdot x_0$ correspondente à dupla de 1s em vermelho na Fig. 8.8b.

8.6 Agrupamentos de 0s em mapas de Karnaugh

O agrupamento de zeros em mapas-K é bastante similar ao agrupamento de uns. Novamente os mapas-K são organizados de forma que as atribuições de variáveis referentes a células vizinhas têm distância igual a 1. Ou seja, dois 0s lado a lado no mapa-K correspondem a termos soma (maxtermos) com distância igual a um, cabendo portanto a aplicação de (8.1) ao produto desses termos.

Essa ideia é novamente desenvolvida aqui por meio de um exemplo prático. Considere a função $f(x_3, x_2, x_1, x_0) = \text{Conjunto-ZERO}(5, 7, 13, 15)$, cujo mapa-K está mostrado na Fig. 8.9a. Como há quatro maxtermos na composição dessa função, ela pode ser escrita na forma canônica PDS assim:

$$f(x_3, x_2, x_1, x_0) = (x_3 + \overline{x_2} + x_1 + \overline{x_0}) \cdot (x_3 + \overline{x_2} + \overline{x_1} + \overline{x_0}) \cdot (\overline{x_3} + \overline{x_2} + x_1 + \overline{x_0}) \cdot (\overline{x_3} + \overline{x_2} + \overline{x_1} + \overline{x_0}). \quad (8.8)$$

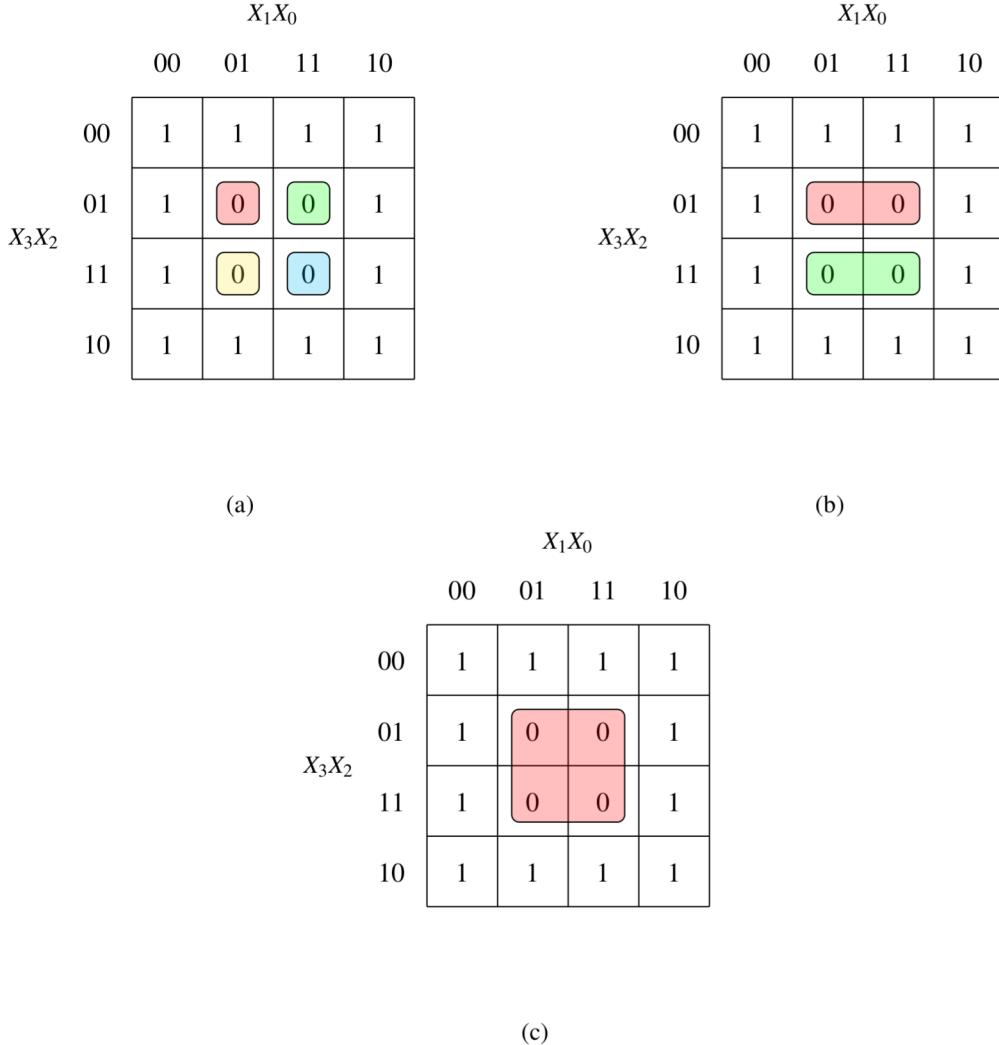


Figura 8.9: Exemplo de agrupamentos de 0s em mapas-K.

Observe que na Fig. 8.9a cada maxtermo está destacado com uma cor: $M_5 = x_3 + \bar{x}_2 + x_1 + \bar{x}_0$ (em vermelho), $M_7 = x_3 + \bar{x}_2 + \bar{x}_1 + \bar{x}_0$ (em verde), $M_{13} = \bar{x}_3 + \bar{x}_2 + x_1 + \bar{x}_0$ (em amarelo), $M_{15} = \bar{x}_3 + \bar{x}_2 + \bar{x}_1 + \bar{x}_0$ (em azul). Os maxtermos M_5 e M_7 são termos soma de distância 1 assim como os maxtermos M_{13} e M_{15} . Aplicando (8.1) em cada par se chega a:

$$f(x_3, x_2, x_1, x_0) = (x_3 + \bar{x}_2 + \bar{x}_0) \cdot (\bar{x}_3 + \bar{x}_2 + \bar{x}_0). \quad (8.9)$$

O termo $x_3 + \bar{x}_2 + \bar{x}_0$ em (8.9) se origina da aplicação de (8.1) ao produto dos maxtermos M_5 e M_7 , enquanto que o termo $\bar{x}_3 + \bar{x}_2 + \bar{x}_0$ se origina da aplicação de (8.1) ao produto dos maxtermos M_{13} e M_{15} . Note que o termo $x_3 + \bar{x}_2 + \bar{x}_0$ corresponde ao agrupamento destacado em vermelho na Fig. 8.9b, enquanto que o termo $\bar{x}_3 + \bar{x}_2 + \bar{x}_0$ corresponde ao agrupamento destacado em verde na Fig. 8.9b. Fica claro, portanto, que as simplificações decorrentes de (8.1) podem ser obtidas diretamente do

mapa-K por meio de agrupamentos de 0s.

Todavia, um agrupamento ainda maior pode ser feito. Perceba que (8.9) contém o produto de dois termos soma de distância 1. Portanto, (8.1) pode ser novamente aplicada a (8.9) resultando em:

$$f(x_3, x_2, x_1, x_0) = \overline{x_2} + \overline{x_0}. \quad (8.10)$$

Ou seja, o termo $\overline{x_2} + \overline{x_0}$ corresponde ao agrupamento mostrado em vermelho na Fig. 8.9c, que pode ser visto como o agrupamento das duplas mostradas em vermelho e verde na Fig. 8.9b.

Os agrupamentos propostos por (8.1) são feitos sempre de dois em dois termos soma como mostrado no exemplo. Assim, a identidade pode ser aplicada a qualquer agrupamento contendo 2^k 0s vizinhos no mapa. Ou seja, podem ser agrupados 0s sozinhos, duplas de 0s vizinhos, quartetos de 0s vizinhos, octetos de 0s vizinhos e assim por diante.

Um agrupamento de 2^k 0s vizinhos gera a eliminação de k variáveis no termo soma correspondente. Se o mapa-K representa uma função de n variáveis, o termos soma correspondente ao agrupamento conterá $n - k$ variáveis. No exemplo discutido, a função considerada é de 4 variáveis ($n = 4$), os agrupamentos de $2^0 = 1$ 0s vizinhos na Fig. 8.9a geram 4 termos soma de 4 variáveis (sem eliminação), os agrupamentos de $2^1 = 2$ 0s vizinhos na Fig. 8.9b geram 2 termos soma de 3 variáveis (eliminação de uma variável) e o agrupamento de $2^2 = 4$ 0s vizinhos na Fig. 8.9c gera 1 termo soma de 2 variáveis (eliminação de duas variáveis).

8.6.1 Extração da expressão referente a um agrupamento de 0s no mapa-K

A extração da expressão correspondente a um agrupamento de 0s identificado em um mapa-K se dá pela análise das variáveis envolvidas no agrupamento. O termo soma referente a um agrupamento de 0s tem as seguintes características:

- As variáveis que permanecem iguais a 1, para todos os 0s do agrupamento, entram no termo soma correspondente ao agrupamento com barras.
- As variáveis que permanecem iguais a 0, para todos os 0s do agrupamento, entram no termo soma correspondente ao agrupamento sem barras.
- As variáveis que não tem valor fixo para todos os 0s do agrupamento são eliminadas do termo soma.

Por exemplo, o agrupamento em vermelho mostrado na Fig. 8.9b tem $x_3 = 0$ para todos os 0s do agrupamento (x_3 entra sem barra), $x_2 = 1$ para todos os 0s do agrupamento (x_2 entra com barra), $x_0 = 1$ para todos os 0s do agrupamento (x_0 entra com barra) e $x_1 = 0$ para um 0 do agrupamento e $x_1 = 1$ para o outro sendo portanto eliminada. Aplicando esses achados, chega-se ao termo soma $x_3 + \overline{x_2} + \overline{x_0}$ correspondente à dupla de 0s em vermelho na Fig. 8.9b.

8.7 Problemas propostos

Problema 8.1 Preencha os mapas-K para as seguintes funções:

- a) $f(x_2, x_1, x_0) = \text{Conjunto-UM}(0, 3, 6)$
- b) $f(x_2, x_1, x_0) = \text{Conjunto-ZERO}(1, 3, 5, 7)$
- c) $f(x_3, x_2, x_1, x_0) = \text{Conjunto-UM}(0, 4, 8, 10)$
- d) $f(x_3, x_2, x_1, x_0) = \text{Conjunto-ZERO}(0, 8, 9, 12, 13)$
- e) $f(x_5, x_4, x_3, x_2, x_1, x_0) = \text{Conjunto-UM}(1, 8, 17, 22, 35, 53, 60)$
- f) $f(x_4, x_3, x_2, x_1, x_0) = \text{Conjunto-ZERO}(1, 5, 8, 19, 30)$

Problema 8.2 Encontre as expressões algébricas para os agrupamentos indicados em cada item.

- Agrupamentos de 1s mostrados na Fig. 8.10.
- Agrupamento de 0s mostrados na Fig. 8.11.
- Agrupamento de 1s mostrados na Fig. 8.12.
- Agrupamento de 0s mostrados na Fig. 8.13.

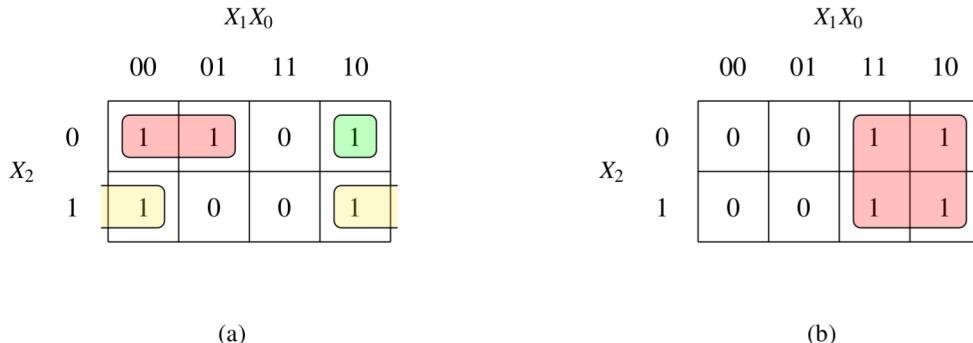


Figura 8.10: Agrupamento de 1 em mapas-K de funções de 3 variáveis para o exercício 8.2.

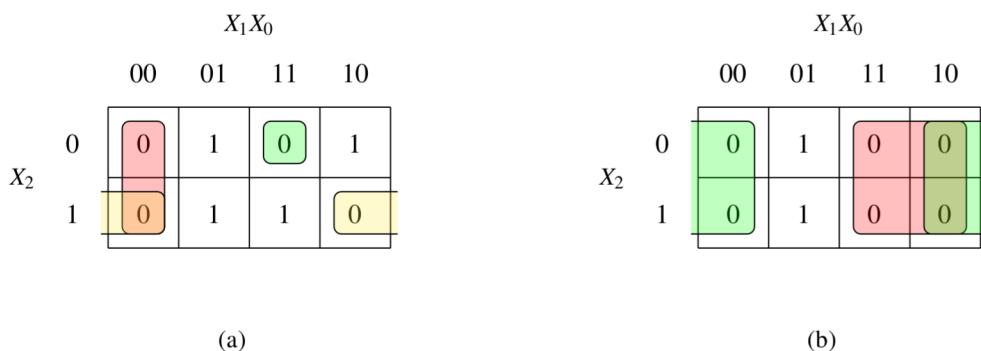


Figura 8.11: Agrupamento de 0 em mapas-K de funções de 3 variáveis para o exercício 8.2.

Problema 8.3 Nesse exercício o objetivo é encontrar agrupamentos de 0s ou 1s no mapa-K. Os grupos válidos de 0s ou 1s são agrupamentos de potências de 2, isto é, grupos válidos são agrupamentos de 1, 2, 4, 8, etc. Em cada item encontre os maiores agrupamentos possíveis conforme solicitado e escreva as respectiva expressão algébrica para cada agrupamento encontrado.

- Agrupamentos de 1s na Fig. 8.14a.
- Agrupamentos de 0s na Fig. 8.14b.
- Agrupamentos de 1s na Fig. 8.15a.
- Agrupamentos de 0s na Fig. 8.15b.
- Agrupamentos de 1s na Fig. 8.16.

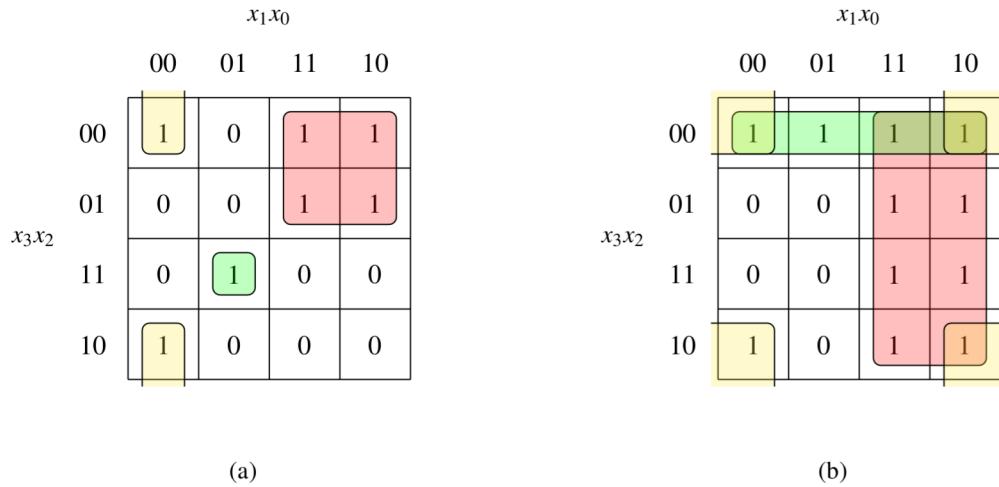


Figura 8.12: Agrupamento de 1 em mapas-K de funções de 4 variáveis para o exercício 8.2.

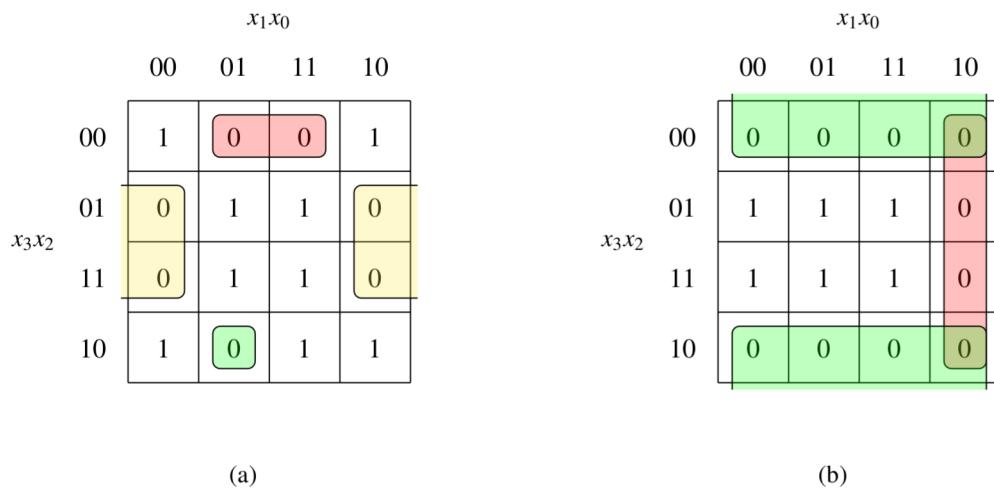


Figura 8.13: Agrupamento de 0 em mapas-K de funções de 4 variáveis para o exercício 8.2.

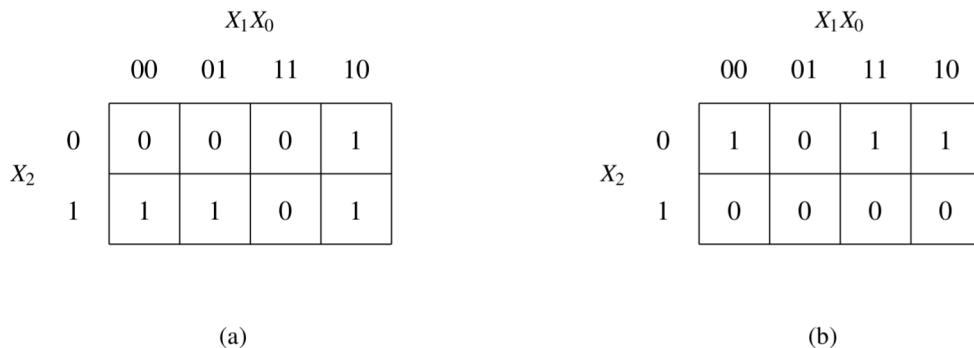


Figura 8.14: Mapas-K de funções de 3 variáveis para o exercício 8.3.

Figura 8.15: Mapas-K de funções de 4 variáveis para o exercício 8.3.

		x_1x_0						x_1x_0			
		00	01	11	10			00	01	11	10
x_3x_2	00	0	0	1	1			0	0	1	1
	01	0	0	1	1			0	0	1	1
	11	0	0	0	0			0	1	0	0
	10	1	1	1	1			0	0	1	0
		$x_4=0$						$x_4=1$			

Figura 8.16: Mapa-K de uma função de 5 variáveis para o exercício 8.3.