Minimização de funções booleanas

Última revisão em 15 de março de 2016

Este documento é parte das notas de aula da disciplina MAC0329 (Álgebra Booleana e Circuitos Digitais). Nesta parte abordaremos a manipulação de expressões na forma soma de produtos (SOP) e na forma produto de somas (POS), principalmente a simplificação delas.

Já vimos que uma função lógica pode ser descrita por meio de uma tabela-verdade (apesar de essa representação só ser praticável manualmente quando o número de entradas é pequeno ...) e, reciprocamente, que uma tabela-verdade define uma função.

1 Algumas definições

Antes de mais nada, vamos recordar as operações lógicas básicas E, OU e NÃO. Elas são definidas conforme a tabela 1.

		Função lógica		
		E	OU	NÃO
x_1	x_2	$x_1\, x_2$	$x_1 + x_2$	$\overline{x}_1$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Tabela 1: Tabela-verdade das operações lógicas E, OU e NÃO.

Algumas outras definições serão úteis ao longo do texto e são apresentadas aqui.

- iremos denominar por **variável booleana** qualquer variável que toma valores em $B = \{0, 1\}$;
- o **complemento** de uma variável booleana x, denotado $\overline{x}$, é uma variável booleana tal que $\overline{x} = \overline{a}$ sempre que $x = a$ para qualquer $a \in B$;
- um **literal** é uma variável booleana x ou o seu complemento $\overline{x}$.

2 Expressões na forma SOP e POS

Dada a definição de uma função por meio de sua tabela-verdade, podemos escrever a função por meio de uma expressão na forma soma de produtos (SOP). Para tanto, devemos considerar as entradas (atribuição de valores às variáveis da função) para as quais a função toma valor 1. Similarmente, podemos escrever a função por meio de uma expressão na forma produto de somas (POS), conforme mostrado no próximo exemplo.

Exemplo: Seja a tabela-verdade (**OBS.:** Lembrem-se de sempre escrever as entradas da tabela verdade em ordem lexicográfica) :

Entrada			Saída
a	b	c	$f(a, b, c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A função definida na tabela-verdade acima toma valor 1 para as entradas 010, 100, 101, 110 e 111. O produto $\overline{a}\,b\,\overline{c}$ toma valor 1 quando a entrada é 010 (pois, $\overline{a}\,b\,\overline{c} = \overline{0}\,1\,\overline{0} = 1 \cdot 1 \cdot 1 = 1$). Para qualquer outra entrada esse produto toma valor 0. Assim, podemos associar um único produto para cada entrada. Portanto, uma função pode ser escrita como a soma dos produtos associados às entradas para as quais ela toma valor 1. A função da tabela acima pode ser escrita como:

$$f(a, b, c) = \overline{a}\,b\,\overline{c} + a\,\overline{b}\,\overline{c} + a\,\overline{b}\,c + a\,b\,\overline{c} + a\,b\,c \tag{1}$$

Trata-se de uma expressão na forma soma de produtos (SOP). Os produtos na equação acima são canônicos pois envolvem todas as três variáveis, sendo cada uma delas ou na forma não barrada ou na forma barrada.

De forma dual, podemos também escrever uma função na forma produto de somas (POS). Observe que a função da tabela toma valor zero para as entradas 000, 001 e 011. A soma $a + b + c$ toma valor 0 para a entrada 000 e valor 1 para as demais entradas.

Assim, similarmente as caso dos produtos, podemos associar uma soma para cada entrada. Portanto, uma função pode também ser escrita como o produto de somas associadas às entradas para as quais ela toma valor 0. No caso da função f acima, temos que:

$$f(a,b,c) = (a+b+c)\,(a+b+\overline{c})\,(a+\overline{b}+\overline{c}) \tag{2}$$

Trata-se de uma expressão na forma produto de somas (POS), com somas canônicas.

Como o conjunto de todas as sequências de n bits corresponde à representação binária dos números entre 0 e 2^n-1, o produto canônico (também chamado de **mintermo**) associado a uma entrada de n bits pode ser caracterizado por um índice decimal. A tabela 2 apresenta todos os mintermos em três variáveis e a notação com índice decimal associada a cada um deles.

Entrada binária	mintermo	notação
0 0 0	$\overline{x}_1\overline{x}_2\overline{x}_3$	m_0
0 0 1	$\overline{x}_1\overline{x}_2 x_3$	m_1
0 1 0	$\overline{x}_1 x_2\overline{x}_3$	m_2
0 1 1	$\overline{x}_1 x_2 x_3$	m_3
1 0 0	$x_1\overline{x}_2\overline{x}_3$	m_4
1 0 1	$x_1\overline{x}_2 x_3$	m_5
1 1 0	$x_1 x_2\overline{x}_3$	m_6
1 1 1	$x_1 x_2 x_3$	m_7

Tabela 2: Tabela de mintermos em 3 variáveis.

Usando-se o índice decimal associado a cada produto canônico, podemos escrever a expressão de forma mais compacta como:

$$f(a,b,c) = m_2 + m_4 + m_5 + m_6 + m_7 = \sum m(2,4,5,6,7)$$

Maxtermo (ou **soma canônica**) em n variáveis $x_1, x_2, \ldots, x_n$ tem definição similar ao mintermo: em vez de produto, consiste de soma de n literais, cada um correspondendo a uma variável. As expressões $\overline{x}_1 + \overline{x}_2 + \overline{x}_3$ e $\overline{x}_1 + x_2 + x_3$ são exemplos de maxtermos em três variáveis. A tabela 3 lista todos os maxtermos de 3 variáveis.

Na forma produto de somas compacta temos:

$$f(a,b,c) = M_0 \cdot M_1 \cdot M_3 = \prod M(0,1,3)$$

Entrada binária	maxtermo	notação
0 0 0	$x_1 + x_2 + x_3$	M_0
0 0 1	$x_1 + x_2 + \overline{x}_3$	M_1
0 1 0	$x_1 + \overline{x}_2 + x_3$	M_2
0 1 1	$x_1 + \overline{x}_2 + \overline{x}_3$	M_3
1 0 0	$\overline{x}_1 + x_2 + x_3$	M_4
1 0 1	$\overline{x}_1 + x_2 + \overline{x}_3$	M_5
1 1 0	$\overline{x}_1 + \overline{x}_2 + x_3$	M_6
1 1 1	$\overline{x}_1 + \overline{x}_2 + \overline{x}_3$	M_7

Tabela 3: Tabela de maxtermos com 3 variáveis.

3 Minimização de funções booleanas

Vimos que uma mesma função pode ser expressa por mais de uma expressão. Dependendo do contexto no qual essas expressões são utilizadas, pode-se desejar encontrar, dentre todas as expressões que representam uma mesma função booleana, aquela que satisfaz algum critério. Por exemplo, pode ser do interesse obter uma expressão "mais curta", ou então, uma expressão que não envolve um termo produto com mais de um determinado número de literais.

Uma das simplificações bastante estudadas no contexto de circuitos digitais é a minimização lógica dois-níveis, isto é, a expressão de uma função na forma soma de produtos, envolvendo o menor número possível de termos produto e, em cada produto, o menor número possível de literais. O número de produtos e o número de literais em cada produto definem, respectivamente, o número de portas lógicas E e o número de entradas na correspondente porta E, em uma implementação direta da expressão.

Algoritmos de minimização lógica dois-níveis partem da expressão na forma SOP.

Exemplo: Sejam três variáveis a, b e c. Dados os produtos abc e $\overline{a}bc$, seja a disjunção (soma) $abc + \overline{a}bc$. Observe que podemos usar a mesma regra da álgebra elementar e colocar bc em evidência: $abc + \overline{a}bc = (a + \overline{a})bc = bc$ ($a + \overline{a} = 1$ sempre e, portanto, $(a + \overline{a})bc = 1 \cdot b\,c = b\,c$). O termo resultante bc é também um produto, porém sem a variável a. Note que o produto bc toma valor 1 para as entradas 011 e 111; o valor da variável a não afeta o valor desse produto. Veja uma compilação na tabela a seguir:

Expressão	Entradas para as quais a expressão toma valor 1
$\overline{a}bc$	$\{011\}$
abc	$\{111\}$
$\overline{a}bc + abc = bc$	$\{011,\ 111\}$

Em termos de circuitos, a simplificação acima significa que duas portas lógicas E de três entradas cada podem ser substituídas por uma porta E de duas entradas! Além disso, não será mais necessária uma porta OU.

Um produto pode ser associado a um intervalo. No caso do produto bc, as entradas para as quais o produto toma valor 1 é dado pelo conjunto $\{011,\ 111\}$. Esse conjunto tem um extremo inferior, 011, e um extremo superior, 111, definindo o intervalo $[011, 111]$. Um intervalo pode ser representado compactamente trocando-se as coordenadas não fixas por X. Assim $X11$ corresponde ao intervalo $[011, 111]$. Se o produto considerado fosse a, teríamos que ele toma valor 1 para as entradas $\{100, 101, 110, 111\}$ que pode ser entendido como o intervalo $[100, 111]$. Em notação compacta, esse intervalo pode ser denotado por $1XX$. Similarmente, um intervalo pode ser associado a um produto. No caso de três variáveis a, b e c, o intervalo $[000, 100] = \{000, 100\}$ (ou simplemente $X00$) correponde ao produto $\overline{b}\,\overline{c}$. O intervalo $X0X = [000, 101] = \{000, 001, 100, 101\}$ corresponde ao produto $\overline{b}$. Um intervalo contém necessariamente 2^k elementos, onde $0 \leq k \leq n$.

Esses conceitos/terminologias estão resumidos no quadro a seguir:

Produto	elementos cobertos	intervalo	notação compacta (cubo/intervalo)	dimensão (k)	tamanho (2^k)
ab	$\{110, 111\}$	$[110, 111]$	$11X$	1	$2^1 = 2$
c	$\{001, 011, 101, 111\}$	$[001, 111]$	$XX1$	2	$2^2 = 4$

Observação: Daqui em diante utilizaremos equivalentemente os termos **produto**, **cubo** ou **intervalo** quando nos referirmos a um produto.

3.1 Simplificação algébrica

Usando regras algébricas, como por exemplo em $abc + \overline{a}bc = (a + \overline{a})bc = bc$, podemos realizar a simplificação de uma expressão.

Retomando a expressão do início, podemos simplificar como segue:

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

$$\begin{aligned} &= b\,\overline{c} + a \\ &= a + b\,\overline{c} \end{aligned}$$

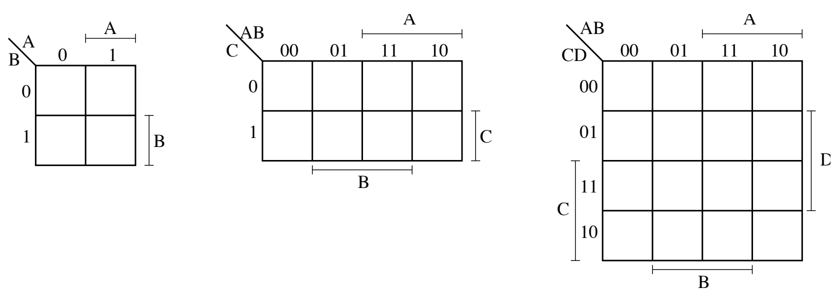
A simplificação algébrica, porém, pode ser complicada pois requer o conhecimento das regras e, além disso, dependendo da ordem na qual as regras são aplicadas pode ser necessário um grande número de passos para se chegar à simplificação (sem contar que às vezes podemos, depois de vários passos, voltar à situação inicial). Se a expressão envolver muitas variáveis, fazer a simplificação na mão traz adicionalmente uma grande chance de se cometer erros bobos (trocar o nome de uma variável sem querer, esquecer o barra sobre uma variável de um passo para o outro, etc).

Iremos ver as regras algébricas e exercitar a simplificação e manipulação algébrica de expressões oportunamente, mas não neste momento.

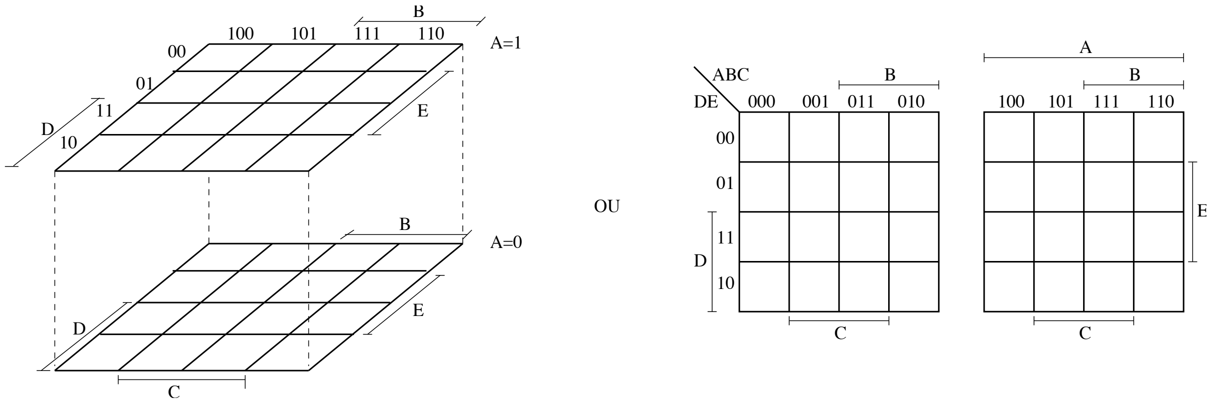
3.2 Mapas de Karnaugh

Mapas de Karnaugh são diagramas que são utilizados para auxiliar o processo de minimização lógica dois-níveis, quando o número de variáveis não passa de 6. Embora não seja usado na prática, como ferramenta pedagógica é interessante e será explorado a seguir.

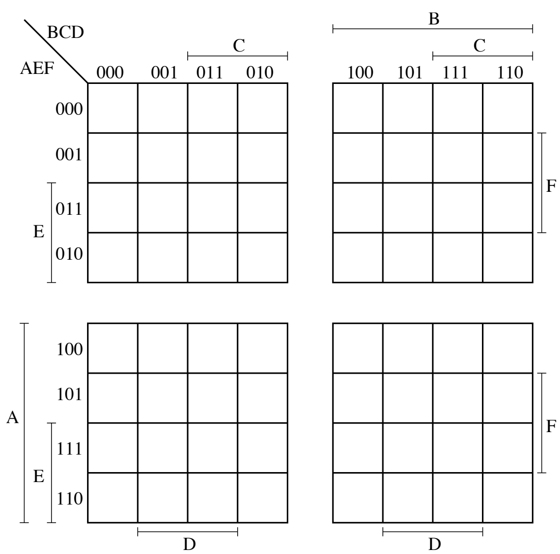
Mapa de Karnaugh de 2, 3 e 4 variáveis:



Mapa de Karnaugh de 5 variáveis:



Mapa de Karnaugh de 6 variáveis:



Cada célula dos mapas corresponde a um elemento de $\{0,1\}^n$ (no caso $n = 2, 3, 4, 5$ ou 6). A concatenação do cabeçalho da coluna com o cabeçalho da linha de uma célula dá o elemento correspondente àquela célula. No caso de 3 variáveis, o mapa à esquerda na figura 1 mostra em cada célula a entrada correspondente, enquanto o mapa à direita mostra em cada célula o valor decimal das respectivas entradas. Observe que o cabeçalho está disposto em uma sequência não-usual. Por exemplo, para duas variáveis, a sequência natural seria 00, 01, 10, 11. Porém, a sequência utilizada é 00, 01, 11, 10, que possui a característica de dois elementos adjacentes (na sequência) diferirem em apenas 1 bit (essa propriedade vale se juntarmos os extremos esquerdo e direito do mapa e, similarmente, os extremos superior e inferior).

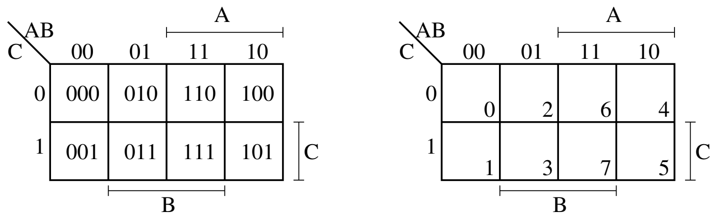


Figura 1: As entradas correspondentes a cada célula do mapa de Karnaugh de 3 variáveis em notação binária e decimal, respectivamente.

Vejamos, por meio de um exemplo, como pode ser realizada a minimização utilizando o mapa de Karnaugh. Seja $f(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 + \overline{x}_1\,\overline{x}_2\,x_3 + \overline{x}_1\,x_2\,\overline{x}_3 + \overline{x}_1\,x_2\,x_3 + x_1\,x_2\,x_3$. Algebricamente, podemos proceder como segue:

$$
\begin{aligned}
f(x_1, x_2, x_3) &= \overline{x}_1\,\overline{x}_2\,\overline{x}_3 + \overline{x}_1\,\overline{x}_2\,x_3 + \overline{x}_1\,x_2\,\overline{x}_3 + \overline{x}_1\,x_2\,x_3 + x_1\,x_2\,x_3 \\
&= \overline{x}_1\,\overline{x}_2\,(\overline{x}_3 + x_3) + \overline{x}_1\,x_2\,(\overline{x}_3 + x_3) + x_1\,x_2\,x_3 \\
&= \overline{x}_1\,\overline{x}_2 + \overline{x}_1\,x_2 + x_1\,x_2\,x_3 \\
&= \overline{x}_1\,(\overline{x}_2 + x_2) + x_1\,x_2\,x_3 \\
&= \overline{x}_1 + x_1\,x_2\,x_3 \\
&= \overline{x}_1 + x_2\,x_3
\end{aligned}
$$

Aparentemente a expressão acima é minimal. Para utilizarmos o mapa de Karnaugh, precisamos primeiramente transformar os mintermos da função para a notação cúbica. Assim,

Mintermo	notação cúbica
$\overline{x}_1\,\overline{x}_2\,\overline{x}_3$	000
$\overline{x}_1\,\overline{x}_2\,x_3$	001
$\overline{x}_1\,x_2\,\overline{x}_3$	010
$\overline{x}_1\,x_2\,x_3$	011
$x_1\,x_2\,x_3$	111

Em seguida, as células correspondentes a esses mintermos devem ser marcados com 1 no mapa, conforme mostrado no mapa da esquerda na figura 2.

O processo consiste, então, em procurar, para cada 1 no mapa, o maior agrupamento retangular de 1's que inclui o primeiro 1, e que tenha 2^k elementos ($k >= 1$), chamados cubos maximais. No exemplo da figura 2, o maior agrupamento que cobre 000 é o cubo

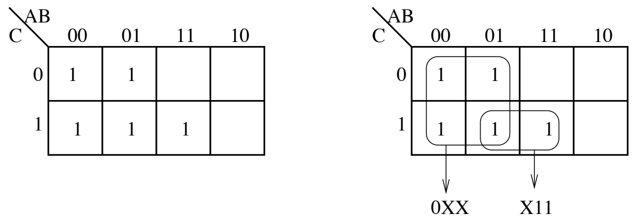


Figura 2: Exemplo do uso do mapa de Karnaugh: minimização da função $f = \sum m(0, 1, 2, 3, 7)$.

0XX. Este cubo não cobre o elemento 111. Assim, tomamos também o maior cubo que cobre 111, que no caso é o cubo X11. Depois desse procedimento, todos os mintermos da função encontram-se cobertos por algum cubo. Assim, podemos dizer que uma solução SOP minimal corresponde aos cubos 0XX e X11. O produto correspondente ao cubo 0XX é $\overline{x}_1$ e o correspondente a X11 é $x_2\, x_3$. Portanto, uma forma SOP minimal é $f(x_1, x_2, x_3) = \overline{x}_1 + x_2\, x_3$.

Exemplo: Minimize a função $f(a, b, c, d) = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$. A resposta é $f(a, b, c, d) = c + \overline{a}\, b\, d + \overline{b}\, \overline{d}$. Veja o mapa da figura 3.

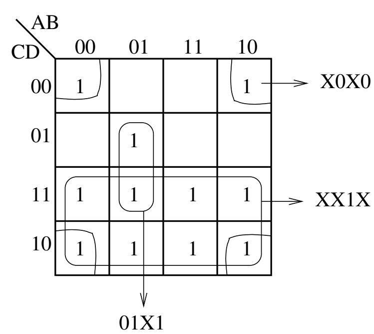


Figura 3: Minimização da função $f(a, b, c) = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$.

Exemplo: Minimize a função $f(a, b, c, d) = \sum m(0, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15)$. Neste caso, há mais de uma solução. A figura 4 mostra todos os cubos maximais de f. As possíveis soluções são:

$$f(a, b, c, d) = \overline{a}\, b + a\, \overline{b} + \overline{a}\, \overline{c}\, \overline{d} + b\, c \qquad f(a, b, c, d) = \overline{a}\, b + a\, \overline{b} + \overline{b}\, \overline{c}\, \overline{d} + b\, c$$
$$f(a, b, c, d) = \overline{a}\, b + a\, \overline{b} + \overline{a}\, \overline{c}\, \overline{d} + a\, c \qquad f(a, b, c, d) = \overline{a}\, b + a\, \overline{b} + \overline{b}\, \overline{c}\, \overline{d} + a\, c$$

Este exemplo mostra que a solução não é única (ou seja, podem existir mais de uma forma SOP minimal de mesmo custo) e que nem todos os implicantes primos fazem parte da forma SOP minimal.

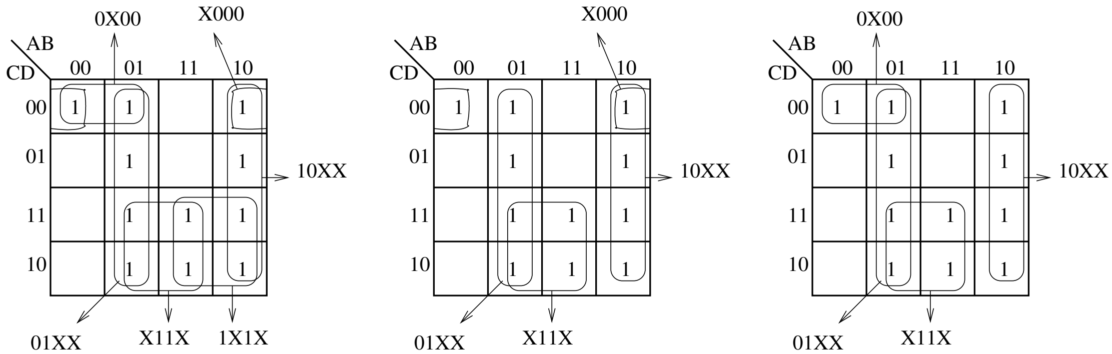


Figura 4: Minimização da função $f(a,b,c) = \sum m(0,4,5,6,7,8,9,10,11,14,15)$. Esquerda: todos os implicantes primos (ou cubos maximais). Centro: uma solução. Direita: outra solução.

Mapa de Karnaugh para encontrar a forma POS minimal: o mapa de Karnaugh pode ser utilizado também para encontrar a forma POS (produto de somas) minimal de uma função booleana? A resposta é sim. Considere a função $f(a,b,c) = \sum m(0,4,5,7)$. A minimização SOP de f por mapa de Karnaugh é mostrada na figura 5. A forma SOP minimal é $f(a,b,c) = \overline{b}\,\overline{c} + a\,c$.

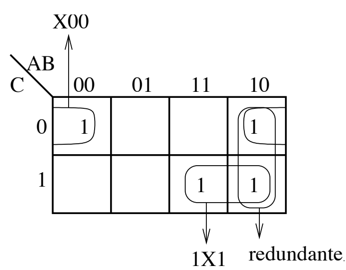


Figura 5: Exemplo do uso do mapa de Karnaugh: minimização da função $f(a,b,c) = \sum m(0,4,5,7)$.

Usando o fato de que $f = \overline{\overline{f}}$, podemos calcular inicialmente a minimização SOP de $\overline{f}(a,b,c) = \sum m(1,2,3,6)$ por mapa de Karnaugh (figura 6). O resultado obtido é $\overline{f}(a,b,c) = b\,\overline{c} + \overline{a}\,c$.

Em seguida, temos $f = \overline{b\,\overline{c} + \overline{a}\,c} = (\overline{b\,\overline{c}})(\overline{\overline{a}\,c}) = (\overline{b} + c)(a + \overline{c})$.

Tudo isto pode ser diretamente realizado no mapa de Karnaugh conforme mostrado na figura 7. Em vez de marcar os 0-cubos da função no mapa, marcamos os 0-cubos do complemento de f (ou, equivalentemente, os zeros da função). Aplica-se o processo de encontrar os cubos maximais. Para escrever a função na forma POS minimal, basta escrevermos o termo soma correspondente a cada cubo. No exemplo, o cubo 0X1 corresponde ao termo soma $a + \overline{c}$ e o cubo X10 ao termo soma $\overline{b} + c$. Assim, temos que a forma POS

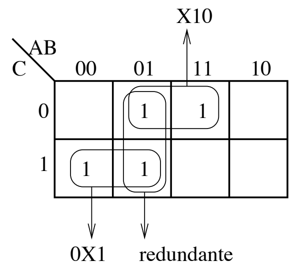


Figura 6: Minimização da função $\overline{f}(a, b, c) = \sum m(1, 2, 3, 6)$.

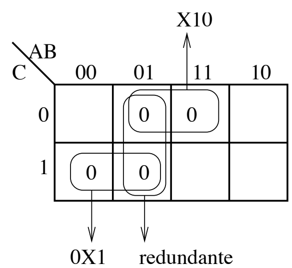


Figura 7: Minimização POS da função $f(a, b, c) = \prod M(1, 2, 3, 6)$.

minimal é $f(a, b, c) = (a + \overline{c})(\bar{b} + c)$.

3.3 Minimização na presença de don't cares

Em algumas situações, o valor de uma função para algumas entradas não são relevantes (tipicamente porque tais entradas nunca ocorrerão na prática). Em tais situações, tanto faz se a função toma valor 0 ou 1 nessas entradas, que serão denominadas de **don't cares**.

Para minimizar uma função incompletamente especificada, os don't cares podem ser utilizados para formar cubos maximais junto com os 1's da função. Don't cares agrupados serão mapeados para 1 pela função simplificada resultante, enquanto os não utilizados serão mapeados para zero.

3.4 Minimização de múltiplas funções

Exemplo 1: Considere as funções f_1 e f_2 dadas pelas seguintes tabelas:

$x_1\, x_2\, x_3$	$f_1(x_1, x_2, x_3)$
000	1
001	1
010	0
011	0
100	0
101	1
110	0
111	0

$x_1\, x_2\, x_3$	$f_2(x_1, x_2, x_3)$
000	0
001	0
010	0
011	0
100	0
101	1
110	1
111	1

A minimização individual destas duas funções resulta em $f_1(x_1, x_2, x_3) = \overline{x}_1\, \overline{x}_2 + \overline{x}_2\, x_3$ e $f_2(x_1, x_2, x_3) = x_1\, x_3 + x_1\, x_2$. Para implementá-las, são necessárias 4 portas E.

Note, porém, que podemos escrever $f_1(x_1\, x_2\, x_3) = \overline{x}_1\, \overline{x}_2 + \overline{x}_2\, x_3 = \overline{x}_1\, \overline{x}_2 + x_1\, \overline{x}_2\, x_3$ e $f_2(x_1\, x_2\, x_3) = x_1\, x_3 + x_1\, x_2 = x_1\, x_2 + x_1\, \overline{x}_2\, x_3$. Neste caso, há um produto comum às duas funções e portanto para implementá-las são necessárias 3 portas E, pois uma das portas E é compartilhada pelas duas funcões.

Exemplo 2: Considere as funções (já minimizadas individualmente) :

$$f_0 = a$$

$$f_1 = \overline{b}$$

$$f_2 = bc + ab$$

$$f_3 = \overline{a}\, \overline{b} + \overline{a}\, c$$

Se expressas desta forma, para implementá-las, são necessárias 6 portas E. No entanto, note que elas podem ser reescritas como:

$$f_0 = a\overline{b} + ab$$

$$f_1 = \overline{a}\overline{b} + a\overline{b}$$

$$f_2 = \overline{a}bc + ab$$

$$f_3 = \overline{a}\, \overline{b} + \overline{a}\, b\, c$$

e neste caso são necessárias apenas 4 portas E.

Estes exemplos mostram que minimizar individualmente as funções não necessariamente representa a melhor solução para a minimização conjunta. Observe também que, no segundo exemplo, na minimização conjunta reduzimos o número total de produtos, mas o número de somas aumentou. Isto é uma desvantagem? Se o objetivo é a realização em um PLA (*Programmable logic array*) não faz diferença se a função é formada, por exemplo, por 2 ou 3 termos produto. Porém, se o objetivo for minimizar também o número de entradas das portas, então é preciso tomar cuidado para não aumentar demasidamente o número de termos (pois isso implica aumento no número de entradas da porta OU).

3.5 PLA

A minimização lógica dois-níveis ganhou impulso na década de 1980 devido aos dispositivos conhecidos como **PLA** (*Programmable Logic Arrays*). Eles consistem de um conjunto de entradas, com uma malha programável de conexões para um conjunto de portas E, e uma malha programável de conexões entre as saídas das portas E para um conjunto de portas OU. Por malha programável entende-se que os cruzamentos podem ser conectados (programados) para conduzir o sinal. No estado inicial, nenhum cruzamento está conectado nas malhas de um PLA.

A figura 8 mostra um modelo lógico básico de um PLA típico, com 3 variáveis de entrada e três saídas. Os círculos (pontos pretos) sobre o cruzamento das linhas indicam onde há conexão. No exemplo, as portas lógicas E realizam, respectivamente de cima para baixo, as funções (produtos) $a\,\overline{b}$, $\overline{a}\,b\,\overline{c}$, $\overline{b}\,c$ e $a\,\overline{c}$; as portas lógicas OU realizam, respectivamente da esquerda para a direita, as funções $f_1(a,b,c) = \overline{a}\,b\,\overline{c} + a\,\overline{b}$, $f_2(a,b,c) = \overline{a}\,b\,\overline{c} + \overline{b}\,c$ e $f_3(a,b,c) = \overline{a}\,b\,\overline{c} + a\,\overline{c}$.

Para não sobrecarregar o diagrama, em geral desenha-se de forma simplificada como o mostrado na figura 9.

PLAs comerciais têm tipicamente[1] entre 10 e 20 entradas, entre 30 e 60 portas E (produtos) e entre 10 e 20 portas OU (saídas). Em um PLA com 16 entradas, 48 produtos e 8 saídas, existem $2 \times 16 \times 48 = 1536$ cruzamentos na malha E e $8 \times 48 = 384$ cruzamentos na malha OU. Um número considerável de funções relativamente complexas podem ser realizadas via um PLA. Claramente, quanto menor o número de variáveis e termos produtos utilizados na expressão de uma função, menor será o “tamanho” do PLA necessário para a realização da função.

Exercícios:

[1]Informação colhida em torno de 2010 ...

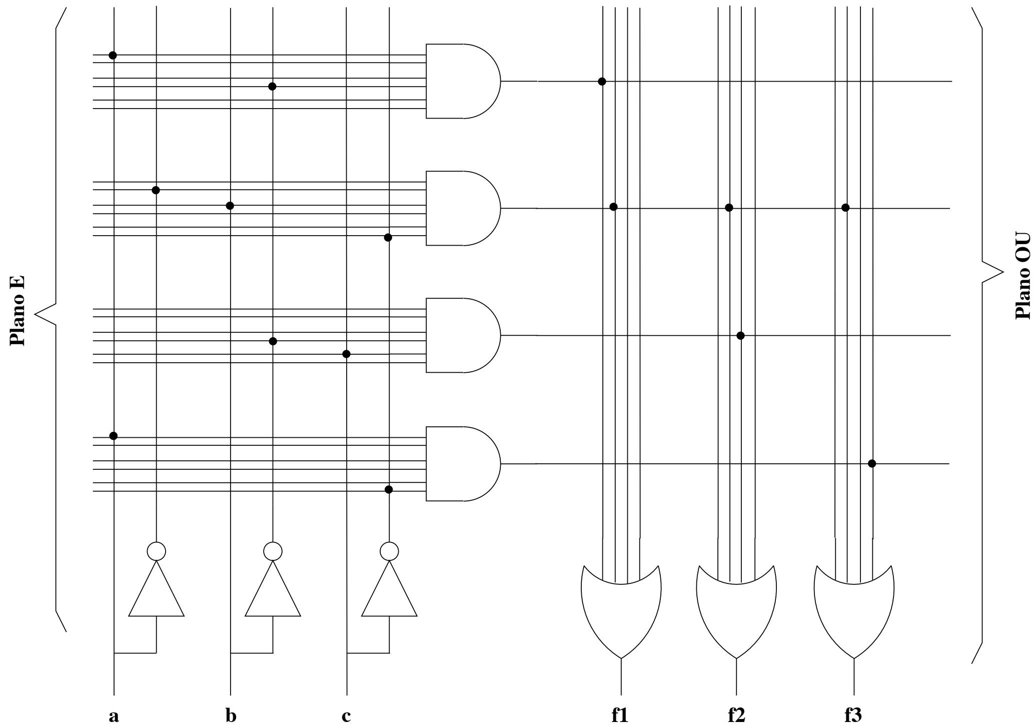


Figura 8: Esquema lógico de um PLA.

1. Minimize as funções dos exemplos 1 e 2 usando o mapa de Karnaugh e compare os resultados com os que foram apresentados acima.

2. Mostre como fica a realização das quatro funções do Exemplo 2, utilizando 4 portas E, em um PLA.

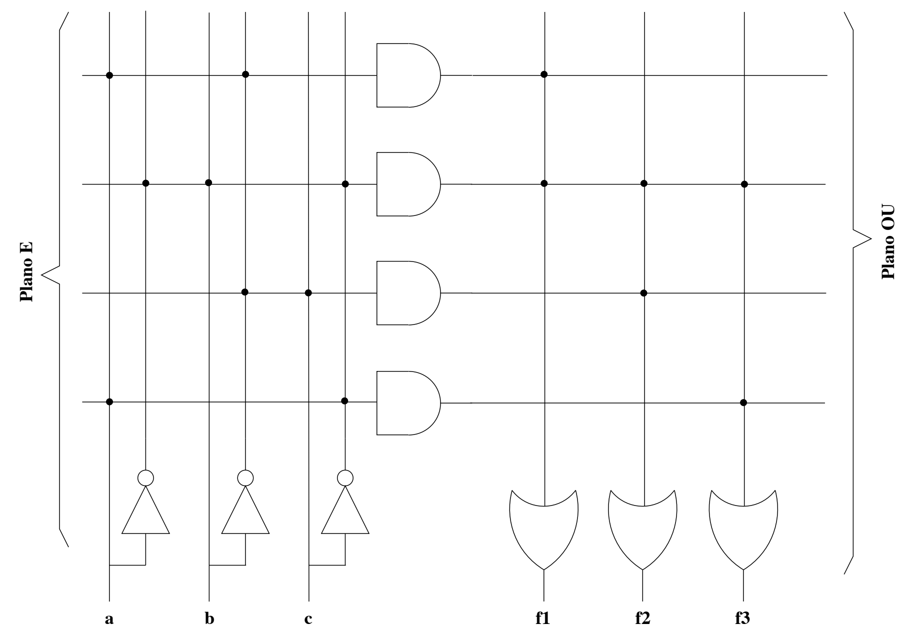


Figura 9: Esquema simplificado de um PLA.