Lógica combinacional dois-níveis

Computadores digitais processam dados em formato binário. Esses processamentos podem ser encarados como mapeamentos do tipo $f : \{0,1\}^n \to \{0,1\}^m$. Vimos que qualquer mapeamento $f : \{0,1\}^n \to \{0,1\}$ pode ser expresso como soma de produtos canônicos (soma de mintermos) ou como produto de somas canônicas (produto de maxtermos). As expressões do tipo soma e produto podem ser realizadas em circuito pelas portas OU e E, respectivamente. Então, em príncípio, qualquer mapeamento $f : \{0,1\}^n \to \{0,1\}^m$ pode ser realizado por circuitos que utilizam apenas os inversores e as portas E e OU.

Deste capítulo em diante trataremos apenas de funções booleanas do tipo $f : \{0,1\}^n \to \{0,1\}$. Essas funções são, por alguns autores, chamadas de **funções de chaveamento**.

1.1 Lógica combinacional e seqüencial

Circuitos lógicos são classificados em dois tipos: combinacionais e seqüenciais. Os **circuitos combinacionais** são aqueles nos quais as saídas são determinadas em função apenas das entradas atuais. Os **circuitos seqüenciais** são aqueles nos quais as saídas dependem não apenas das entradas atuais mas também de dados prévios nos instantes anteriores. Pode-se dizer que circuitos seqüenciais envolvem realimentação, ou seja, eles possuem "memória".

O número de níveis de um circuito é definido como o número máximo de portas lógicas que um sinal de entrada deve atravessar para chegar até a saída. No caso de expressões do tipo soma de produtos, uma realização direta em circuito consiste de um conjunto de portas E (que são alimentados pelos sinais de entrada, invertidos ou não) no primeiro nível e, no segundo nível, uma porta OU (que recebe como entradas as saídas das portas E do primeiro nível). A saída da porta OU no segundo nível é o valor da função. Assim, um circuito desses é um circuito dois-níveis.

Lógica combinacional dois-níveis relaciona-se com o estudo de expressões que culminem em uma realização por circuito combinacional dois-níveis. Em particular, um problema muito estudado no contexto de circuitos lógicos é o problema de minimização lógica dois níveis, ou seja, o de encontrar uma menor expressão na forma soma de produtos que seja equivalente a uma função $f : \{0,1\}^n \to \{0,1\}$. Do ponto de vista de circuito, o número de níveis igual a dois implica que o circuito é eficiente em termos de tempo de processamento e a minimização do número de produtos na expressão implica minimização do número de portas lógicas (e, portanto, do tamanho do circuito e seu custo).

1.2 Minimização lógica dois-níveis

Ao falarmos em minimização lógica dois-níveis, estamos pensando em expressões na forma soma de produtos. Inicialmente definiremos o que é uma expressão minimal.

Definição: Uma expressão booleana escrita como soma de produtos é **minimal** se (1) não existe nenhuma outra expressão equivalente na forma soma de produtos com um número menor de termos e (2) não existe nenhuma outra expressão equivalente na forma soma de produtos com igual número de termos mas com menor número de literais.

Dada uma expressão minimal na forma soma de produtos, ao se remover um produto ou um literal de qualquer um dos produtos, a expressão resultante não mais representa a mesma função.

Dada uma função qualquer, como pode ser calculada uma expressão minimal dessa função na forma soma de produtos? Antes de prosseguirmos em direção à resposta, introduzimos alguns termos e conceitos.

1.2.1 Produtos, cubos e intervalos

Produtos como expressão:

Já vimos que produto é uma expressão booleana que consiste de conjunção de literais que não envolvem uma mesma variável. Em particular, o produto canônico (ou mintermo) é um produto em que cada variável ocorre uma vez, ou na forma barrada ou na forma não-barrada. Vimos também que um produto canônico em n variáveis toma valor 1 em apenas um elemento do conjunto $\{0,1\}^n$.

Considere três variáveis a, b e c e os mintermos abc e $\overline{a}bc$. Sabemos que $abc + \overline{a}bc = (a + \overline{a})bc = bc$. O termo bc é também um produto, porém ele não é canônico. O produto bc toma valor 1 para os elementos 011 e 111 de $\{0,1\}^3$. Em outras palavras, o valor da variável a não afeta o valor desse produto.

Produtos como subconjuntos de $\{0,1\}^n$ ou sub-cubos:

Lembramos que na função $f(a,b,c) = abc + \overline{a}bc$, os mintermos na notação compacta são m_7 (pois $111_{(2)} = 7_{(10)}$) e m_3 (pois $011_{(2)} = 3_{(10)}$). Assim, é usual escrevermos $f(a,b,c) = \sum m(3,7)$. Uma vez que existe uma correspondência um-para-um entre os mintermos em n variáveis e os elementos de $\{0,1\}^n$, uma função expressa como soma de mintermos pode ser vista como um subconjunto de $\{0,1\}^n$.

Soma de mintermos	subconjunto de $\{0,1\}^3$
$\overline{a}bc$	$\{011\}$
abc	$\{111\}$
$\overline{a}bc + abc$	$\{011, 111\}$

Os elementos de $\{0,1\}^n$ (aqui denotados como seqüências ou strings de n números binários) podem ser encarados como pontos no n-espaço. A coleção dos 2^n elementos de $\{0,1\}^n$ forma os vértices de um hipercubo. A figura 1.1 mostra um hipercubo no espaço de dimensão 3.

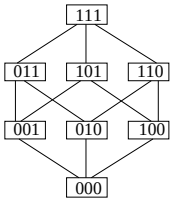


Figura 1.1: Um hipercubo de dimensão 3.

No contexto de circuitos lógicos, hipercubos são denominados n-**cubos**. Os vértices de um n-cubo são denominados 0-cubos. Dois 0-cubos formam um 1-cubo se eles diferem em apenas uma coordenada. Quatro 0-cubos formam um 2-cubo se eles são iguais a menos de duas coordenadas. De modo geral, 2^k 0-cubos formam um k-cubo se eles são exatamente iguais a menos de k coordenadas.

Produtos como intervalos do poset $(\{0,1\}^n, \leq)$:

Observe que cubos não são subconjuntos arbitrários de $\{0,1\}^n$. Um cubo é um conjunto de elementos em $\{0,1\}^n$ para os quais um produto toma valor 1, i.e., se p é um produto então o cubo correspondente a p é o conjunto $p\langle 1\rangle = \{\mathbf{b} \in \{0,1\}^n : p(\mathbf{b}) = 1\}$.

Cubos, no contexto de reticulados, são sub-reticulados do reticulado $\{0,1\}^n$, denominados **intervalos**. Um intervalo num poset é caracterizado por dois extremos: o menor e o maior elementos contidos nele. Assim, no poset $(\{0,1\}^3, \leq)$, o intervalo de extremo inferior 100 e extremo superior 101 é denotado $[100, 101]$ e definido por $[100, 101] = \{\mathbf{x} \in \{0,1\}^3 : 100 \leq \mathbf{x} \leq 101\}$.

Denotamos um k-cubo ou intervalo de dimensão k colocando um X nas coordenadas que não são iguais. Assim, no caso de três variáveis a, b e c, o 1-cubo $\{000, 100\}$ (ou, equivalentemente, o intervalo $[000, 100]$), que correponde ao produto $\overline{b}\,\overline{c}$, é representado por $X00$. O 2-cubo $\{000, 001, 100, 101\}$ (ou, equivalentemente, o intervalo $[000, 101]$), que corresponde ao produto $\overline{b}$, é representado por $X0X$. Um intervalo contém necessariamente 2^k elementos, onde $0 \leq k \leq n$. Quanto maior a dimensão de um cubo, menor o número de literais presentes no correspondente produto.

Exemplo: A figura 1.2 mostra alguns cubos. Dizemos que o 0-cubo 000 está *contido* no (ou é coberto pelo) 1-cubo $X00$, ou ainda, que o 1-cubo $X00$ *cobre* o 0-cubo 000. Analogamente, dizemos que o 1-cubo $X00$ está contido no 2-cubo $X0X$ ou que o 2-cubo $X0X$ cobre o cubo $X00$.

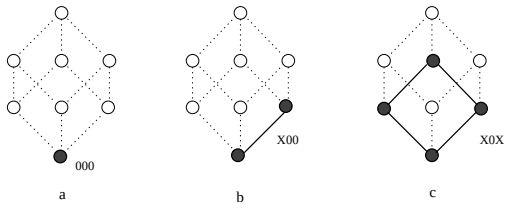


Figura 1.2: Os cubos 000, $X00$ e $X0X$.

Resumo:

Produtos, cubos e intervalos referem-se a mesma coisa. A tabela a seguir resume esses conceitos:

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	$2^2 = 4$

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

Uma função booleana com poucas variáveis (tipicamente 3 ou 4) pode ser graficamante ilustrada através do diagrama de Hasse. Usaremos a convenção de desenhar elementos de $f\langle 1\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 1\}$ como círculos preenchidos, enquanto os elementos de $f\langle 0\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 0\}$ serão representados por círculos não preenchidos. A representação via diagrama de Hasse da função $f_1(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 + \overline{x}_1\,\overline{x}_2\,x_3 + \overline{x}_1\,x_2\,x_3 + x_1\,x_2\,x_3$ é mostrada na figura 1.3.

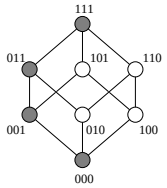


Figura 1.3: Representação via diagrama de Hasse da função $f_1(x_1, x_2, x_3) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 + \overline{x}_1\,\overline{x}_2\,x_3 + \overline{x}_1\,x_2\,x_3 + x_1\,x_2\,x_3$.

Sempre que um dos produtos de uma soma de produtos toma valor 1, a soma também toma valor 1. No caso da função $f(a, b, c) = abc + \overline{a}bc$, se $abc = 1$ então $f(a, b, c) = 1$. Isto pode ser expresso como $\overline{a}bc \leq f$. Analogamente temos $abc \leq f$ e $bc \leq f$.

Dada uma função f e um produto p, dizemos que o conjunto $p\langle 1\rangle$ **é um cubo de** f se $p \leq f$ (ou, equivalentemente, se $p\langle 1\rangle \subseteq f\langle 1\rangle$). Logo, $\overline{a}bc$ e abc, por exemplo, são cubos de $f(a, b, c) = abc + \overline{a}bc$.

Neste sentido, a forma SOP canônica de f pode ser vista como a coleção de todos os 0-cubos de f, aqueles que correspondem aos elementos em $f\langle 1\rangle$.

Os dois primeiros cubos da Fig. 1.4 (em negrito) são cubos da função f_1 mas os dois últimos não são.

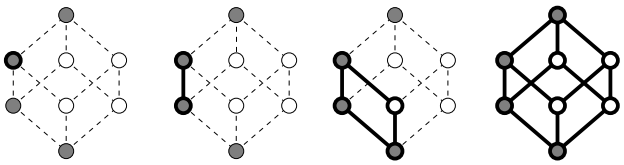


Figura 1.4: Exemplos de um 0-cubo (intervalo 011), um 1-cubo (intervalo $0X1$), um 2-cubo (intervalo $0XX$) e um 3-cubo (intervalo XXX), respectivamente (em negrito).

Definição: Um **implicante primo** (ou simplesmente **primo**) de uma função booleana f é um produto p tal que $p \leq f$, e não há outro produto p', $p < p'$, tal que $p' \leq f$.

Os implicantes primos são cubos ou intervalos maximais contidos em $f\langle 1\rangle$ (i.e., um cubo de f que não é totalmente contido em outro cubo de f). Por exemplo, na função $f(a,b,c) = abc + \overline{a}bc$, abc é um cubo de f, mas não é implicante primo de f pois $abc < bc \leq f$. Já bc é implicante primo de f.

Exemplo: A função Booleana $f = \sum m(0,1,4,5,6)$ é representada pelos vértices 000, 001, 100, 101 e 110. Os mintermos de f e os implicantes primos de f (cubos $X0X$ e $1X0$) são mostrados respectivamente nas figuras 1.5a e 1.5b.

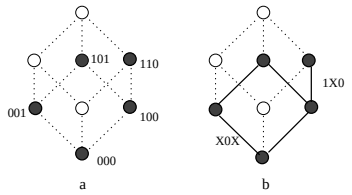


Figura 1.5: (a) Mintermos e (b) implicantes primos de $f = \sum m(0,1,4,5,6)$.

Voltemos agora à questão do início desta seção: dada uma função qualquer, qual é a expressão equivalente minimal na forma soma de produtos?

Teorema: Qualquer produto em uma expressão minimal na forma soma de produtos é um implicante primo.

A prova deste teorema é simples. Suponha que exista algum produto p na expressão que não seja um implicante primo. Por definição, existe um produto p' tal que $p < p'$ e tal que p' implica a função. Então, ao substituirmos p por p' na expressão, obtemos uma expressão equivalente, porém com custo menor. Isto contradiz com o fato de que a expressão era minimal.

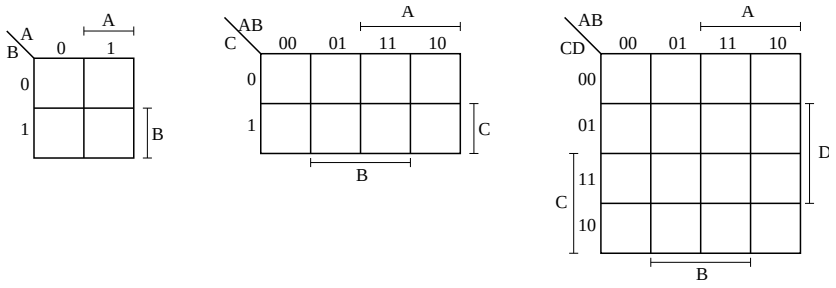
Este teorema diz, em outra palavras, que para encontrarmos uma expressão minimal de uma função, basta considerarmos apenas os implicantes primos da função.

Nas próximas seções serão apresentadas algumas técnicas utilizadas para o cálculo de uma expressão minimal na forma soma de produtos.

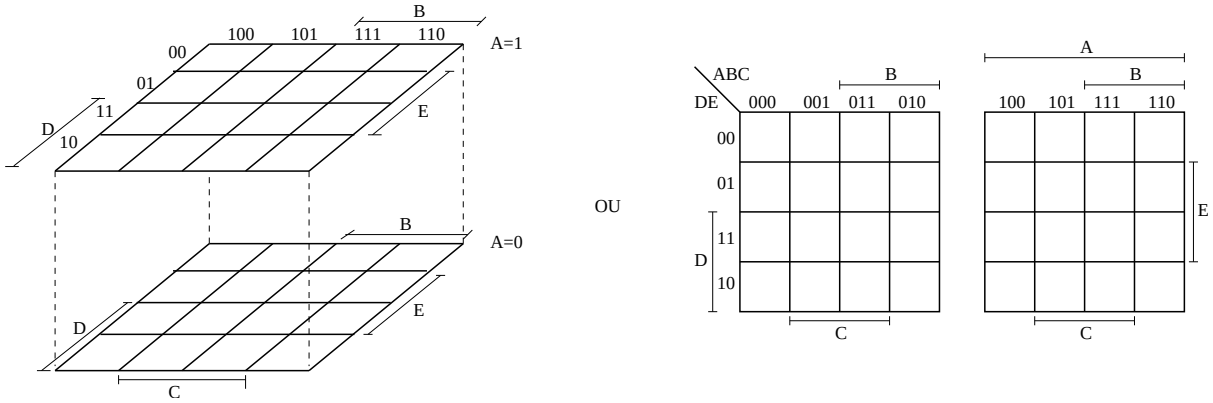
1.2.2 Mapas de Karnaugh

A minimização de uma expressão Booleana pode ser realizada algebricamente aplicando-se os axiomas e leis da álgebra Booleana. Entretanto, a manipulação algébrica, além de ser uma tarefa cansativa, pode facilmente induzir uma pessoa a cometer erros, principalmente quando o número de variáveis envolvidas é grande. Além disso, muitas vezes é difícil ter certeza de que o resultado obtido é minimal. Mapas de Karnaugh são diagramas que são utilizados para auxiliar este processo.

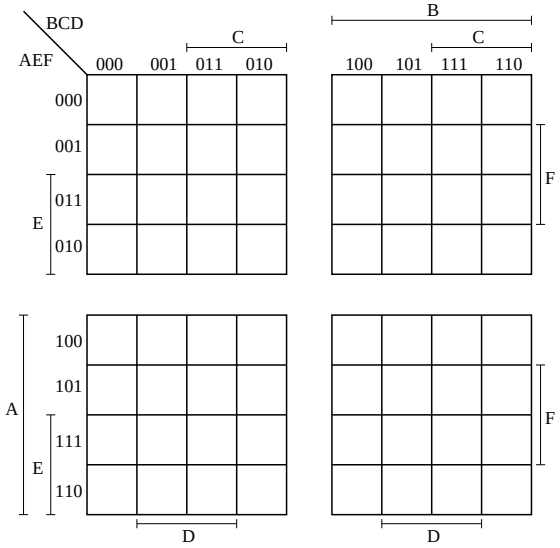
Mapa de Karnaugh de 2, 3 e 4 variáveis:



Mapa de Karnaugh de 5 variáveis:



Mapa de Karnaugh de 6 variáveis:



Cada célula dos mapas corresponde a um elemento de $\{0,1\}^n$. 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 1.6 mostra em cada célula o 0-cubo correspondente, enquanto o mapa à direita mostra em cada célula o valor decimal dos respectivos 0-cubos. Observe que o cabeçalho está disposto em uma seqüência não-usual. Por exemplo, para duas variáveis, a seqüência natural seria 00, 01, 10, 11. Porém, a seqüência utilizada é 00, 01, 11, 10, que possui a característica de dois elementos adjacentes (na seqüência) diferirem em apenas 1 bit.

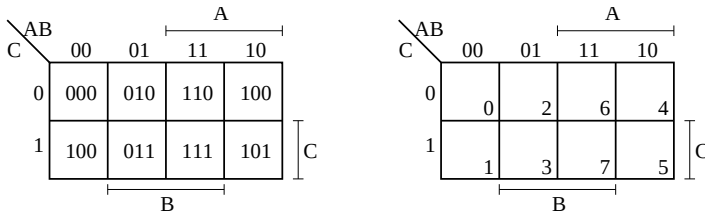


Figura 1.6: Os 0-cubos correspondentes a cada célula da mapa de Karnaugh de 3 variáveis em notação binária e decimal, respectivamente.

Vejamos através 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 0-cubos devem ser marcados com 1 no mapa, conforme mostrado no mapa da esquerda na figura 1.7. O processo consiste, então, em procurar, para cada 0-cubo da função, o maior cubo da função que o cobre. Isto, no mapa de Karnaugh, corresponde a juntar o 0-cubo em questão com 0-cubos adjacentes a ele de forma a sempre formar um retângulo (ou quadrado) com 2^k elementos ($k >= 1$). No exemplo da figura 1.7, o maior cubo que cobre 000 é o cubo 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 $f(x_1, x_2, x_3) = \overline{x}_1 + x_2\,x_3$.

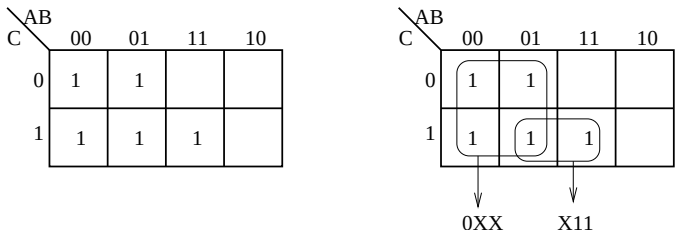


Figura 1.7: Exemplo do uso do mapa de Karnaugh: minimização da função $f = \sum m(0, 1, 2, 3, 7)$.

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 1.8.

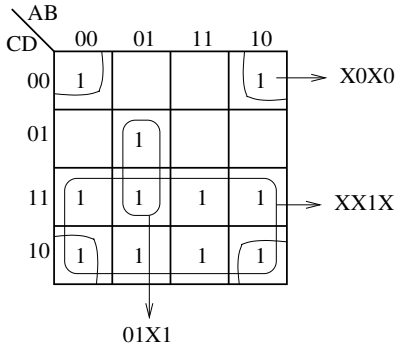


Figura 1.8: Minimização da função $f(a, b, c) = \sum m(0, 2, 3, 5, 6, 7, 8, 10, 11, 14, 15)$.

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 1.9 mostra todos os cubos maximais de f. As possíveis soluções são:

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

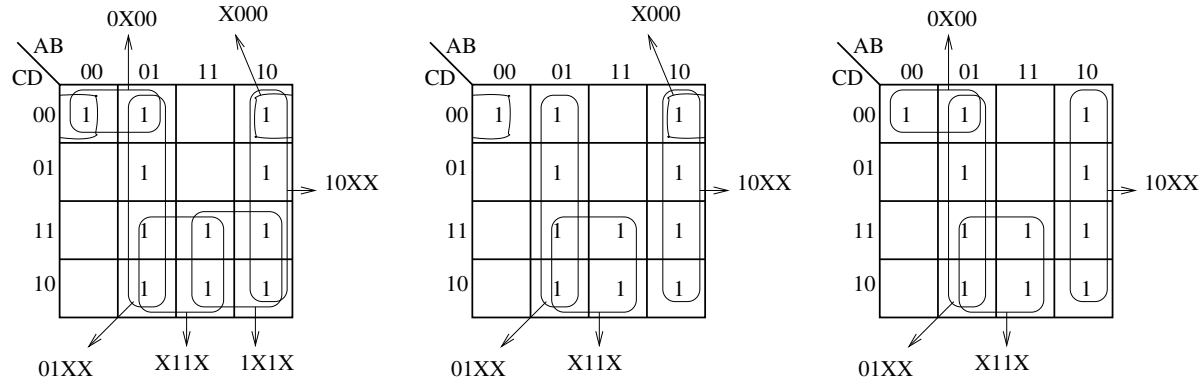


Figura 1.9: Minimização da função $f(a, b, c) = \sum m(0, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15)$. Esquerda: todos os implicantes primos (ou cubos maximais). Centro: uma solução. Direita: outra solução.

Mapa de Karnaugh para encontrar a forma POS minimal: Será que o mapa de Karnaugh pode ser utilizado também para se 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 1.10. A forma SOP minimal é $f(a,b,c) = \overline{b}\,\overline{c} + a\,c$.

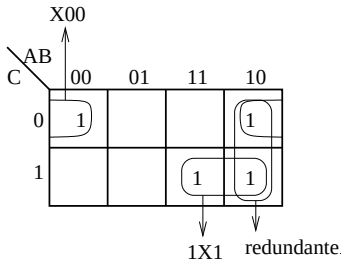


Figura 1.10: Exemplo do uso do mapa de Karnaugh: minimização da função $f(a,b,c) = \sum m(0,4,5,7)$.

A minimização SOP de $\overline{f}(a,b,c) = \sum m(1,2,3,6)$ por mapa de Karnaugh é mostrada na figura 1.11. O resultado obtido é $\overline{f}(a,b,c) = b\,\overline{c} + \overline{a}\,c$.

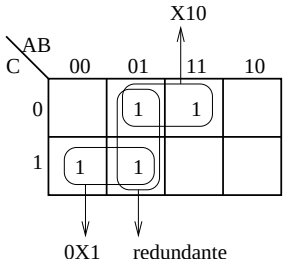


Figura 1.11: Minimização da função $\overline{f}(a,b,c) = \sum m(1,2,3,6)$.

Agora, observe que $f = \overline{\overline{f}}$. Portanto, posso escrever $f = \overline{b\,\overline{c} + \overline{a}\,c} = (\overline{b\,\overline{c}})(\overline{\overline{a}\,c}) = (\overline{b} + c)(a + \overline{c})$.

Tudo isto pode ser diretamente realizado no mapa de Karnaugh conforme mostrado na figura 1.12. Em vez de marcar os 0-cubos da função no mapa, marcamos os 0-cubos do complemento de f (ou,

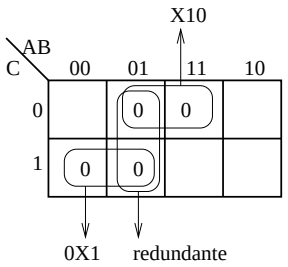


Figura 1.12: Minimização POS da função $f(a,b,c) = \prod M(1,2,3,6)$.

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

1.2.3 Minimização Tabular de Quine-McCluskey

Mapas de Karnaugh representam uma maneira visual e intuitiva de se minimizar funções booleanas. No entanto, eles só se aplicam a funções com até 6 variáveis e não são sistemáticos (adequados para programação). O algoritmo tabular de Quine-McCluskey para minimização de funções Booleanas é um método clássico que sistematiza este processo de minimização para um número arbitrário de variáveis.

Tanto os mapas de Karnaugh como o algoritmo de Quine-McCluskey (QM) requerem que a função booleana a ser minimizada esteja na forma SOP canônica. A idéia básica do algoritmo QM consiste em encarar os mintermos da SOP canônica como pontos no n-espaço, ou seja, como vértices de um n-cubo. A partir do conjunto destes vértices (ou 0-cubos) procura-se gerar todos os 1-cubos possíveis combinando-se dois deles (equivale a gerar as arestas do cubo que ligam dois 0-cubos da função). A partir da combinação de dois 1-cubos procura-se gerar todos os possíveis 2-cubos e assim por diante, até que nenhum cubo de dimensão maior possa ser gerado a partir da combinação de dois cubos de dimensão menor. Os cubos resultantes (aqueles que não foram combinados com nenhum outro) ao final de todo o processo são os **implicantes primos** (ou seja, cubos maximais) da função.

Este processo de combinar dois cubos pode ser facilmente associado ao processo algébrico de simplificação. Os mintermos da expressão na forma canônica inicial correspondem aos 0-cubos. Combinar dois 0-cubos para gerar um 1-cubo corresponde a combinar dois mintermos para eliminar uma variável e gerar um termo com menos literais para substituí-los, como mostramos no seguinte exemplo :

$$x_1x_2x_3 + x_1x_2\overline{x}_3 = x_1x_2(x_3 + \overline{x}_3) = x_1x_2 \cdot 1 = x_1x_2$$

Quando considerados no 3-espaço, o processo mostrado na expressão algébrica acima corresponde ao processo de agruparmos os 0-cubos 111 e 110 para geração do 1-cubo $11X$, como ilustra a figura 1.13.

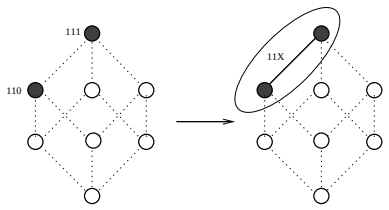


Figura 1.13: Passo elementar do algoritmo de Quine-McCluskey

À primeira vista, poderíamos afirmar que a soma de todos os implicantes primos corresponde à expressão minimal da função Booleana. No entanto, existem casos em que a soma de dois ou mais implicantes primos cobre um outro. Neste caso, este último termo é redundante, no sentido de que ele pode ser eliminado do conjunto de implicantes primos, sem que a expressão resultante deixe de ser equivalente à expressão original. Podemos ilustrar esta situação no seguinte exemplo.

Exemplo: Considere a expressão Booleana $f(a,b,c) = \sum m(0,1,3,7)$. Os implicantes primos dessa função são $00X$, $0X1$ e $X11$ (calcule usando o mapa de Karnaugh). Graficamente, estes implicantes primos (ou cubos) correspondem respectivamente aos intervalos $[000, 001]$, $[001, 011]$ e $[011, 111]$ ilustrados na figura 1.14(a). Note, porém, que o intervalo $[001, 011]$ é redundante, ou seja, a mesma expressão pode ser expressa apenas pelos implicantes primos $00X$ e $X11$ (figura 1.14(b)).

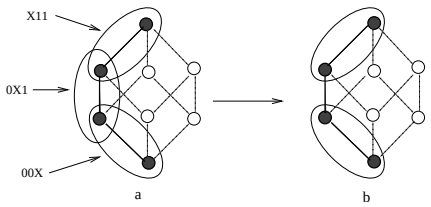


Figura 1.14: Os (a) implicantes primos e uma (b) cobertura mínima .

Conforme teorema apresentado algumas páginas atrás, sabemos que os produtos que aparecem na forma SOP minimal de uma função booleana são implicantes primos da função. Mas nem todos os implicantes primos aparecem na forma SOP minimal, como mostrado no exemplo acima. Portanto, um procedimento para obter a forma SOP minimal de uma função pode ser:

1. Calcular todos os implicantes primos da função
2. Calcular uma cobertura mínima

O ponto central da segunda etapa é o cálculo de um menor subconjunto do conjunto de implicantes primos suficientes para cobrir[1] todos os mintermos da função Booleana. Tal conjunto é denominado uma **cobertura mínima**.

No caso de mapas de Karnaugh, estas duas etapas são realizadas conjuntamente de forma um tanto "intuitiva". No caso do algoritmo QM, estas etapas são realizadas explícita e separadamente.

Cálculo de implicantes primos

A primeira etapa do algoritmo QM consiste de um processo para determinação de todos os implicantes primos. A seguir descrevemos os passos que constituem esta etapa, mostrando como exemplo o cálculo dos implicantes primos da função $f(x_1, x_2, x_3) = \sum m(0, 1, 4, 5, 6)$.

- Primeiro passo : converter os mintermos para a notação binária.
  000, 001, 100, 101, 110
- Segundo passo : Separar os mintermos em grupos de acordo com o número de 1's em sua representação binária e ordená-los em ordem crescente, em uma coluna, separando os grupos com uma linha horizontal.

000
001
100
101
110

[1] Um conjunto de implicantes primos (cubos maximais) cobre um mintermo (0-cubo) se este é coberto por pelo menos um dos implicantes primos.

- Terceiro passo : combinar todos os elementos de um grupo com todos os elementos do grupo adjacente inferior para geração de cubos de dimensão maior. Para cada 2 grupos comparados entre si, gerar um novo grupo na próxima coluna e colocar os novos cubos. Marcar com $\surd$ os cubos que foram usados para gerar novos cubos.

$\surd$	000		$00X$
$\surd$	001		$X00$
$\surd$	100	$\Longrightarrow$	$X01$
$\surd$	101		$10X$
$\surd$	110		$1X0$

  Observação : o novo cubo gerado será inserido no novo conjunto se e somente se ele ainda não estiver contido nele.

  Repetir o processo para cada nova coluna formada, até que nenhuma combinação mais seja possível.

$\surd$	000		$\surd$	$00X$		
$\surd$	001		$\surd$	$X00$		
$\surd$	100	$\Longrightarrow$	$\surd$	$X01$	$\Longrightarrow$	$X0X$
$\surd$	101		$\surd$	$10X$		
$\surd$	110			$1X0$		

- Quarto passo : Listar os implicantes primos. Os implicantes primos são os cubos que não foram combinados com nenhum outro, ou seja, aqueles que não estão com a marca $\surd$.

  $1X0$ e $X0X$

Cálculo de uma cobertura mínima

Uma cobertura mínima pode ser calculada com a ajuda de uma tabela denominada **Tabela de Implicantes Primos**. Este processo é mostrado a seguir, usando como exemplo a minimização da função $f(x_1, x_2, x_3, x_4, x_5) = \sum(1, 2, 3, 5, 9, 10, 11, 18, 19, 20, 21, 23, 25, 26, 27))$.

1. Construir a Tabela de Implicantes Primos: No topo das colunas desta tabela deve-se colocar os mintermos de f e, à esquerda de cada linha, os implicantes primos. Os implicantes primos devem ser listados em ordem decrescente de acordo com a sua dimensão, isto é, em ordem crescente de acordo com o número de literais. Deve-se acrescentar uma coluna à esquerda e uma linha na parte inferior da tabela.

   Em cada linha, marcar as colunas com $\surd$ quando o implicante primo da linha cobre o mintermo da coluna.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
	XX01X		$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$					$\surd$	$\surd$
	X10X1					$\surd$		$\surd$						$\surd$		$\surd$
	0X0X1	$\surd$		$\surd$		$\surd$		$\surd$								
	00X01	$\surd$			$\surd$											
	X0101				$\surd$							$\surd$				
	1010X										$\surd$	$\surd$				
	10X11									$\surd$			$\surd$			
	101X1											$\surd$	$\surd$			

2. Selecionar os implicantes primos essenciais: deve-se procurar na tabela as colunas que contém apenas uma marca $\surd$. A linha na qual uma dessas colunas contém a marca $\surd$ corresponde a um implicante primo essencial. Em outras palavras, este implicante primo é o único que cobre o mintermo da coluna e, portanto, não pode ser descartado. Então, deve-se marcar com um asterisco (*) esta linha na coluna mais à esquerda, para indicar que este é um implicante primo essencial. A seguir, deve-se marcar, na última linha da tabela, todas as colunas cujo mintermo é coberto pelo implicante primo selecionado.

   No exemplo, o mintermo 2 é coberto apenas pelo implicante primo $XX01X$. Logo $XX01X$ é essencial.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
*	XX01X		√	√			√	√	√	√					√	√
	X10X1					√		√						√		√
	0X0X1	√		√		√		√								
	00X01	√			√											
	X0101				√							√				
	1010X										√	√				
	10X11									√			√			
	101X1											√	√			
			√	√			√	√	√	√					√	√

   A linha correspondente a um implicante primo essencial, bem como as colunas cujos mintermos são cobertos por esse implicante primo, devem ser descondirados no prosseguimento do processo.

   Deve-se repetir o processo enquanto existir, na tabela restante, algum implicante primo essencial.

   No exemplo, vemos que o mintermo 25 é coberto apenas pelo implicante primo $X10X1$ e que o mintermo 20 é coberto apenas pelo implicante primo $1010X$. Logo, esses dois implicantes primos também são essenciais.

		1	2	3	5	9	10	11	18	19	20	21	23	25	26	27
*	XX01X		√	√			√	√	√	√					√	√
*	X10X1					√		√						√		√
	0X0X1	√		√		√		√								
	00X01	√			√											
	X0101				√							√				
*	1010X										√	√				
	10X11									√			√			
	101X1											√	√			
			√	√		√	√	√	√	√	√	√		√	√	√

3. Reduzir a tabela: eliminar as colunas cujos mintermos já foram cobertos (ou seja, manter apenas as colunas correspondentes aos mintermos não cobertos pelos implicantes primos essenciais). Eliminar as linhas correspondentes aos implicantes primos essenciais e as linhas que não cobrem nenhum dos mintermos restantes na tabela.

   No exemplo, após a redução, temos a seguinte tabela:

		1	5	23
	0X0X1	√		
	00X01	√	√	
	X0101		√	
	10X11			√
	101X1			√

4. Selecionar os implicantes primos secundariamente essenciais: eliminar as linhas dominadas e as colunas dominantes e, em seguida, selecionar os essenciais.

   Colunas dominantes: Diz-se que uma coluna β na Tabela de Implicantes Primos domina uma coluna α se e somente se todos os implicantes que cobrem o mintermo da coluna α cobrem também o mintermo da coluna β. Se β domina α, então a coluna β pode ser removida da tabela.

   Linhas dominadas ou equivalentes: Sejam A e B duas linhas na Tabela de Implicantes Primos reduzida. Então dizemos que a linha A domina B se o implicante da linha A cobre, ao menos, todos os mintermos cobertos pelo implicante da linha B. Dizemos que as linhas A e B são equivalentes se os respectivos implicantes primos cobrem exatamente os mesmos mintermos na tabela. Se A domina B, ou se A e B são equivalentes, e se a dimensão do implicante da linha A é maior ou igual ao do implicante da linha B, então a linha B pode ser eliminada da tabela.

   Após a eliminação de colunas dominantes e linhas dominadas (ou equivalentes), deve-se repetir o mesmo processo do passo 2, porém os implicantes primos essenciais serão chamados secundariamente essenciais e marcados com dois asteriscos (**).

   No exemplo, a linha do implicante primo $X0101$ pode ser eliminada pois é dominada pela linha do implicante $00X01$. A linha do implicante $101X1$ pode ser eliminada pois é equivalente a do implicante $10X11$. Neste último caso, note que, alternativamente, podemos eliminar a linha do implicante $10X11$ em vez da linha do implicante $101X1$.

		1	5	23
	0X0X1	√		
**	00X01	√	√	
**	10X11			√
		√	√	√

   Deve-se repetir o processo descrito neste passo até que não seja mais possível fazer qualquer eliminação ou até que a tabela fique vazia. Se a tabela não ficar vazia, a tabela restante é chamada **tabela cíclica**.

5. Resolver a tabela cíclica: Para isso, uma possível abordagem é o método de Petrick, um método de busca exaustiva. Ele fornece todas as possíveis combinações dos implicantes primos restantes que são suficientes para cobrir os mintermos restantes. Deve-se escolher dentre elas, uma que envolve o menor número de termos. Caso existam mais de uma nestas condições, deve-se escolher a de custo mínimo (aquela que envolve menor número de literais).
   Exemplo: Considere a tabela cíclica a seguir:

		0	4	13	15	10	26	16
a	0X10X		√	√				
b	011XX			√	√			
c	01X1X				√	√		
d	1X0X0						√	√
e	00X00	√	√					
f	X1010					√	√	
g	X0000	√						√

   Para que todos os mintermos da tabela cíclica sejam cobertos, a seguinte expressão deve ser verdadeira.

$$(e+g)(a+e)(a+b)(b+c)(c+f)(d+f)(d+g) = 1$$

Transformando esta expressão em soma de produtos, obtemos todas as possíveis soluções (cada produto é uma solução viável). Dentre os produtos, deve-se escolher aquele(s) de menor custo (menor número de implicantes primos e implicantes com menor número de literais). (Se eu não errei nos cálculos, as soluções são $\{a, c, d, e\}$, $\{b, c, d, e\}$ e $\{a, c, d, g\}$ pois os outros tem custo maior).

Outro exemplo: Considere a tabela cíclica a seguir e suponha que o custo de A é menor que o de B e que o custo de C é menor que o de D.

	m_1	m_2	m_3
A	√		
B	√	√	
C			√
D		√	√

Então as possíveis soluções são $(A + B)(B + D)(C + D) = (AB + AD + B + BD)(C + D) = (B + AD)(C + D) = BC + BD + ACD + AD$. Dos que envolvem dois implicantes, certamente o custos de BC e de AD são menores que o custo de BD. Então a escolha final fica entre BC e AD.

Resumo do Procedimento para cálculo de cobertura mínima:

1. Montar a tabela de implicantes primos

2. Identificar todos os implicantes primos essenciais e eliminar as linhas correspondentes, bem como as colunas dos mintermos cobertos por esses implicantes.

3. Eliminar colunas dominantes: Se uma coluna β tem $\surd$ em todas as linhas que uma outra coluna *alpha* tem $\surd$, a coluna β é dominante e pode ser eliminada (pois se escolhermos um implicante primo que cobre α, β será necessariamente coberto também).

4. Eliminar linhas dominadas ou equivalentes: se uma linha A tem $\surd$ em todas as colunas em que a linha B tem $\surd$, então a linha A domina a linha B. Se elas tem $\surd$ exatamente nas mesmas colunas, então elas são equivalentes. Se, além disso, o número de literais de A é menor que o de B, então a linha B pode ser eliminada (pois se tivéssemos uma cobertura envolvendo B, ao trocarmos B por A na cobertura teríamos uma cobertura de menor custo).

   Observação: Se o objetivo da minimização é encontrar apenas UMA solução minimal (e NÃO TODAS), então podemos eliminar uma linha B se existe uma linha A tal que A domina B, ou A é equivalente a B, e ambos têm um mesmo custo.

5. Identificar os implicantes essenciais secundários e eliminar as linhas correspondentes, bem como as colunas dos mintermos cobertos por esses implicantes.

6. Repetir 3, 4 e 5 enquanto possível

7. Se a tabela não estiver vazia, aplicar o método de Petrick (que lista todas as possíveis soluções para o restante da tabela) e escolher uma solução de custo mínimo.

Exemplo: Considere a função $f(a,b,c) = a\,\overline{b}\,c + \overline{a}\,b\,\overline{c} + a\,b\,\overline{c} + a\,b\,c = \sum m(2,5,6,7)$.

Podemos realizar a simplificação algébrica da seguinte forma:

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

Por QM temos

√	010
√	101
√	110
√	111

$\Longrightarrow$

$X10$
$1X1$
$11X$

Os implicantes primos são $X10$ ($b\,\overline{c}$), $1X1$ ($a\,c$) e $11X$ ($a\,b$). Uma cobertura mínima pode ser calculada usando-se a Tabela de Implicantes Primos.

		2	5	6	7
*	X10	√		√	
*	1X1		√		√
	11X			√	√
		√	√	√	√

Os implicantes primos $X10$ e $1X1$ são essencias e cobrem todos os mintermos da função. Logo formam uma cobertura mínima.

1.2.4 Funções incompletamente especificadas

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 pelo algoritmo QM, é interessante considerarmos todos os don't cares na etapa de cálculo dos implicantes primos, pois isto aumenta a chance de obter cubos maiores (portanto produtos com menos literais). Observe que, durante as iterações para a geração dos implicantes primos, um cubo que cobre apenas don't cares não pode ser eliminado pois ele pode, eventualmente em iterações futuras, se juntar a outro cubo para formar outro cubo maior.

De forma mais genérica do que a vista anteriormente, podemos então caracterizar uma função booleana f através dos seus conjuntos um, zero e dc (de don't care), definidos respectivamente por $f\langle 1\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 1\}$, $f\langle 0\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 0\}$ e $f\langle *\rangle = \{0,1\}^n \setminus (f\langle 1\rangle \cup f\langle 0\rangle)$.

Na parte de cálculo dos implicantes primos devem ser utilizados todos os elementos de $f\langle 1\rangle \cup f\langle *\rangle$. Na parte de cálculo de uma cobertura mínima devem ser considerados apenas os elementos de $f\langle 1\rangle$.

1.2.5 Cálculo da forma POS minimal

Similarmente ao que já vimos na minimização por mapas de Karnaugh, para se obter a forma POS minimal de uma função procede-se da seguinte forma. No cálculo dos implicantes primos, em vez de listar os mintermos, lista-se os 0s da função e aplica-se o método tabular. Realiza-se o cálculo da cobertura mínima utilizando-se nas colunas os 0s da função. Ao final, expressa-se o resultado como produto dos implicantes primos selecionados complementados.

A explicação é a seguinte: aos tomarmos os 0s da função em vez dos 1s, estamos considerando a minimização SOP de $\overline{f}$. Agora, uma vez que $f = \overline{\overline{f}}$, ao complementarmos a forma SOP minimal de $\overline{f}$, obtemos a forma POS minimal de f.

Exemplo: Minimizar na forma POS a função $f(a,b,c) = \prod M(3,6,7)$. Algebricamente temos:

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

Por QM temos:

√	011
√	110
√	111

$\Longrightarrow$

$X11$
$11X$

Os implicantes são $X11$ e $11X$, que escritos na forma de produtos correspondem respectivamente a bc e ab. Complementando estes produtos temos: $\overline{b}+\overline{c}$ e $\overline{a}+\overline{b}$, que são as somas que aparecem na forma POS minimal.

1.2.6 O algoritmo ISI

Podemos dizer que o algoritmo QM utiliza uma abordagem *bottom-up* para gerar todos os implicantes primos. Outra possível abordagem é a *top-down*, utilizada pelo ISI (*Inremental splitting of intervals*). O ISI inicia o processo a partir do n-cubo e, sucessivamente, elimina os zeros da função, tomando cuidado em representar os elementos que restam após uma eliminação através do conjunto de seus intervalos maximais. Assim, depois de eliminar todos os zeros, os elementos restantes correspondem aos uns (mintermos) da função, os quais estarão representados pelos intervalos maximais, ou seja, pelos implicantes primos da função.

Para fazer paralelo com algo mais concreto, podemos pensar que o QM é aquele processo em que o fulano constrói uma parede juntando os tijolos, enquanto o ISI é aquele em que o fulano esculpe uma estátua removendo partes de uma pedra (ou madeira).

Passo básico do ISI

Remover um elemento de um cubo (intervalo) e representar os elementos restantes pelos subintervalos maximais contidos neles é a chave do algoritmo ISI.

Sejam $[A, B]$ e $[C, D]$ dois intervalos em $\{0,1\}^n$. A diferença de $[A, B]$ e $[C, D]$ é o conjunto

$$[A, B] \setminus [C, D] = \{\mathbf{x} \in \{0,1\}^n : \mathbf{x} \in [A, B] \text{ e } \mathbf{x} \notin [C, D]\} \tag{1.1}$$

que pode também ser expresso por $[A, B] \setminus [C, D] = [A, B] \cap [C, D]^c$.

Proposição: Sejam $[A, B]$ e $[C, D]$ dois intervalos de $\{0,1\}^n$ cuja interseção é não-vazia. Então,

$$\begin{aligned}[A, B] \setminus [C, D] = \\ \Big\{[A, B'] : B' \text{ é elemento maximal daqueles que não contém nenhum elemento de } [C, D]\Big\} \cup \\ \Big\{[A', B] : A' \text{ é elemento minimal daqueles que não estão contidos em nenhum elemento de } [C, D]\Big\}.\end{aligned} \tag{1.2}$$

A equação acima mostra como a diferença de intervalos pode ser expressa em termos da coleção de intervalos maximais contidos na diferença.

Se $dim([A, B]) = k$, qualquer intervalo maximal contido na diferença tem dimensão $k - 1$. O número de intervalos maximais contidos na diferença é dado por $dim([A, B]) - dim([A, B] \cap [C, D])$.

Exemplo: Mostramos a seguir algumas diferenças representadas pelos intervalos maximais contidos na diferença.

Sejam $[A, B] = [000, 111]$ e $[C, D] = [001, 111]$. O número de intervalos em $[A, B] \setminus [C, D]$ é $dim([A, B]) - dim([A, B] \cap [C, D]) = 3 - 2 = 1$ e a dimensão dos intervalos resultantes é $dim([A, B]) - 1 = 3 - 1 = 2$. Veja Fig. 1.15.

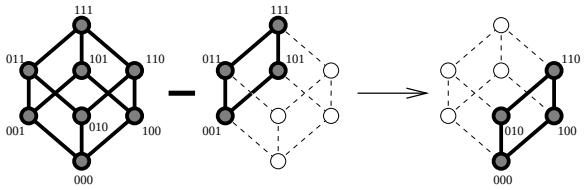


Figura 1.15: $[000, 111] \setminus [001, 111] = \{[000, 110]\}$.

Sejam $[A, B] = [000, 111]$ e $[C, D] = [001, 011]$. O número de intervalos em $[A, B] \setminus [C, D]$ é $dim([A, B]) - dim([A, B] \cap [C, D]) = 3 - 1 = 2$ e a dimensão dos intervalos resultantes é $dim([A, B]) - 1 = 3 - 1 = 2$. Veja Fig. 1.16.

Sejam $[A, B] = [000, 111]$ e $[C, D] = [011, 011]$. Este é o caso em que apenas um ponto do cubo é removido. O número de intervalos em $[A, B] \setminus [C, D]$ é $dim([A, B]) - dim([A, B] \cap [C, D]) = 3 - 0 = 3$ e a dimensão dos intervalos resultantes é $dim([A, B]) - 1 = 3 - 1 = 2$. Veja Fig. 1.17.

Um exemplo completo

Consideremos a minimização da função $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3$. Temos $f\langle 0\rangle = \{111, 011, 010\}$. A figura 1.18 mostra os elementos que permanecem após as sucessivas remoções dos elementos em $f\langle 0\rangle$. Cada seta indica um passo de remoção (note que não mostramos os intervalos maximais resultantes individualmente, mostramos os elementos restantes em negrito). A figura 1.19 mostra o mesmo processo em uma estrutura de árvore. Cada nível da árvore corresponde ao resultado após um passo de remoção. Note que alguns intervalos podem ser descartados por estarem contidos em outros.

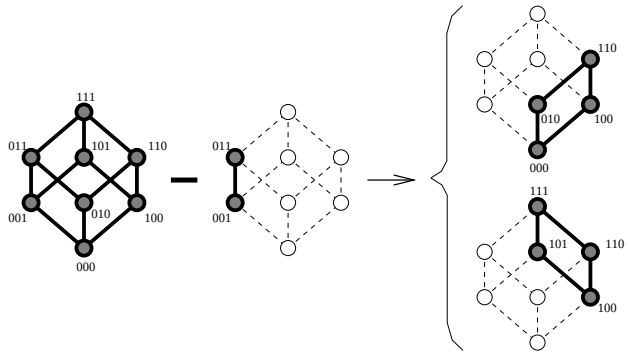


Figura 1.16: $[000, 111] \setminus [001, 011] = \{[000, 110], [100, 111]\}$.

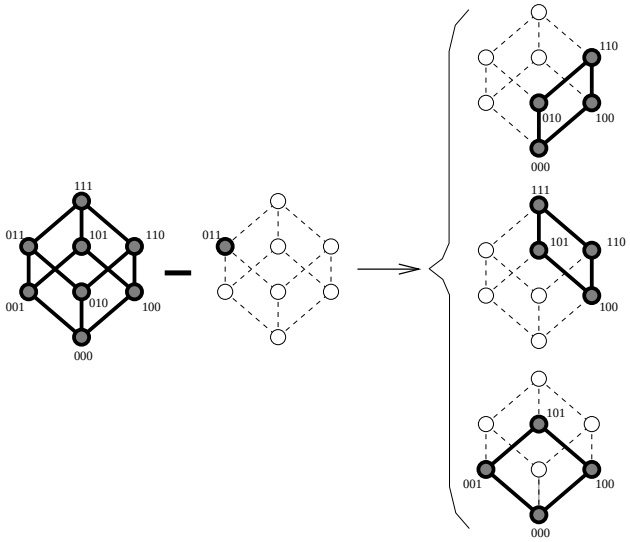


Figura 1.17: $[000, 111] \setminus [011, 011] = \{[000, 110], [100, 111], [000, 101]\}$.

ISI: Minimização de funções incompletamente especificadas

O algoritmo ISI remove sucessivamente os zeros do cubo inicial de forma a obter, ao final do processo, o conjunto de todos os intervalos maximais de $f\langle 1\rangle$ (i.e., implicantes primos). Se a função possui don't cares, então ao final do processo serão obtidos os intervalos maximais de $f\langle 1\rangle \cup f\langle *\rangle$. No entanto, intervalos que são formados inteiramente por elementos de $f\langle *\rangle$ não serão necessários na etapa de cálculo de uma cobertura mínima. Portanto, estes podem ser descartados, assim que são detectados durante o processo de remoção dos zeros.

Mostramos a idéia através da sua aplicação na minimização de f, com $f\langle 0\rangle = \{010, 111\}$, $f\langle 1\rangle = \{000, 001, 101\}$ e $f\langle *\rangle = \{100, 011, 110\}$, cujo processo é ilustrado nas figuras 1.20 and 1.21.

Os elementos que são don't cares são representados no diagrama de Hasse como vértices sem os círculos; círculos preenchidos representam elementos de $f\langle 1\rangle$ enquanto os não preenchidos representam os de $f\langle 0\rangle$. As arestas em negrito indicam que os elementos adjacentes a ela fazem parte de um intervalo.

No início do processo, todos os elementos fazem parte do intervalo XXX. Após a remoção de 111, três intervalos são gerados: $0XX$, $X0X$ and $XX0$. Nenhum deles pode ser descartado pois todos eles contém pelo menos um elemento de $f\langle 1\rangle$. Em seguida, o elemento 010 é removido dos intervalos $0XX$,

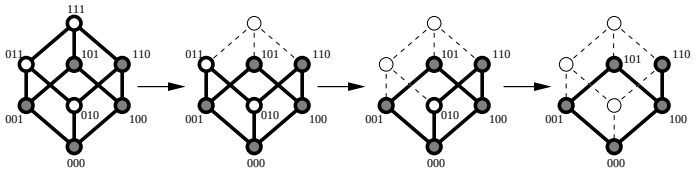


Figura 1.18: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 011, 111\}$ e $f\langle 1\rangle = \{000, 001, 100, 101, 110\}$). Ordem de remoção: 111, 011 e 010.

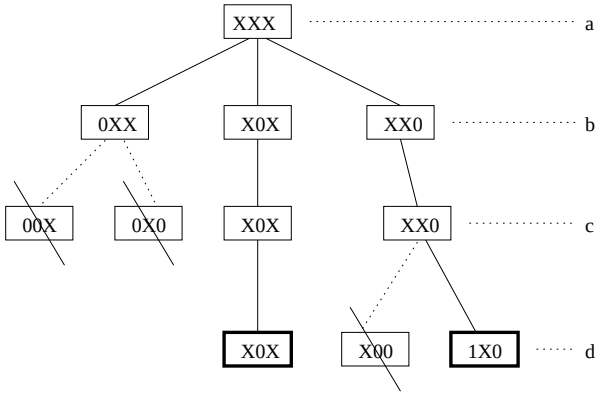


Figura 1.19: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 011, 111\}$ e $f\langle 1\rangle = \{000, 001, 100, 101, 110\}$). (a) intervalo inicial; (b) após remoção de 111 ; (c) após remoção de 011; (d) resultado final, após remoção de 010.

and $XX0$, produzindo os intervalos $00X, 0X1$ and $X00, 1X0$, respectivamente. Os intervalos $00X$ and $X00$ podem ser descartados pois estão contidos em $X0X$. O intervalo $1X0$ pode ser descartado pois não cobre nenhum elemento de $f\langle 1\rangle$. Assim, os intervalos que restam são $X0X$ and $0X1$. O segundo será eliminado no processo de cálculo de cobertura mínima.

Note que a função resutante é consistente com a inicial; dos elementos em $f\langle *\rangle$, 011 e 110 são mapeados para 0, e somente 100 é mapeado para 1.

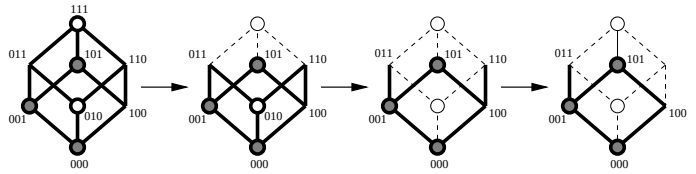


Figura 1.20: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 111\}$, $f\langle 1\rangle = \{000, 001, 101\}$ e $f\langle *\rangle = \{100, 011, 110\}$).

Exercícios

1. Minimize a função $f(a, b, c) = \sum m(1, 2, 3, 4, 5, 6)$.
2. Minimize a função $f(a, b, c, d) = \sum m(0, 2, 3, 6, 7, 8, 9, 10, 13)$.
3. Minimize a função $f(a, b, c, d,) = \sum m(0, 4, 5, 6, 7, 8, 9, 10, 11, 14, 15)$ por QM.

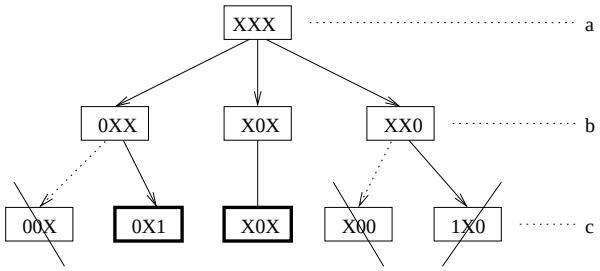


Figura 1.21: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 111\}$, $f\langle 1\rangle = \{000, 001, 101\}$ e $f\langle *\rangle = \{100, 011, 110\}$).

Confira os resultados: os implicantes primos são 01XX, 10XX, X11X, 1X1X, 0X00, X000 e os essenciais são 01XX, 10XX. Chega-se a uma tabela cíclica constituída dos implicantes 0X00 (1), X000 (2), X11X (3), 1X1X (4) e pelo métdo de Petrick conclui-se que as possíveis soluções são dadas por $(1+2)(3+4) = 1, 3+1, 4+2, 3+2, 4$. Ou seja, há quatro possíveis soluções de custo equivalente. Compare coma solução por mapa de Karnaugh (figura 1.9).

4. Minimizar $f(a, b, c, d) = \sum m(0, 2, 8, 12, 13) = \prod M(1, 3, 4, 5, 6, 7, 9, 10, 11, 14, 15)$.
   Resposta (conferir): $f = 00X0 + 110X + 1X00$ ou $f = 00X0 + 110X + X000$, $f = (\overline{a} + \overline{c})(a + \overline{b})(b + \overline{d})$.
5. O que se pode dizer sobre a forma SOP e POS minimal de uma função? Tem uma que sempre utiliza menor número de portas ou o número de portas nas ambas as formas podem ser iguais?
6. Minimizar $f(w, x, y, z) = \sum m(0, 7, 8, 10, 12) + d(2, 6, 11)$.
   Resposta (conferir): $\overline{x}\,\overline{z} + w\,\overline{y}\,\overline{z} + \overline{w}\,x\,y$ ($X0X0$, $1X00$ e $011X$)
7. Minimizar na forma POS a função $f(x, y, z) = \prod M(0, 1, 6, 7)$
8. Minimizar na forma SOP a função $f(a, b, c, d) = \sum m(0, 1, 4, 5, 12, 13)$
9. Minimizar na forma SOP a função $f(a, b, c, d) = \sum m(0, 2, 8, 9) + d(1, 13)$
10. Minimize a função $f(a, b, c, d, e) = \sum m(1, 2, 3, 5, 9, 10, 11, 18, 19, 20, 21, 23, 25, 26, 27)$.
11. Minimizar a função $f(a, b, c, d, e) = \sum m(0, 4, 10, 13, 15, 16, 22, 23, 26) + d(5, 11, 12, 14, 18, 21, 24)$.
12. Minimizar na forma SOP a função $f(a, b, c, d, e) = \sum m(0, 1, 2, 9, 13, 16, 18, 24, 25) + d(8, 10, 17, 19)$

1.3 Minimização dois-níveis de múltiplas funções

1.3.1 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 1.22 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 $f_3(a,b,c) = \overline{a}\,b\,\overline{c} + a\,\overline{c}$.

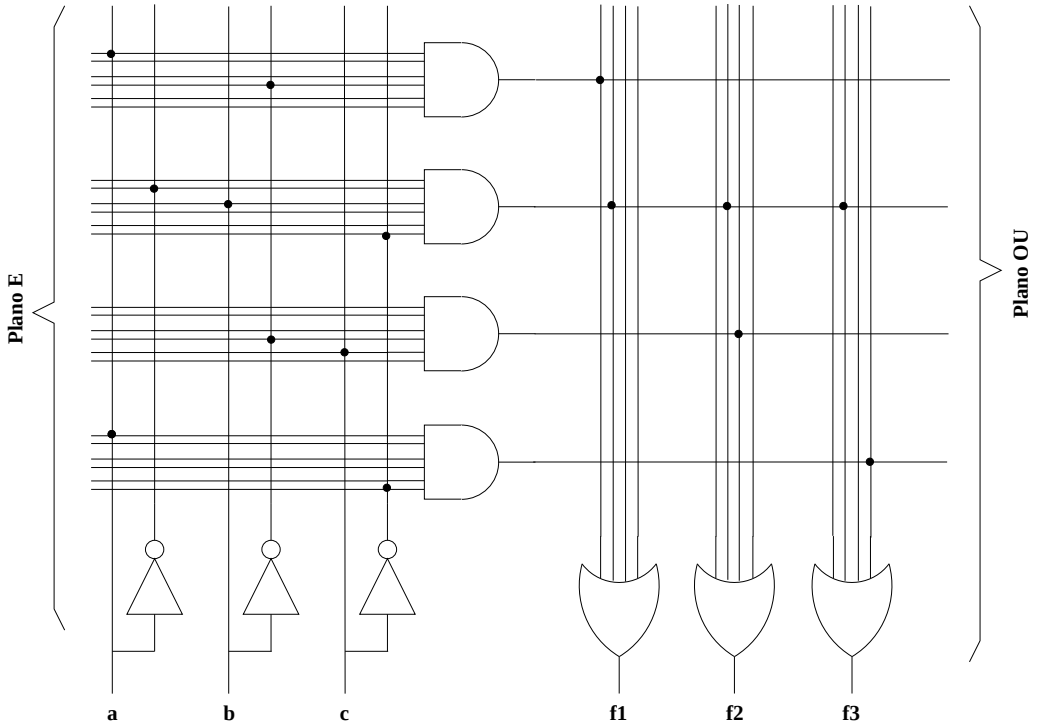


Figura 1.22: Esquema lógico de um PLA.

Para não sobrecarregar o diagrama, em geral desenha-se de forma simplificada como o mostrado na figura 1.23.

PLAs comerciais têm tipicamente 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 $2\times16\times48 = 1536$ cruzamentos na malha E e $8\times48 = 384$ 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.

1.3.2 Minimização conjunta

Uma vez que em um PLA podem ser realizadas várias funções, a minimização de um conjunto de funções (e não apenas de uma única função) torna-se o alvo de interesse. Vamos analisar dois exemplos:

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

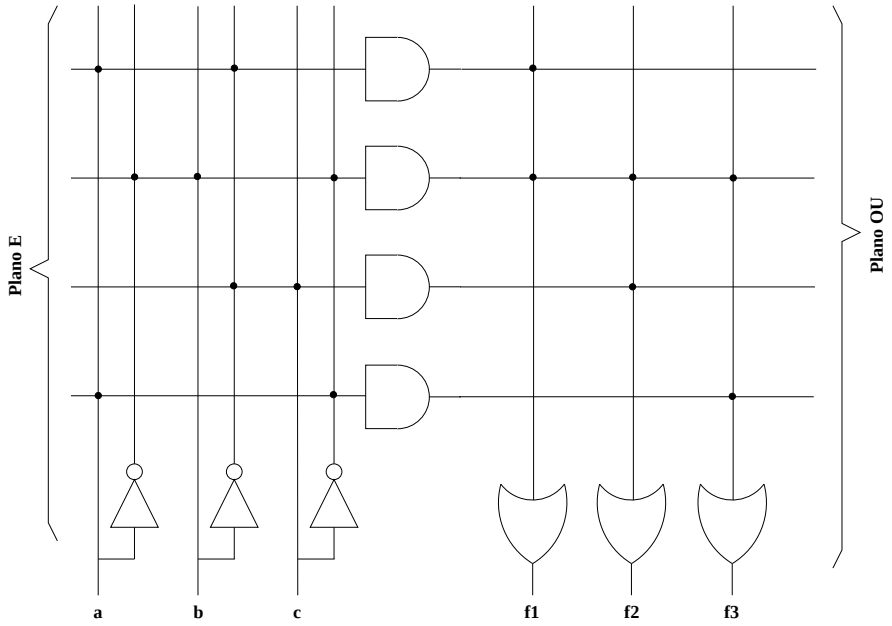


Figura 1.23: Esquema simplificado de um PLA.

$x_1\,x_2\,x_3$	$f_1(x_1,x_2,x_3)$
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 em PLA, 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 em PLA, 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 pôde-se reduzir o número total de produtos, mas o número de somas aumentou. Existe alguma vantagem nisso ? Que impacto isto tem no custo de implementação ? Por essas e outras, minimização de múltiplas funções é um problema mais complexo que o da minimização individual de funções.

Exercícios:

1. Desenhe UM circuito que implementa as duas funções do Exemplo 1, utilizando apenas portas NÃO-E.

2. Desenhe o PLA que realiza as quatro funções do Exemplo 2, utilizando 4 portas E.

Considerações sobre o custo a ser minimizado

Quando pensamos em minimizar um conjunto de funções, reduzir o número de certos elementos necessários para a sua implementação é a principal preocupação. Entre este elementos, alguns são bastante intuitivos:

- reduzir o número total de produtos. Em um PLA, significa reduzir o número de linhas no plano E.
- reduzir o número de entradas para as portas E (tentar usar produtos com o menor número possível de literais)
- reduzir o número de entradas para as portas OU (ou seja, utilizar o menor número possível de produtos para cada função)

Estas noções podem ser equacionadas da seguinte forma :

> CUSTO = número total de portas E + a (número total de entradas para as portas E) + b(número total de entradas para as portas OU).

com $0 \leq a, b < 1$ (i.e., minimizar o número total de produtos é o principal critério; os outros são considerados secundários ou irrelevantes).

Em um PLA com certo número fixo de entradas, de portas E e de portas OU, a área do chip também é fixa. Em particular, a área ocupada por um implicante primo (produto) independe do seu número de literais. Portanto, podemos dizer que o custo de todos os implicantes primos é o mesmo e é razoável assumirmos $a = 0$. Em termos de seleção de uma cobertura mínima, isto implica que o número de literais nos implicantes primos não é relevante. Similarmente, utilizar um produto a mais do que o mínimo necessário para a realização de uma função (ou seja, aumentar o número de entradas em uma porta OU) não afeta o custo; ou seja, $b = 0$. Isto significa que uma vez que um implicante primo é identificado como essencial para uma das funções, ele pode ser utilizado por qualquer uma das outras funções (mesmo que não seja implicante primo dela), sem custo adicional. Assim, na minimização

de funções com o objetivo de implementação em PLA é (de um modo geral) razoável considerarmos $a = b = 0$, isto é, preocuparmo-nos em apenas minimizar o número total de produtos necessários para realizar todas as funções.

A realização das funções de outra forma, não utilizando PLA, pode requerer a e b não nulos. No entanto, ainda é razoável supormos $a \ll 1$ e $b \ll 1$ (isto é, o principal critério é a minimização do número total de produtos). Se utilizarmos $a \ll 1$ e $b = 0$, a chance de termos uma tabela cíclica complicada é maior do que quando utilizamos $a = b = 0$.

Quando consideramos a minimização de múltiplas funções, podemos aplicar um processo análogo ao do algoritmo QM. Numa primeira etapa são calculados todos os implicantes primos e numa segunda etapa é escolhida uma cobertura mínima. O problema da escolha de uma cobertura mínima pode ficar mais simples se considerarmos $a = b = 0$ na equação do custo acima (veremos isso mais adiante através de exemplos).

A noção de implicante primo é agora definida com relação às múltiplas funções. Devemos considerar não apenas os implicantes primos de cada uma das funções, mas também todos os implicantes maximais que podem ser compartilhados por 2 ou mais funções. Vamos esclarecer isto analisando um exemplo concreto.

Exemplo 3: Considere as funções do exemplo 1, escritas na forma SOP canônica

$$f_1(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2x_3 = \sum m(0, 1, 5)$$

$$f_2(x_1, x_2, x_3) = x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3 = \sum m(5, 6, 7)$$

Escritos na notação cúbica, os mintermos de f_1 são 000, 001 e 101 enquanto os de f_2 são 101, 110 e 111. Os implicantes primos (com respeito a f_1 e f_2) são:

a) os implicantes primos de f_1: $00X$ e $X01$

b) os implicantes primos de f_2: $1X1$ e $11X$

c) os implicantes de f_1 e de f_2 que não são cobertos por outro implicante de f_1 e de f_2: 101 (ou seja, 101 é implicante (não é primo!) de ambas as funções e não existe nenhum outro implicante de f_1 e de f_2 que o cobre).

Na minimização de múltiplas funções, compartilhar produtos é uma forma de se minimizar o número de portas E no circuito. Os implicantes do tipo (c) são, em outras palavras, os maiores subcubos compartilhados entre duas ou mais funções. Portanto, o critério de minimização deve levar em consideração não apenas os implicantes primos de cada função, mas também todos os implicantes do tipo do item (c).

O algoritmo QM adaptado para o cálculo de implicantes primos de múltiplas funções

Exemplo 4: Considere novamente as funções do exemplo 1: $f_1(x_1, x_2, x_3) = \sum m(0, 1, 5)$ e $f_2(x_1, x_2, x_3) = \sum m(5, 6, 7)$.

A tabela 1.1 mostra o cálculo dos implicantes primos de $\mathbf{f} = (f_1, f_2)$. Inicialmente, listam-se todos os mintermos (independente da função a qual eles pertencem), um em cada linha, agrupados de acordo

com o número de ocorrência de 1s. Ao lado, cria-se uma coluna para cada uma das funções e, para cada coluna, marcam-se as linhas correspondentes aos mintermos da função. Assim, por exemplo, na coluna correspondente à função f_1 aparece 1 nas linhas dos mintermos 000, 001 e 101.

Para gerar os implicantes primos, procede-se de forma análoga ao da minimização de uma única função, tomando-se cuidado para (1) combinar somente os pares de subcubos que fazem parte de pelo menos uma mesma função e (2) quando dois cubos C e C' são combinados e um novo cubo C'' é gerado, marcar o subcubo C para descarte somente se o cubo gerado C'' é também cubo das mesmas funções das quais C é cubo (idem para C'). Por exemplo, 001 pode ser combinado com 101 para gerar o cubo $X01$. Neste momento, o cubo 001 (da função f_1) pode ser descartado pois ele é coberto por $X01$ (que também é cubo da função f_1). No entanto, o cubo 101 (das funções f_1 e f_2) NÃO pode ser descartado pois o cubo $X01$ que o cobre não é cubo da função f_2. Quando dois cubos são combinados e um novo cubo é gerado, ele deve ser colocado em uma outra tabela (mais a direita na figura 1.1), indicando-se a qual das funções ela pertence. O processo deve ser repetido até que nenhuma combinação seja mais possível. Ao final do processo, os implicantes candidatos são aqueles que não foram marcados para descarte ao longo do processo. No exemplo da figura 1.1 são aqueles marcados por a, b, c, d e e.

$x_1x_2x_3$	f_1	f_2	
000	1		$\surd$
001	1		$\surd$
101	1	1	e
110		1	$\surd$
111		1	$\surd$

$x_1x_2x_3$	f_1	f_2	
00X	1		a
X01	1		b
1X1		1	c
11X		1	d

Tabela 1.1: Método tabular para cálculo dos implicantes primos de $\mathbf{f} = (f_1, f_2)$.

O segundo passo do algoritmo é a seleção de uma cobertura mínima. A tabela 1.2 mostra a **tabela de implicantes primos** de $\mathbf{f} = (f_1, f_2)$, utilizada para a seleção de uma cobertuta mínima. A tabela contém colunas correpondentes a cada um dos mintermos de cada uma das funções, e linhas correspondentes aos implicantes primos calculados no passo anterior. Os números ao lado esquerdo de um implicante primo indica que ele é implicante daquelas funções. Por exemplo, na coluna ao lado esquerdo de $00X$ aparece (1) para indicar que $00X$ é um implicante da função f_1; ao lado de 101 aparece $(1, 2)$ para indicar que 101 é implicante de f_1 e de f_2. Para cada linha, marcam-se com $\surd$ as colunas correspondentes aos mintermos cobertos pelo implicante primo, desde que o implicante primo seja implicante da função correspondente ao mintermo (por exemplo, $X01$ cobre 101, mas não se marca na coluna 101 da função f_2 pois $X01$ não é implicante de f_2).

Após a construção da tabela, deve-se primeiramente selecionar os implicantes primos essencias, que são marcados com $*$ ($00X$ e $11X$). Procede-se com a redução da tabela, eliminando-se a linha dos essenciais, bem como as colunas dos mintermos cobertos por eles. Após a redução da tabela, obtemos a tabela da direita da figura 1.2.

Se levarmos em consideração apenas a minimização do número de produtos, podemos dizer que o implicante (e) domina os implicantes (b) e (c). Portanto, as linhas (b) e (c) podem ser eliminadas, resultando apenas a linha (e). Temos então o resultado $00X$, $11X$ e 101.

No entanto, se levarmos em conta também, além da minimização do número de produtos, o número de literais em cada produto, não podemos dizer que a linha (e) domina as outras duas. Neste caso, temos uma table cíclica e utilizaremos o método de Petrick para resolvê-lo.

			f_1			f_2		
			000	001	101	101	110	111
*	1	(a) 00X	√	√				
	1	(b) X01		√	√			
	2	(c) 1X1				√		√
*	2	(d) 11X					√	√
	1,2	(e) 101			√	√		
			√	√		√		√

			f_1	f_2
			101	101
	1	(b) X01	√	
	2	(c) 1X1		√
	1,2	(e) 101	√	√

Tabela 1.2: Tabela de implicantes primos de $\mathbf{f} = (f_1, f_2)$.

Para cobrir ambas as colunas, a seguinte igualdade deve ser verdadeira:

$$(b + e)(c + e) = 1$$

Escrevendo a expressão acima na forma SOP, temos

$$bc + be + ce + e = 1$$

Daqui podemos concluir que a solução de menor custo é escolher (e). Assim, temos o resultado $00X$, $11X$ e 101 (por coincidência, o mesmo obtido considerando o custo mais simples).

Exercício: Desenhe o PLA que realiza as duas funções minimizadas do exemplo 4.

Nem sempre as soluções relativas ao custo mais simples serão as mesmas às relativas ao custo mais complexo, como veremos no seguinte exemplo.

Exemplo 5: Considere as seguintes funções.
$f_\alpha(a, b, c, d) = \sum m(2, 4, 10, 11, 12, 13)$
$f_\beta(a, b, c, d) = \sum m(4, 5, 10, 11, 13)$
$f_\gamma(a, b, c, d) = \sum m(1, 2, 3, 10, 11, 12)$

O processo de cálculo dos implicantes primos pode ser realizado através dos mapas de Karnaugh. Na figura 1.24 aparecem 7 mapas de Karnaugh, das funções f_α, f_β, f_γ, $f_\alpha \cdot f_\beta$, $f_\alpha \cdot f_\gamma$, $f_\beta \cdot f_\gamma$ e $f_\alpha \cdot f_\beta \cdot f_\gamma$, respectivamente.

Começa-se marcando os implicantes primos (cubos maximais) no mapa da função $f_\alpha \cdot f_\beta \cdot f_\gamma$. O único implicante primo de $f_\alpha \cdot f_\beta \cdot f_\gamma$ é $101X$. Em seguida, marcam-se os implicantes primos da interseção de duas funções, desde que os mesmos não sejam cobertos pelos implicantes de um produto de funções de ordem maior. Por xemplo, em $f_\alpha \cdot f_\gamma$, $101X$ é um implicante primo mas ele é coberto pelo implicante primo $101X$ de $f_\alpha \cdot f_\beta \cdot f_\gamma$. Portanto, este implicante primo não é marcado. Repete-se o processo para cada uma das funções, marcando-se apenas os implicantes primos que não aparecem em nenhuma das funções $f_\alpha \cdot f_\beta$, $f_\alpha \cdot f_\gamma$, $f_\beta \cdot f_\gamma$ e $f_\alpha \cdot f_\beta \cdot f_\gamma$.

Assim, os implicantes obtidos são os mostrados na tabela a seguir. A coluna da esquerda indica as funções implicadas pelo implicante.

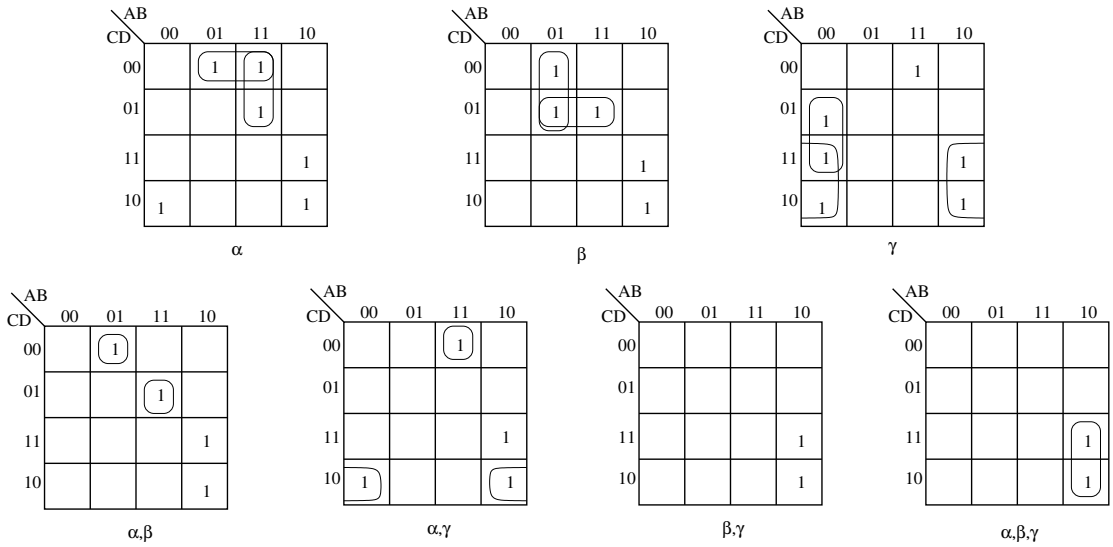


Figura 1.24: Implicantes primos da função $\mathbf{f} = (f_\alpha, f_\beta, f_\gamma)$.

$101X$	$\alpha\ \beta\ \gamma$
$X010$	$\alpha\ \gamma$
1100	$\alpha\ \gamma$
0100	$\alpha\ \beta$
1101	$\alpha\ \beta$
$X01X$	γ
$00X1$	γ
$X101$	β
$010X$	β
$110X$	α
$X100$	α

O método tabular para determinação dos implicantes primos é mostrado na tabela 1.3.

	f_α	f_β	f_γ	IP
0001			1	√
0010	1		1	√
0100	1	1		i
0011			1	√
0101		1		√
1010	1	1	1	√
1100	1		1	j
1011	1	1	1	√
1101	1	1		k

	f_α	f_β	f_γ	IP
00X1			1	f
001X			1	√
X010	1		1	g
010X		1		d
X100	1			b
X011			1	√
X101		1		e
101X	1	1	1	h
110X	1			c

	f_α	f_β	f_γ	IP
X01X			1	a

Tabela 1.3: Método tabular para cálculo dos implicantes primos de $\mathbf{f} = (f_\alpha, f_\beta, f_\gamma)$.

Compare os implicantes primos obtidos pelos mapas de Karnaugh e pelo método QM adaptado.

A tabela de implicantes primos é mostrada na figura 1.4. Primeiramente são identificados os implicantes essenciais.

			f_α						f_β					f_γ					
			2	4	10	11	12	13	4	5	10	11	13	1	2	3	10	11	12
	γ	(a) X01X													√	√	√	√	
	α	(b) X100		√			√												
	α	(c) 110X					√	√											
	β	(d) 010X							√	√									
	β	(e) X101								√			√						
*	γ	(f) 00X1												√		√			
*	$\alpha\gamma$	(g) X010	√		√										√		√		
*	$\alpha\beta\gamma$	(h) 101X			√	√					√	√					√	√	
	$\alpha\beta$	(i) 0100		√					√										
*	$\alpha\gamma$	(j) 1100					√												√
	$\alpha\beta$	(k) 1101						√					√						
			√		√	√	√				√	√		√	√	√	√	√	√

Tabela 1.4: Tabela dos implicantes primos e seleção de uma cobertura mínima de $\mathbf{f} = (f_\alpha, f_\beta, f_\gamma)$.

Obtemos os **essenciais:** f, g, h, j

Caso 1: Minimizar APENAS o número de produtos (implementação em PLA)

			f_α		f_β		
			4	13	4	5	13
	α	(b) X100	√				
	α	(c) 110X		√			
	β	(d) 010X			√	√	
	β	(e) X101				√	√
**	$\alpha\beta$	(i) 0100	√		√		
**	$\alpha\beta$	(k) 1101		√			√
			√	√	√	√	√

- o número de literais presentes nos implicantes não é importante: b e c podem ser eliminados pois são dominados por i e k, respectivamente.
- i e k são essenciais secundários.
- Para cobrir 5 podemos selecionar d ou e.

RESULTADO (ao selecionarmos d)

$$f_\alpha : g + h + i + j + k = X010 + 101X + 0100 + 1100 + 1101$$
$$f_\beta : d + h + i + k = X101 + 101X + 0100 + 1101$$
$$f_\gamma : f + g + h + j = 00X1 + X010 + 1100 + 101X$$

Note que na expressão de f_β, o termo 1101 é redundante e portanto pode ser eliminado. Assim temos $f_\beta = d + h + i = X101 + 101X + 0100$.

Caso 2: Minimizar número de produtos E número de literais nos produtos

			f_α		f_β		
			4	13	4	5	13
	α	(b) X100	$\surd$				
	α	(c) 110X		$\surd$			
	β	(d) 010X			$\surd$	$\surd$	
	β	(e) X101				$\surd$	$\surd$
	$\alpha\beta$	(i) 0100	$\surd$		$\surd$		
	$\alpha\beta$	(k) 1101		$\surd$			$\surd$

- a tabela acima é cíclica. Não podemos eliminar a linha (b) nem a (c) pois, embora elas sejam dominadas respectivamente pelas linhas (i) e (k), o custo dessas últimas é maior.
- devemos aplicar o método de Petrick

  $$(b+i)(c+k)(d+i)(d+e)(e+k) = 1$$

  que resulta em

  $$cei + bcde + eik + dik + bdk$$

  Desses, os de menor custo são *cei* e *bdk*
- Para cada função, selecionar o menor subconjunto que a cobre.

RESULTADO (ao selecionarmos *cei*)

$f_\alpha : c + g + h + i = 110X + X010 + 101X + 0100$
$f_\beta : e + h + i = X101 + 101X + 0100$
$f_\gamma : f + g + h + j = 00X1 + X010 + 101X + 1100$

Comparando os casos 1 e 2 acima, em ambos precisamos de 7 produtos. Há, no entanto, diferença no número de produtos na função f_α. Em PLA, a porta OU dessa função terá uma entrada a mais que no caso não-PLA.

Exemplo 6: minimizar as funções
$f_1(a,b,c,d) = \sum m(3,4,5,7,9,13,15) + d(11,14)$
$f_2(a,b,c,d) = \sum m(3,4,7,9,13,14) + d(0,1,5,15)$

Os implicantes primos são mostrados na figura 1.25.

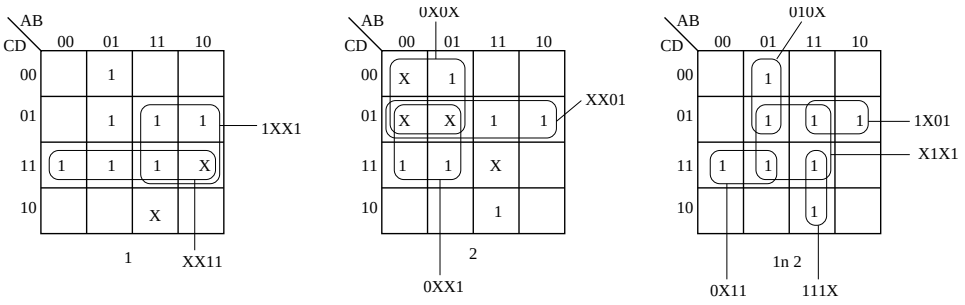


Figura 1.25: Implicantes primos da função $\mathbf{f} = (f_1, f_2)$.

Uma cobertura mínima pode ser encontrada com o auxílio da Tabela de Implicantes Primos. Os implicantes essenciais são $111X$ e $010X$.

			f_1							f_2					
			3	4	5	7	9	13	15	3	4	7	9	13	14
	1	1XX1					√	√	√						
	1	XX11	√			√			√						
	2	XX01											√	√	
	2	0XX1								√		√			
	2	0X0X									√				
*	1,2	111X							√						√
	1,2	0X11	√			√				√		√			
	1,2	1X01					√	√					√	√	
	1,2	X1X1			√	√		√	√			√		√	
*	1,2	010X		√	√						√				
				√	√				√		√				√

Eliminando os essenciais e colunas, e as linhas vazias, temos a seguinte tabela reduzida. Se considerarmos implementação em PLA, pdemos eliminar a linha dos implicantes $XX11$ e $0XX1$ pois estes são dominados pela linha do implicante $0X11$. Similarmente, podemos eliminar as linhas dos implicantes $1XX1$ e $XX01$.

			f_1				f_2			
			3	7	9	13	3	7	9	13
	1	1XX1			√	√				
	1	XX11	√	√						
	2	XX01							√	√
	2	0XX1					√	√		
	1,2	0X11	√	√			√	√		
	1,2	1X01			√	√			√	√
	1,2	X1X1		√		√		√		√

A tabela resultante é a seguinte. Escolhendo-se os secundariamente essenciais obtém-se uma cobertura para todos os mintermos restantes na tabela.

			f_1				f_2			
			3	7	9	13	3	7	9	13
**	1,2	0X11	√	√			√	√		
**	1,2	1X01			√	√			√	√
	1,2	X1X1		√		√		√		√
			√	√	√	√	√	√	√	√

Assim, a solução final é:
$f_1 : 0X11, 1X01, 111X, 010X$
$f_2 : 0X11, 1X01, 111X, 010X$
Ou seja, $f_1 = f_2$! (Por que isso aconteceu !??)

Exemplo 7: minimizar as funções
$f_1(a,b,c,d) = \sum m(0,2,7,10) + d(12,15)$
$f_2(a,b,c,d) = \sum m(2,4,5) + d(6,7,8,10)$
$f_3(a,b,c,d) = \sum m(2,7,8) + d(0,5,13)$

	f_1	f_2	f_3	IP
0000	1		1	√
0010	1	1	1	m
0100		1		√
1000		1	1	l
0101		1	1	√
0110		1		√
1010	1	1		√
1100	1			k
0111	1	1	1	j
1101			1	√
1111	1			√

	f_1	f_2	f_3	IP
00X0	1		1	i
X000			1	h
0X10		1		g
X010	1	1		f
010X		1		√
01X0		1		√
10X0		1		e
01X1		1	1	d
X101			1	c
011X		1		√
X111	1			b

	f_1	f_2	f_3	IP
01XX		1		a

Tabela 1.5: Método tabular para cálculo dos implicantes primos de $\mathbf{f} = (f_1, f_2, f_3)$.

O cálculo dos implicantes pelo método tabular é mostrado na tabela 1.5. Os implicantes 1100, $X101$ e $10X0$ cobrem apenas don't cares e podem ser desconsiderados na etapa de cálculo de cobertura mínima.

Os implicantes podem também ser obtidos pelo mapa de Karnaugh. Veja a figura 1.26.

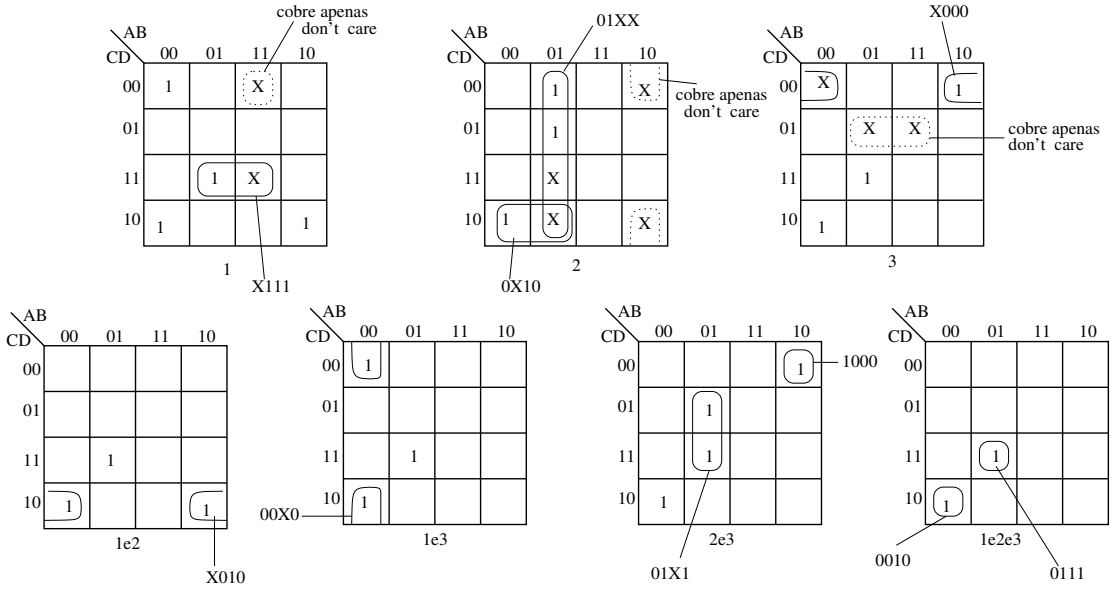


Figura 1.26: Implicantes primos da função $\mathbf{f} = (f_1, f_2, f_3)$.

Uma cobertura mínima pode ser encontrada com o auxílio da Tabela de Implicantes Primos, que é mostrada na seguinte tabela. Nesta mesma tabela, os implicantes essenciais aparecem marcados por *.

			f_1				f_2			f_3		
			0	2	7	10	2	4	5	2	7	8
*	2	(a) 01XX						√	√			
	1	(b) X111			√							
	23	(d) 01X1							√		√	
*	12	(f) X010		√		√	√					
	2	(g) 0X10					√					
	3	(h) X000										√
*	13	(i) 00X0	√	√						√		
	123	(j) 0111			√						√	
	23	(l) 1000					√					√
	123	(m) 0010		√			√			√		
			√	√		√	√	√	√	√		

Eliminando-se a linha dos essencias e as colunas dos mintermos cobertos por eles, obtém-se a seguinte tabela, onde a linha do implicante 1000 é dominada pela linha do implicante $X000$. Ao se eliminar a linha do implicante 1000, o implicante $X000$ torna-se essencial. Das três linhas que restam, é imediato verificar que a escolha que minimiza o custo é 0111, que será marcada com ***.

			f_1	f_3	
			7	7	8
	1	X111	√		
	23	01X1		√	
**	3	X000			√
***	123	0111	√	√	
	23	1000			√
			√	√	√

⟵ dominada pela linha do $X000$

Portanto obtemos:
f_1: $X010$, $00X0$, 0111
f_2: $01XX$, $X010$, $0111 \Longrightarrow 01XX$, $X010$ (pois 0111 é coberto por $01XX$)
f_3: $00X0$, $X000$, 0111.

Exercícios:

1. Para cada uma das soluções do exemplo 5, desenhe o PLA correspondente. Comente as diferenças.

2. Deseja-se projetar um circuito com quatro entradas e duas saídas e que realiza a adição módulo 4. Por exemplo, $(3+3) mod\, 4 = 2$, etc. Os números a serem adicionados são dados em binário respectivamente por $x_2\, x_1$ e $y_2\, y_1$. A saída também deve ser dada em binário ($z_2\, z_1 = 00$ se a soma é 0, $z_2\, z_1 = 01$ se a soma é 1, etc).

   a) determine uma função na forma SOP canônica para z_1 e para z_2

   b) Simplifique-as individualmente

   c) Simplifique-as em conjunto

3. Código BCD refere-se à codificação de dígitos decimais de 0 a 9 pela respectiva representação binária. Para tanto são necessários 4 bits. As combinações binárias de 0000 a 1001 são utilizadas para codificação e as demais não são utilizadas.

O incremento por 1 do código BCD pode ser definido pela seguinte tabela-verdade:

$abcd$	$wxyz$
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	0000
1010	XXXX
...	...
1111	XXXX

Em outras palavras, pode-se pensar esta tabela-verdade como representando 4 funções (w, x, y, e z) com 4 entradas.

Utilize o programa ESPRESSO para obter a minimização individual e conjunta dessas funções. Compare o custo para implementação desses dois casos.

4. Considere um subtrator para números de dois bits. As entradas $a\,b$ e $c\,d$ definem dois números binários N_1 e N_2 (i.e., $N_1 = ab$ e $N_2 = cd$). Suponha que $N_1 \geq N_2$. As saídas $f\,g$ do circuito corsespondem à diferença $N_1 - N_2$ (i.e., $f\,g = N_1 - N_2$).

   a) Escreva a tabela-verdade para $f\,g$

   b) Encontre a forma SOP minimal de f e de g

   c) Encontre a forma POS minimal de f e de g

   d) Qual das duas formas ((b) ou (c)) é mais econômica?

   e) Minimize f e g em conjunto (forma SOP) para implementação em PLA e compare o resultado obtido com os resultados do item (b).

5. Minimizar para implementar em PLA as funções
   $f_1(a,b,c,d) = \sum m(0,2,6,7,15) + d(8,10,14)$
   $f_2(a,b,c,d) = \sum m(0,1,3,7,15) + d(8,10,14)$

1.4 Outros algoritmos de minimização lógica 2-níveis

Algoritmos de minimização tabular (como o Quine-McCluskey) têm algumas desvantagens do ponto de vista computacional:

- eles listam explicitamente todos os implicantes primos, cuja quantidade pode ser de ordem exponencial no número de variáveis n
- requerem que a função a ser minimizada esteja na forma SOP canônica. Não é raro que, juntamente com os don't cares, o número de produtos canônicos seja da ordem de 2^{n-1}

- a tabela-cíclica pode ser bem grande.

O algoritmo QM é um processo demorado e além disso utiliza muita memória, o que limita sua utilidade na prática para funções com relativamente poucas variáveis (poucas dezenas).

Muito esforço ocorreu nas décadas de 1980 e início da década de 1990 no sentido de se desenvolver programas para minimização de funções com várias variáveis. Um dos motivadores deste esforço são os PLAs, dispositivos programáveis, que permitem a realização dois níveis de múltiplas funções. Os esforços realizados foram no sentido de achar alternativas que não requeressem o cálculo explícito de todos os implicantes primos, formas eficientes para o cálculo de coberturas mínimas e heurísticas eficientes nesses processos.

Hoje em dia, ESPRESSO (desenvolvido por um grupo da Universidade de Berckley [Brayton et al., 1984, McGreer et al., 1993]) é a referência para minimização lógica dois-níveis. ESPRESSO é o resultado de uma série de esforços, realizados principalmente na década de 1980, com o objetivo de contornar as limitações do algoritmo QM. Para maiores detalhes veja [Brayton et al., 1984], capítulo 7 de [Micheli, 1994], seção 6.10 de [Hill and Peterson, 1993]. ESPRESSO não calcula os implicantes primos explicitamente; ele utiliza uma série de heurísticas. Além disso, ele também realiza minimização de múltiplas funções e de funções multi-valoradas (lógica multi-valores). Depois do ESPRESSO, foram propostas outras melhorias (como o uso de BDD — Binary Decision Diagrams) por Coudert e outros [Coudert, 1994, Coudert, 1995], e mais recentemente o BOOM [Hlavicka and Fiser, 2001, Fišer and Hlavicka, 2003].

Referências Bibliográficas

[Brayton et al., 1984] Brayton, R. K., Hachtel, G. D., McMullen, C. T., and Sangiovanni-Vincentelli, A. L. (1984). *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers.

[Coudert, 1994] Coudert, O. (1994). Two-level Logic Minimization: an Overview. *Integration, the VLSI Journal*, 17(2):97–140.

[Coudert, 1995] Coudert, O. (1995). Doing Two-level Minimization 100 Times Faster. In *Proc. of Symposium on Discrete Algorithms (SODA)*, San Francisco CA.

[Fišer and Hlavicka, 2003] Fišer, P. and Hlavicka, J. (2003). Boom - a heuristic boolean minimizer. *Computers and Informatics*, 22(1):19–51.

[Hill and Peterson, 1993] Hill, F. J. and Peterson, G. R. (1993). *Computer Aided Logical Design with Emphasis on VLSI*. John Wiley & Sons, fourth edition.

[Hlavicka and Fiser, 2001] Hlavicka, J. and Fiser, P. (2001). BOOM - A Heuristic Boolean Minimizer. In *Proc. of ICCAD*, pages 439–442.

[McGreer et al., 1993] McGreer, P. C., Sanghavi, J., Brayton, R. K., and Sangiovanni-Vincentelli, A. L. (1993). Espresso-Signature : A New Exact Minimizer for Logic Functions. *IEEE trans. on VLSI*, 1(4):432–440.

[Micheli, 1994] Micheli, G. D. (1994). *Synthesis and Optimization of Digital Circuits*. McGraw-Hill.

Última revisão em 27 de março de 2007.