6 Minimização de múltiplas funções

6.1 PLA

A minimização lógica dois-níveis ganhou impulso 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.

A figura 1 mostra um modelo lógico básico de um PLA típico, com 3 variáveis de entrada e três saídas. Os círculos sobre o cruzamento das linhas indicam onde há conexão. No exemplo, as portas lógicas E realizam as funções (produtos) $a\,\overline{b}$, $\overline{a}\,b\,\overline{c}$, $\overline{b}\,c$ e $a\,\overline{c}$, respectivamente; as portas lógicas OU realizam 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 $f_3(a,b,c) = \overline{a}\,b\,\overline{c} + a\,\overline{c}$.

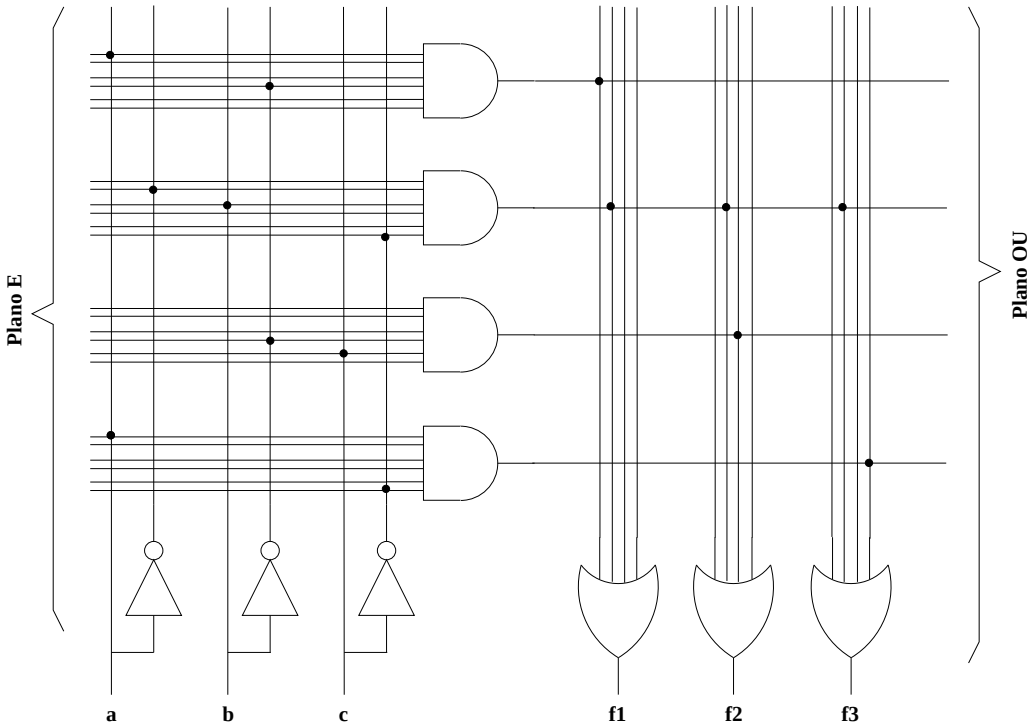

Figura 1: Esquema lógico de um PLA.

Para não sobrecarregar o diagrama, em geral desenha-se de forma simplificada como o mostrado na figura 2.

PLAs comerciais têm tipicamente entre 10 e 20 entradas, entre 30 e 60 portas E e entre 10 e 20 portas OU. Portanto, 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.

6.2 Minimização dois-níveis de múltiplas funções

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:

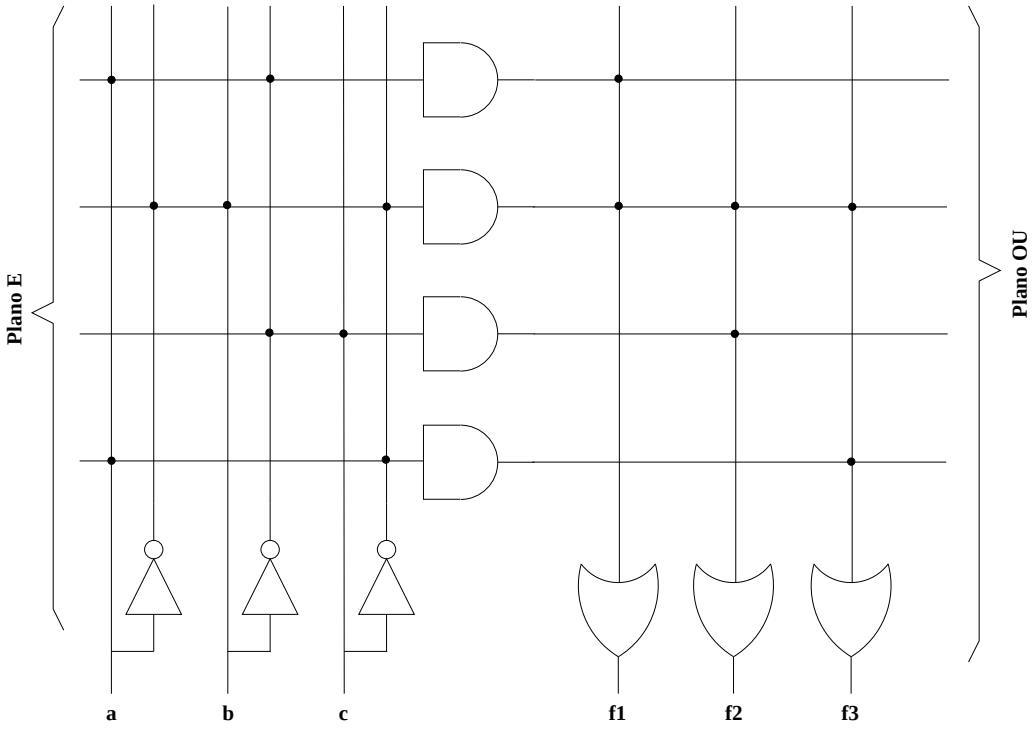

Figura 2: Esquema simplificado de um PLA.

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 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 6 portas (quatro E e duas OU).

Note, porém, que $f_1(x_1\,x_2\,x_3) = \overline{x}_1\,\overline{x}_2 + \overline{x}_2\,x_3 = \overline{x}_1\,\overline{x}_2 + x_1\,\overline{x}_2\,x_3$ e $f_2(x_1\,x_2\,x_3) = x_1\,x_3 + x_1\,x_2 = x_1\,x_2 + x_1\,\overline{x}_2\,x_3$, ou seja, há um produto comum às duas funções. Neste caso, para implementá-las são necessárias 5 portas (três E e duas OU), pois uma das portas E é compartilhada pelas duas funcões.

Exemplo 2: Considere as funções (já minimizadas) :

$f_0 = a$
$f_1 = \overline{b}$
$f_2 = bc + ab$
$f_3 = \overline{a}\,\overline{b} + \overline{a}\,c$
Para implementá-las, são necessárias 6 portas lógicas 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.

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 + α (número total de entradas para as portas E) + β(número total de entradas para as portas OU).

com $0 \leq \alpha, \beta < 1$ (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, portas E e portas OU, a área do chip também é fixa. Portanto, reduzir o número de entradas para as portas E ou reduzir o número de entradas para as portas OU não tem impacto no custo. Assim, na minimização de funções com o objetivo de implementação em PLA é razoável considerarmos $\alpha = \beta = 0$.

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 $\alpha = \beta = 0$ 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. Dadas k funções, que podem ser representadas por um vetor $\mathbf{f} = (f_1, f_2, \ldots, f_k)$, dizemos que um produto p de k' ($k' <= k$) dessas funções é implicante primo se ele é implicante primo de uma delas (no sentido visto até agora) ou se não existe nenhum outro produto dessas k' funções que o cobre.

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

Exemplo 3: Considere as funções do exemplo 1, escritas na forma SOP canônica $f_1(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2x_3$ e $f_2(x_1, x_2, x_3) = x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$.

A tabela 1 mostra o cálculo dos implicantes primos de $\mathbf{f} = (f_1, f_2)$. Note que 101, que é um produto de ambas as funções, é implicante primo pois não existe um produto que o cobre e que seja também produto de ambas as funções.

$x_1x_2x_3$	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: Método tabular para cálculo dos implicantes primos de $\mathbf{f} = (f_1, f_2)$.

A tabela 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. Na tabela da esquerda são identificados os dois primos essenciais (00X e 11X). Após a redução da tabela, obtemos a tabela da direita.

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

			f_1	f_2
			101	101
	1	(b) X01	$\surd$	
	2	(c) 1X1		$\surd$
	1,2	(e) 101	$\surd$	$\surd$

Tabela 2: Tabela de implicantes primos de $\mathbf{f} = (f_1, f_2)$.

Se levarmos em consideração apenas a minimização do número de produtos, podemos dizer que o primo (e) domina os primos (b) e (c). Portanto, as linhas (b) e (c) podem ser eliminadas, resultando apenas a linha (c). 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.

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

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

Escrevedo 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 (o mesmo do custo mais simples).

Exercício: Desenhe o PLA que realiza as duas funções minimizadas do exemplo 3.

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

Exemplo 4: 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 método tabular para determinação dos implicantes primos é mostrado na tabela 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 3: Método tabular para cálculo dos implicantes primos de $\mathbf{f} = (f_\alpha, f_\beta, f_\gamma)$.

O cálculo de cobertura mínima é mostrado a seguir. Primeiramente são identificados os primos 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						√					√						
			√		√	√	√				√	√		√	√	√	√	√	√

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

Caso 1: Minimizar APENAS o número de 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		$\surd$			$\surd$

- o número de literais presentes nos IP não interessa: 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)

$f_\alpha = g + h + i + j + k$
$f_\beta = d + h + i + k$
$f_\gamma = f + g + h + j$

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
- 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*)

$$f_\alpha = c + g + h + i$$
$$f_\beta = e + h + i$$
$$f_\gamma = f + g + h + j$$

Exercício: Para cada uma das soluções do exemplo 4, desenhe o PLA correspondente. Comente as diferenças.

6.3 O algoritmo ESPRESSO

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 é 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.

ESPRESSO [Brayton et al., 1984] é o resultado de uma série de esforços, realizados principalmente na década de 1980, com o objetivo de contornar as limitações do algoritmo QM. Hoje em dia, ESPRESSO é a referência para minimização lógica dois-níveis. Para maiores detalhes veja [Brayton et al., 1984], capítulo 7 de [Micheli, 1994], seção 6.10 de [Hill and Peterson, 1993] (deste último há cópia disponível na seção de reprografia do bloco B).

Na década de 1990, técnicas que não armazenam os implicantes primos explicitamente foram desenvolvidas (trabalhos de Coudert e Madre) [Coudert and Madre, 1992, Coudert, 1994] e mais recentemente apareceu um algoritmo denominado BOOM [Hlavicka and Fiser, 2001]. Aqueles que tiverem interesse, consultem as referências indicadas ou venham falar comigo.

Referências

[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 and Madre, 1992] Coudert, O. and Madre, J. (1992). Implicit and Incremental Computation of Primes and Essential Primes of Boolean Functions. In *Proc. of 29th DAC*, Anaheim, CA, USA.

[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.

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