Pequena Revisão

Os principais tópicos abordados até esta parte do curso estão resumidos a seguir.

Álgebra Booleana: $\langle A, +, \cdot, \bar{}, 0, 1\rangle$ é uma álgebra Booleana se valem as propriedades:
- A1: comutativa
- A2: distributiva
- A3: existência de elemento identidade com relação a + e a ·
- A4: existência de complemento

Exemplos de álgebras booleanas:
- álgebra dos conjuntos
- lógica proposicional
- circuitos de chaveamento

Expressões booleanas:

- Em qualquer álgebra booleana $\langle A, +, \cdot, \bar{}, 0, 1\rangle$ existem exatamente 2^{2^n} funções booleanas com n variáveis. Nem todas as funções de A^n em A são funções booleanas. No caso da álgebra booleana $\langle B, +, \cdot, \bar{}, 0, 1\rangle$ (onde $B = \{0, 1\}$), todas as funções de B^n em B são funções booleanas.
- Uma função pode ser expressa por (infinitas) diferentes expressões. Expressões que representam uma mesma função são ditas **equivalentes**.
- Algumas formas de mostrar equivalência de expressões
  - diagramas de Venn (no caso de conjuntos)
  - tabelas-verdade
  - manipulação algébrica
- Formas canônicas (A representação em ambas as formas canônicas é única)
  - SOP (soma de produtos) canônica
  - POS (produto de somas) canônica
- Conjunto de operadores funcionalmente completos • $\{+, \cdot, \bar{}\}$
  - $\{+, \bar{}\}$
  - $\{\cdot, \bar{}\}$
  - $\{\neg, \rightarrow\}$
  - NÃO-E (Barra de Sheffer |)
  - NÃO-OU (Negação conjunta ↓)
- A relação $\leq$ definida em uma álgebra booleana $\langle A, +, \cdot, \bar{}, 0, 1\rangle$ por $x \leq y \iff x + y = y$ é uma relação de ordem parcial. O conjunto $(A, \leq)$ é um reticulado. Toda álgebra booleana é um reticulado.
- Um reticulado booleano (distributivo e complementado) é uma álgebra booleana.
- Qualquer elemento de um reticulado pode ser expresso de forma única como supremo de átomos.
- O conjunto de funções booleanas com n variáveis é também uma álgebra booleana; cada função pode ser expressa como soma de produtos canônicos.
- Duas álgebras booleanas com um mesmo número de átomos são isomorfas.

4 Teoria de chaveamentos e aplicações no projeto lógico de circuitos digitais

OBS: não há notas de aula para esta parte.

Circuitos de chaveamento: aula do dia 03/04. Ver, por exemplo, capítulo 4 de [Mendelson, 1977].

- circuitos em série, circuitos em paralelo, circuitos em série-paralelo, circuitos-ponte
- qualquer expressão booleana (com complementação apenas de variáveis e não de subexpressões compostas) pode ser representada por um circuito em série-paralelo.
- qualquer circuito de chaveamento corresponde a uma função booleana

Circuitos lógicos: aula dos dias 08 e 10/04. Ver, por exemplo, capítulo 4 de [Mendelson, 1977].
- circuitos 2-níveis
- circuitos multi-níveis
- circuitos com múltiplas saídas

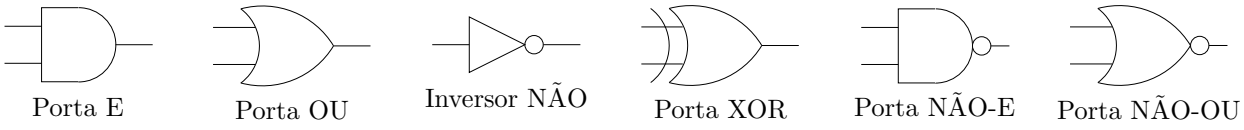

Figura 1: Representação gráfica de algumas portas lógicas

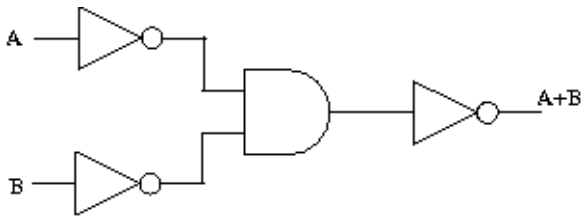

Figura 2: Realização do operador OR com AND e NOT.

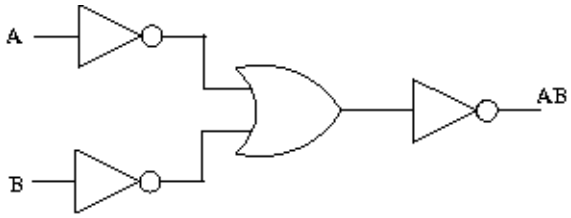

Figura 3: Realização do operador AND com OR e NOT.

Sistema de Numeração Binária

Supõe-se que todos dominam este tópico (pois assim se manifestaram em sala de aula). Logo, este tópico não será dado em sala de aula. Para os que tiverem interesse, há material disponível na seção de reprografia do Bloco B (vulgo Xerox).

5 Minimização lógica dois-níveis

Objetivo: encontrar expressões mínimas na forma soma de produtos ou produto de somas, que podem ser implementados em circuitos com dois níveis de portas lógicas.

Referências para esta parte: capítulo 6 de [Hill and Peterson, 1981], capítulo 7 de [Micheli, 1994], capítulo 4 de [Mendelson, 1977], etc.

5.1 Pequena revisão

Seja $\langle A, +, \cdot, \bar{}, 0, 1\rangle$ uma álgebra booleana e sejam n variáveis $x_1, x_2, \ldots, x_n$ que tomam valores em A. Uma *expressão booleana* em A, nas variáveis $x_1, x_2, \ldots, x_n$, é definida indutivamente como:

1. os elementos em A são expressões booleanas;
2. as variáveis $x_1, x_2, \ldots, x_n$ são expressões booleanas;
3. se E_1 e E_2 são expressões booleanas em A, então $(E_1) + (E_2)$, $(E_1) \cdot (E_2)$ e $\overline{E}_1$ são também expressões booleanas (em geral o operador $\cdot$ e os parênteses são omitidos)
4. Todas as expressões booleanas são geradas aplicando-se as regras 1, 2 e 3 um número finito de vezes.

Qualquer expressão construída de acordo com as regras acima define uma função de A^n em A. Para avaliar a função em $(a_1, a_2, \ldots, a_n) \in A^n$, cada ocorrência de x_i na expressão deve ser trocada por a_i (para todo $i \in \{0, \ldots, n\}$) e o seu valor calculado. Duas expressões são **equivalentes** se as funções correspondentes são a mesma. Expressões booleanas consistindo de uma variável simples ou seu complemento, tais como x_1 ou $\overline{x}_2$, são chamadas de **literais**.

Dada uma álgebra booleana $\langle A, +, \cdot, \bar{}, 0, 1\rangle$ e um conjunto de n variáveis $x_1, x_2, \ldots, x_n$, uma **função booleana** nas n-variáveis em A é uma função $f : A^n \to A$ que pode ser expressa com uma expressão booleana em A. Se $(a_1, a_2, \ldots, a_n) \in A^n$, escrevemos $f(a_1, a_2, \ldots, a_n)$ para o valor $f((a_1, a_2, \ldots, a_n))$.

Seja $A(n)$ o conjunto de todas as funções booleanas em A com n variáveis e seja $\leq$ uma relação definida em $A(n)$ da seguinte forma:

$$f \leq g \iff f(\mathbf{a}) \leq g(\mathbf{a}),\ \forall \mathbf{a} \in A^n\,.$$

Seja $(f \cdot g)(\mathbf{a}) = f(\mathbf{a}) \cdot g(\mathbf{a})$, $(f + g)(\mathbf{a}) = f(\mathbf{a}) + g(\mathbf{a})$, e $\overline{f}(\mathbf{a}) = \overline{f(\mathbf{a})}$, $\forall \mathbf{a} \in A^n$. Fazendo $\mathbf{0}(\mathbf{a}) = 0$ e $\mathbf{1}(\mathbf{a}) = 1$ para todo $\mathbf{a} \in A^n$, o conjunto $(A(n), +, \cdot, \bar{}, \mathbf{0}, \mathbf{1})$ também é uma álgebra booleana.

A cardinalidade de $A(n)$ é $2^{(2^n)}$. Nem todos os mapeamentos de A^n em A são funções booleanas de acordo com a definição acima, i.e., há mapeamantos que não podem ser expressos por uma expressão em A. Se $A = \{0, 1\}$, então qualquer expressão construída de acordo com as regras acima define uma função de $\{0, 1\}^n$ em $\{0, 1\}$. Por outro lado, qualquer função de $\{0, 1\}^n$ em $\{0, 1\}$ pode ser representado por uma expressão em $\{0, 1\}$.

NOTA: Os dados nos computadores digitais são representados em binário e os processamentos nada mais são do que mapeamentos de dados binários em dados binários, ou seja, funções que mapeiam n dígitos binários em m dígitos binários.

Daqui em diante consideremos a álgebra booleana $\langle B(n), +, \cdot, \bar{}, \mathbf{0}, \mathbf{1}\rangle$.

Uma função booleana $f : \{0,1\}^n \to \{0,1\}$ pode ser caracterizada em termos de seu conjunto-um, $f\langle 1\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 1\}$, bem como em termos do seu conjunto-zero, $f\langle 0\rangle = \{\mathbf{b} \in \{0,1\}^n : f(\mathbf{b}) = 0\}$.

As funções booleanas que tomam valor 1 em exatamente um elemento de $\{0,1\}^n$ são denominados **mintermos**. Um mintermo em n variáveis é uma conjunção (produto) de exatamente n literais, os quais envolvem diferentes variáveis. Uma conjunção em que uma variável aparece no máximo uma vez é usualmente denominada de **produto**, e expresso como $p = \prod_{i=1}^{n} \sigma_i$, $\sigma_i \in \{x_i, \overline{x}_i, \text{' '}\}$, com ' ' denotando o caracter vazio. Por exemplo, para $n = 4$, x_1x_3 e $x_2\overline{x}_3\overline{x}_4$ são exemplos de produtos. Mintermos (aqueles em que todas as variáveis aparecem exatamente uma vez, ou na forma barrada ou na forma não-barrada) são também chamados de **produtos canônicos**.

Os átomos de $\langle B(n), +, \cdot, \bar{}, \mathbf{0}, \mathbf{1}\rangle$ são os 2^n mintermos. Portanto, toda função booleana pode ser escrita como uma disjunção (soma) de mintermos distintos. Mais ainda, tal representação é única a menos da ordem dos mintermos e será denominada **soma canônica de produtos** (SOP canônica).

5.2 Simplificação de notação

O conjunto de todas as seqüências de n bits correspondem à representação binária dos números entre 0 e $2^n - 1$. Com base neste fato, podemos caracterizar os produtos canônicos, e as somas canônicas (que serão denominadas **maxtermos**) através da notação decimal correspondente. A tabela 1 apresenta os mintermos e maxtermos para três variáveis e a notação associada a cada um deles.

$x_1x_2x_3$	maxtermos	mintermos
0 0 0	$x_1 + x_2 + x_3 = M_0$	$\overline{x}_1\overline{x}_2\overline{x}_3 = m_0$
0 0 1	$x_1 + x_2 + \overline{x}_3 = M_1$	$\overline{x}_1\overline{x}_2x_3 = m_1$
0 1 0	$x_1 + \overline{x}_2 + x_3 = M_2$	$\overline{x}_1x_2\overline{x}_3 = m_2$
0 1 1	$x_1 + \overline{x}_2 + \overline{x}_3 = M_3$	$\overline{x}_1x_2x_3 = m_3$
1 0 0	$\overline{x}_1 + x_2 + x_3 = M_4$	$x_1\overline{x}_2\overline{x}_3 = m_4$
1 0 1	$\overline{x}_1 + x_2 + \overline{x}_3 = M_5$	$x_1\overline{x}_2x_3 = m_5$
1 1 0	$\overline{x}_1 + \overline{x}_2 + x_3 = M_6$	$x_1\overline{x}_2x_3 = m_6$
1 1 1	$\overline{x}_1 + \overline{x}_2 + \overline{x}_3 = M_7$	$x_1x_2x_3 = m_7$

Tabela 1: Tabela de maxtermos e mintermos

Observe que $x_1\overline{x}_2x_3 = 1$ se e somente se $x_1 = 1, x_2 = 0$ e $x_3 = 1$. Portanto, a SOP canônica de uma expressão booleana pode ser diretamente obtida através da soma dos mintermos correspondentes aos 1's da sua tabela-verdade. Analogamente, $(x_1 + \overline{x}_2 + x_3) = 0$ se e somente se $x_1 = 0, x_2 = 1$ e $x_3 = 0$, e portanto, a POS canônica pode ser obtida pelo produto dos maxtermos correspondentes aos 0's da tabela-verdade.

Exemplo: Dada a tabela-verdade

$x_1x_2x_3$	f_1
000	1
001	1
010	0
011	0
100	1
101	1
110	1
111	0

a forma SOP canônica da função é

$$f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1\overline{x}_2x_3$$

e sua notação simplificada é dada por :

$$f(x_1, x_2, x_3) = \sum m(0, 1, 4, 5, 6).$$

A forma POS canônica é $f(x_1, x_2, x_3) = (x_1 + \overline{x}_2 + x_3)(x_1 + \overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) = \prod M(2, 3, 7)$.

5.3 Minimização de funções booleanas

Quando pensamos em minimização, devemos estabelecer um critério de custo que desejamos minimizar. O seguinte critério é bastante utilizado.

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

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', $p \leq p'$, tal que $p' \leq f$.

Dada uma expressão minimal, se um literal de qualquer um dos produtos da expressão é removida, a expressão resultante não mais representa a mesma função.

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

5.3.1 Representação cúbica (ou via intervalos)

Um conceito utilizado com bastante freqüência no contexto de manipulação de funções booleanas é a de **cubos**.

As seqüências de n bits, correspondentes a todas as possíveis combinações de valores que as n variáveis de uma função Booleana podem tomar, podem ser representadas por pontos no n-espaço. A coleção de todos os 2^n pontos possíveis formam os vértices de um ***n*-cubo**. A figura 4 mostra um 3-cubo.

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.

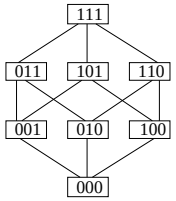

Figura 4: Diagrama do 3-cubo.

Mais formalmente, 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 é o conjunto $p\langle 1\rangle = \{\mathbf{b} \in \{0,1\}^n : p(\mathbf{b}) = 1\}$. Portanto, existem tantos cubos contidos em $\{0,1\}^n$ quanto produtos envolvendo as variáveis $x_1, x_2, \ldots, x_n$.

Cubos não são subconjuntos arbitrários de $\{0,1\}^n$. No contexto de reticulados, cubos são sub-reticulados do reticulado $\{0,1\}^n$, denominados **intervalos**.

Um intervalo é caracterizado por dois extremos: o menor e o mair elementos contidos nele. Assim, 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 colocando um X nas coordenadas que não são iguais. Assim, o 1-cubo $\{000, 100\}$, que corresponde ao intervalo $[000, 100]$, é representado por $X00$. O 2-cubo $\{000, 001, 100, 101\}$, que corresponde ao intervalo $[000, 101]$, é representado por $X0X$.

NOTA: Daqui em diante utilizaremos arbitrariamente 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 convensão de desenhar elementos de $f\langle 1\rangle$ como círculos preenchidos, enquanto os elementos de $f\langle 0\rangle$ 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 5.

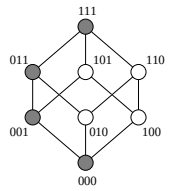

Figura 5: Representação via diagrama de Hasse da função f_1.

Exemplo: A figura 6 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 $X00$.

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 $p\langle 1\rangle \subseteq f\langle 1\rangle$). Neste sentido, mintermos são cubos de tamanho unitário e primos

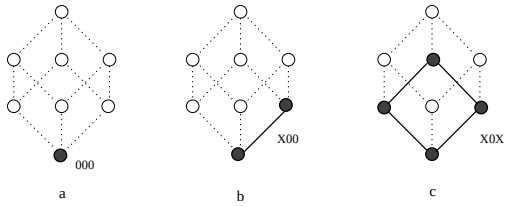

Figura 6: Os cubos 000, $X00$ e $X0X$.

são cubos maximais contidos em $f\langle 1\rangle$ (i.e., um cubo de f que não é totalmente contido em outro cubo de f).

Os dois primeiros cubos da Fig. 7 (em negito) são cubos da função f_1 mas os dois últimos não são.

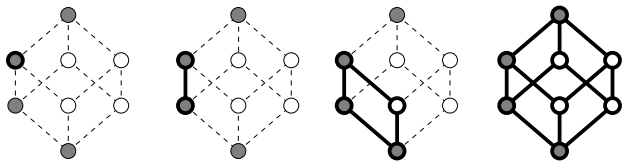

Figura 7: Examplos de um 0-cubo (intervalo 011), um 1-cubo (intervalo $0X1$), um 2-cubo (intervalo $0XX$) e um 3-cubo (intervalo XXX), respectivamente (em negrito).

Exemplo: A função Booleana $f = \sum m(0, 1, 4, 5, 6)$ é representada pelos vértices 000, 001, 100, 101 e 110. Os mintermos (ou intervalos triviais) de f e os implicantes primos (ou cubos ou intervalos maximais) de f são mostrados respectivamente nas figura 8a e 8b.

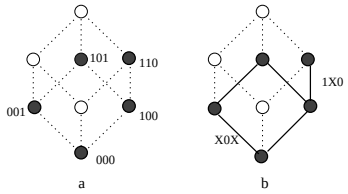

Figura 8: (a) Mintermos e (b) implicantes primos de $f = \sum m(0, 1, 4, 5, 6)$.

A forma SOP canônica de f pode ser entendida como sendo a coleção de 0-cubos de f ou, equivalentemente, como a coleção de intervalos triviais (intervalos com apenas um elemento), cada um deles correspondendo a um elemento em $f\langle 1\rangle$. Em geral, qualquer expressão na forma SOP corresponde a uma coleção de cubos (ou intervalos) onde nenhum deles está propriamente contido em outro.

5.4 Minimização Tabular de Quine-McCluskey

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. O algoritmo tabular de Quine-McCluskey para minimização de funções Booleanas é um método clássico que sistematiza este processo algébrico.

O algoritmo de Quine-McCluskey (QM) requer que a função booleana a ser minimizada esteja na forma SOP canônica. A idéia básica deste algoritmo 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 considerados). 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**.

Este processo de minimização 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 9.

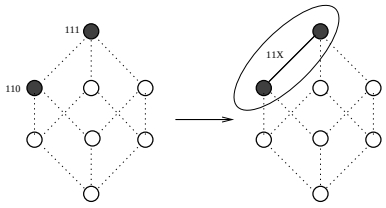

Figura 9: Passo elementar do algoritmo de Quine-McCluskey

5.4.1 Cálculo de implicantes primos

A primeira parte do algoritmo QM consiste de um processo para determinação de todos os implicantes primos. A seguir descrevemos os processos que constituem esta parte e, ao mesmo tempo, mostramos a sua aplicação sobre a 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

- 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 aqueles que não foram combinados com nenhum outro, ou seja, aqueles que não estão com a marca $\surd$.

  $1X0$ e $X0X$

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 $f(a,b,c) = \sum m(0,1,3,7)$. Pelo algoritmo QM obtemos os seguintes implicantes primos:

$00X$, $0X1$ e $X11$.

Graficamente, estes implicantes primos (ou cubos) correspondem respectivamente aos intervalos $[000, 001]$, $[001, 011]$ e $[011, 111]$ ilustrados na figura 10(a). Note, porém, que o intervalo $[001, 011]$ é redundante,

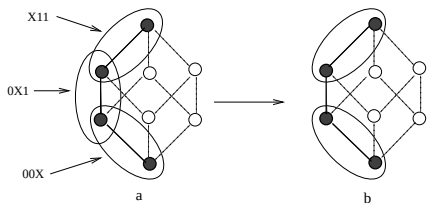

Figura 10: Os (a) implicantes primos e uma (b) cobertura mínima .

ou seja, a mesma expressão pode ser expressa apenas pelos implicantes primos $00X$ e $X11$ (figura 10(b)).

Portanto, o ponto central da segunda parte do algoritmo QM 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**.

5.4.2 Cálculo de uma cobertura mínima

Uma cobertura mínima pode ser calculada com a ajuda de uma tabela denominada Tabela de Implicantes Primos, conforme descrito a seguir (com exemplos para para a 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 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. 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 estas 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.

		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											$\surd$	$\surd$			
			$\surd$	$\surd$			$\surd$	$\surd$	$\surd$	$\surd$					$\surd$	$\surd$

Deve-se repetir este processo enquanto existir uma coluna cujo mintermo não está coberto pelo conjunto de implicantes primos selecionados até o momento e que satisfaz as condições descritas.

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

		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 : manter apenas as colunas correspondentes aos mintermos não cobertos pelos implicantes primos essenciais, e as linhas correspondentes aos implicantes primos que não foram selecionados e que cobrem pelo menos um mintermo ainda não coberto.

		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.

   Eliminar colunas dominantes: Diz-se que uma coluna β na Tabela de Implicantes Primos domina uma coluna α se e somente se β tem um X em todas as linhas em que α tem X. Se β domina α, então ela pode ser removida da tabela.

   Eliminar linhas dominadas: Diz-se que uma linha A na Tabela de Implicantes Primos domina uma outra linha B se e somente se o número de literais em A é menor que o de B, e A tem X em pelo menos todas as colunas em que B tem X. Se A domina B, então existe uma forma minimal que não utiliza o termo da linha B, ou seja, a linha B pode ser removida da tabela.

   A seguir, deve-se repetir o mesmo processo do passo 2, porém os implicantes primos essenciais serão chamados secundariamente essenciais e marcados com dois asteriscos (**), e em seguida fazer a redução da tabela (passo 3).

		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. Aplicar o método de Petrick : este método (de busca exaustiva) pode ser utilizado para resolver a tabela cíclica. Ele fornece todas as possíveis combinações dos implicantes primos restantes que são suficientes para cobrir os mintermos restantes. Deve-se escolher dentre elas, aquela 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 $\{a, c, d, g\}$ (outros tem custo maior).

5.4.3 Pontos fracos do algoritmo QM

- o algoritmo requer que a função esteja na forma SOP canônica
- o número de implicantes primos de uma função booleana com n variáveis pode ser da ordem de $3^n/n$ [Brayton et al., 1984]. Um dos pontos fracos do algoritmo de QM é que ele calcula todos os implicantes primos explicitamente.

5.5 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 pelo algoritmo QM uma função incompletamente especificada, é interessante considerarmos todos os don't cares na etapa de cálculo dos implicantes primo, pois isto aumenta a chance de eliminarmos a quantidadde de produtos.

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 $f\langle 1\rangle$.

5.6 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 contrói uma parede juntando os tijolos, enquanto o ISI é aquele em que o fulano esculpe uma tora de madeira para obter a obra desejada.

5.6.1 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

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

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,

$$\begin{aligned}[A, B] \setminus [C, D] = \\ \Big\{[A, B'] : B' \text{ é elemento maximal daqueles que não contém nenhum elemento de } [C, D]\Big\} \cup \\ \Big\{[A', B] : A' \text{ é elemento minimal daqueles que não estão contidos em nenhum elemento de } [C, D]\Big\}.\end{aligned} \tag{2}$$

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. 11.

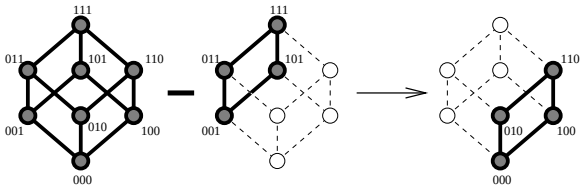

Figura 11: $[000, 111] \setminus [001, 111] = \{[000, 110]\}$.

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. 12.

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. 13.

5.6.2 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 14 mostra os elementos que permanecem após as sucessivas

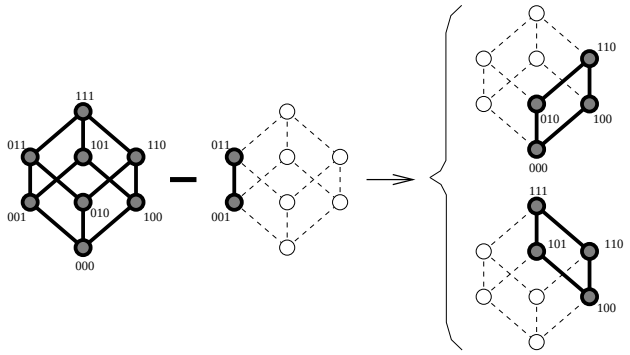

Figura 12: $[000, 111] \setminus [001, 011] = \{[000, 110], [100, 111]\}$.

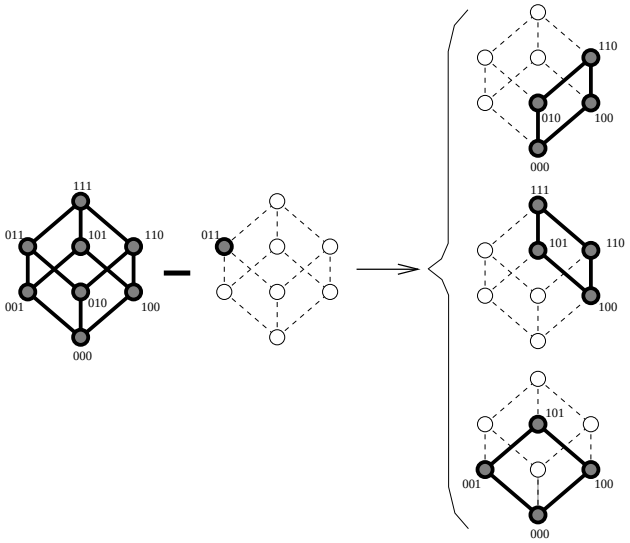

Figura 13: $[000, 111] \setminus [011, 011] = \{[000, 110], [100, 111], [000, 101]\}$.

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 15 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.

5.6.3 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 16 and 17.

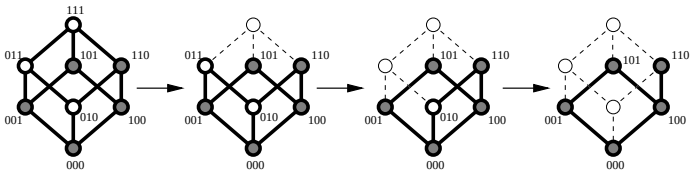

Figura 14: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 011, 111\}$ e $f\langle 1\rangle = \{000, 001, 100, 101, 110\}$). Ordem de remoção: 111, 011 e 010.

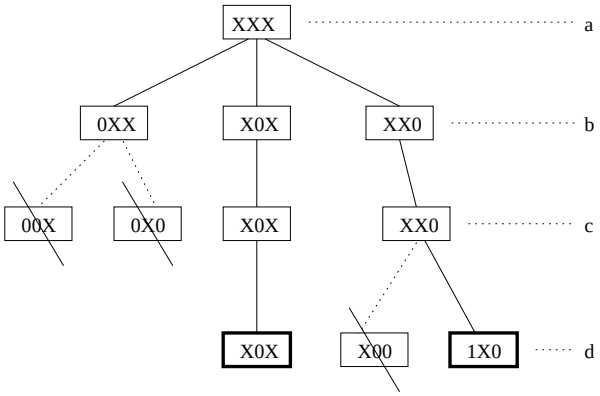

Figura 15: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 011, 111\}$ e $f\langle 1\rangle = \{000, 001, 100, 101, 110\}$). (a) intervalo inicial; (b) após remoção de 111 ; (c) após remoção de 011; (d) resultado final, após remoção de 010.

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 $0XX$, 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.

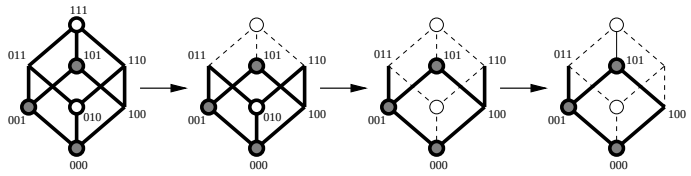

Figura 16: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 111\}$, $f\langle 1\rangle = \{000, 001, 101\}$ e $f\langle *\rangle = \{100, 011, 110\}$).

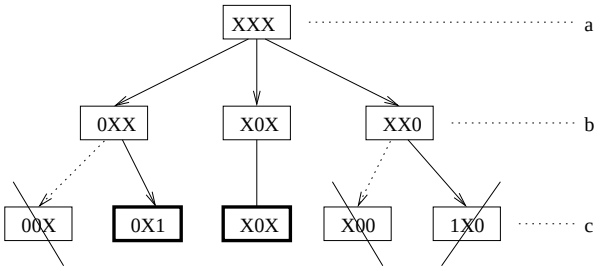

Figura 17: Cálculo de implicantes primos de f ($f\langle 0\rangle = \{010, 111\}$, $f\langle 1\rangle = \{000, 001, 101\}$ e $f\langle *\rangle = \{100, 011, 110\}$).

5.7 Diferença entre QM e ISI

Ambos calculam todos os implicantes primos e depois uma cobertura mínima. No entanto, quando a função possui don't cares, o ISI tem a vantagem de não manipular explicitamente os don't cares.

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.

[Hill and Peterson, 1981] Hill, F. J. and Peterson, G. R. (1981). *Introduction to Switching Theory and Logical Design*. John Wiley, 3rd edition.

[Mendelson, 1977] Mendelson, E. (1977). *Álgebra Booleana e Circuitos de Chaveamento*. Mcgraw-Hill.

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