Sistemas Digitais

Álgebra de Boole Binária e Especificação de Funções

João Paulo Baptista de Carvalho

joao.carvalho@inesc.pt

TÉCNICO LISBOA

inesc id lisboa

Álgebra de Boole Binária

- A Álgebra de Boole binária – através do recurso à utilização de funções booleanas (ou funções lógicas) – é a principal teoria de suporte às metodologias de síntese e análise de circuitos digitais
- Utiliza variáveis binárias, i.e., que só podem assumir um de dois valores: {0,1}; {Low,High}; {True,False}; etc.
- Às variáveis Binárias também se dá o nome de variáveis Lógicas ou Booleanas

TÉCNICO LISBOA

Funções Lógicas de Uma Variável

- Como só estão definidos 2 elementos no universo da Álgebra de Boole binária, o número de funções lógicas é finito, o que potencia uma abordagem algébrica bastante simples
- Veja-se o exemplo para uma única variável:

Tabela de verdade

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

- Só existem 4 funções possíveis!

$f_0(x) = 0$ $\quad f_1(x) = x$ $\quad f_2(x) = \bar{x}$ $\quad f_3(x) = 1$

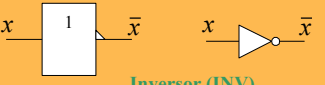
constante 0 — identidade — negação — constante 1

Negação (NOT)

- Das funções apresentadas, a Negação, Complemento, ou NOT, é a mais importante, e caracteriza-se por transformar uma afirmação Verdadeira numa Falsa (e vice-versa)
- Para além da expressão algébrica e da tabela de verdade, a negação pode ser graficamente representada por um dos seguintes símbolos lógicos:

$f(x) = \bar{x}$

NOT



Inversor (INV)

- Dupla Negação: $\bar{\bar{x}} = x$
- Demonstração do teorema da dupla negação por indução completa:

x	$\bar{x}$	$\bar{\bar{x}}$
0	1	0
1	0	1

TÉCNICO LISBOA

Funções de Duas Variáveis

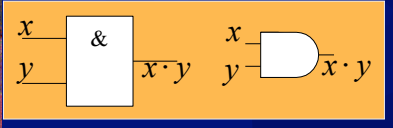
- Existem 16 diferentes funções lógicas de 2 variáveis. As mais importantes são denominadas AND, OR, NAND, NOR e XOR
- Conjunção (AND, Produto Lógico, $\wedge$) e Disjunção (OR, Soma Lógica, $\vee$):

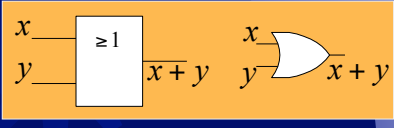
AND

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1





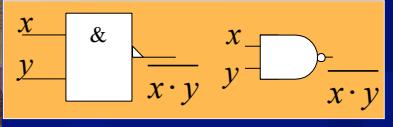
TÉCNICO LISBOA

Funções de Duas Variáveis

- NAND (AND negado), NOR (OR negado) e XOR (OU-Exclusivo):

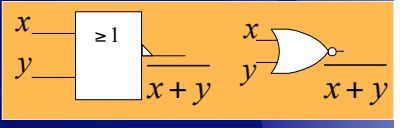
NAND

x	y	$\overline{x.y}$
0	0	1
0	1	1
1	0	1
1	1	0



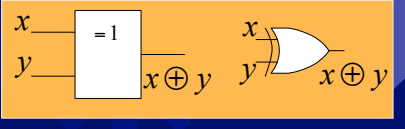
NOR

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0



XOR

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0



TÉCNICO LISBOA

Álgebra de Boole Binária

Uma **Álgebra de Boole binária** é um **sistema algébrico** $B_2 = (A=\{0,1\}, . ,+)$ formado por um **conjunto gerador** A e por duas operações binárias, . , +, designadas por **produto lógico** e **soma lógica**, e por uma operação designada por complemento, tal que:

- **(I)** $\forall_{x,y \in A} (x \cdot y \in A) \wedge (x + y \in A)$ **(Propriedade de Fecho)**

- **(II)** $\forall_{x,y,z \in A}$ verifica-se:

•A1 **(Propriedade Comutativa)**	$x \cdot y = y \cdot x$	$x + y = y + x$
•A2 **(Propriedade Associativa)**	$x + (y + z) = (x + y) + z$	$x.(y.z) = (x.y).z$
•A3 **(Propriedade Distributiva)**	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
•A4 **(Elemento neutro)**	$x \cdot 1 = x$	$x + 0 = x$
•A5 **(Complemento)**	$x \cdot \bar{x} = 0$	$x + \bar{x} = 1$
•A6 **(Idempotência)**	$x \cdot x = x$	$x + x = x$

[Hist.] Boole, George (1815-1864), Matemático britânico. Em 1854, publicou *"An Investigation of the Laws of Thought" onde descreveu um sistema algébrico mais tarde designado por álgebra de Boole*

TÉCNICO LISBOA

Propriedades Básicas da Álgebra de Boole Binária

$$x = x$$

$x + 0 = x$	$x \cdot 1 = x$
$x + 1 = 1$	$x \cdot 0 = 0$
$x + x = x$	$x \cdot x = x$
$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$

$$x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$
$$x \oplus y = (x + y) \cdot (\bar{x} + \bar{y})$$
$$x \oplus 0 = x$$
$$x \oplus 1 = \bar{x}$$ (XOR)
$$\overline{x \oplus y} = \bar{x} \oplus y = x \oplus \bar{y}$$

$x + y = y + x$	$x \cdot y = y \cdot x$	Comutatividade
$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	Associatividade
$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + y \cdot z = (x + y) \cdot (x + z)$	Distributividade
$\overline{x + y} = \bar{x} \cdot \bar{y}$	$\overline{x \cdot y} = \bar{x} + \bar{y}$	DeMorgan
$x \cdot y + x \cdot \bar{y} = x$	$(x + y) \cdot (x + \bar{y}) = x$	Adjacência

TÉCNICO LISBOA

Princípio da Dualidade

- **Qualquer expressão válida numa álgebra de Boole tem uma expressão dual, também válida nessa álgebra**, que se obtém por troca do símbolo operatório '**+**' com o símbolo operatório '**.**', e por troca do par de valores **0** e **1**
  - **Exemplo:**

$$x + 0 = x \quad x \cdot 1 = x$$

TÉCNICO LISBOA

Mais teoremas...

- **Absorção:**

$$x \cdot (x + y) = x \qquad x + x \cdot y = x$$

- **Redundância:**

$$x + \bar{x} \cdot y = x + y \qquad x \cdot (\bar{x} + y) = x \cdot y$$

- **Consenso:**

$$x \cdot y + y \cdot z + \bar{x} \cdot z = x \cdot y + \bar{x} \cdot z$$
$$(x + y) \cdot (y + z) \cdot (\bar{x} + z) = (x + y) \cdot (\bar{x} + z)$$

TÉCNICO LISBOA

Leis de Morgan:

$$\overline{x + y} = \overline{x}.\overline{y}$$

$$\overline{x.y} = \overline{x} + \overline{y}$$

- **Permitem transformar uma soma de produtos num produto de somas e vice-versa**

- **Verificação por Tabelas de Verdade**

x	y	x + y	$\overline{x+y}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$\overline{x}$	$\overline{y}$	$\overline{x}.\overline{y}$
1	1	1
1	0	0
0	1	0
0	0	0

- **Generalização para *n* variáveis**

$$\overline{x_1 + x_2 + \ldots + x_n} = \overline{x}_1 \cdot \overline{x}_2 \ldots \overline{x}_n$$

$$\overline{x_1 \cdot x_1 \ldots x_n} = \overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_n$$

TÉCNICO LISBOA

Representação de Funções

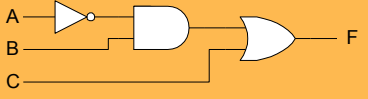
- **Por função Booleana:**

$$f = \bar{a}\,b + c$$

āb e c são os **termos** da função.

ā, b e c são os **literais**.

- **Por Circuito Lógico (ou Logigrama):**



- **Por Tabela de Verdade:**

a	b	c	ā b	f
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	0	1

TÉCNICO LISBOA

Funções mais de 2 variáveis

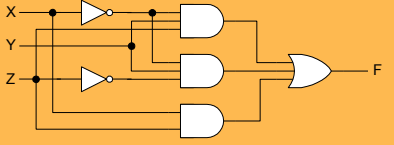
- Exemplo de Simplificação e Representação sob a forma de Logigrama:

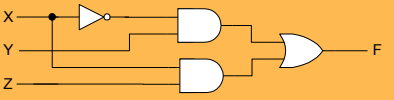
$$f = \bar{x}yz + \bar{x}y\bar{z} + xz$$

$$= \bar{x}y\left(z + \bar{z}\right) + xz$$

$$= \bar{x}y.1 + xz$$

$$= \bar{x}y + xz$$

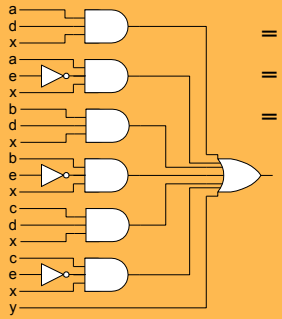




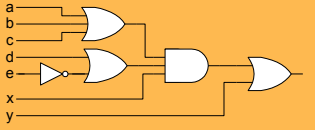
TÉCNICO LISBOA

Exemplo de Simplificação (II)

$$f = adx + a\bar{e}x + bdx + b\bar{e}x + cdx + c\bar{e}x + y$$
$$= \left(ad + a\bar{e} + bd + b\bar{e} + cd + c\bar{e}\right)x + y$$
$$= \left(\left(a+b+c\right)d + \left(a+b+c\right)\bar{e}\right)x + y$$
$$= \left(\left(a+b+c\right)\left(d+\bar{e}\right)\right)x + y$$



Realização a 2 níveis
(soma de produtos)



Realização Multinível

Especificação por tabela: Importância das funções AND, OR e NOT

A	B	C	f_1	f_2	f_3	f
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	1	1
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	1	0	0	1

$$f = f_1 + f_2 + f_3$$

$$f_1 = ABC$$

$$f_2 = AB\overline{C}$$

$$f_3 = A\overline{B}\,\overline{C}$$

$$f = ABC + AB\overline{C} + A\overline{B}\,\overline{C}$$

- É sempre possível definir uma função utilizando o conjunto {AND, OR, NOT} !!!

TÉCNICO LISBOA

Especificação por soma de mintermos

- O método anterior permitiu igualmente a passagem da representação de uma função por uma tabela para uma expressão algébrica
- Essa expressão é uma soma de produtos em que todos os produtos envolvem todas as variáveis da função

m	A	B	C	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

$$f = ABC + AB\overline{C} + A\overline{B}\,\overline{C}$$

- A estes produtos chama-se **mintermos**
- A expressão em termos de soma de mintermos é única
- A esta expressão também se chama 1ª forma canónica ou forma canónica normal disjuntiva
- Cada mintermo corresponde a um dos 1s da função

- É usual referir cada mintermo pelo número correspondente em binário
- Assim: $f = m_4 + m_6 + m_7 \Leftrightarrow f = \sum m(4,6,7)$

TÉCNICO LISBOA

Especificação por produto de maxtermos

- Em vez de se trabalharem com '1's, também é possível construir a expressão da função através dos seus '0's
- Essa expressão é um produto de somas em que todas as somas envolvem todas as variáveis da função

$$g = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + B + \overline{C})$$

M	A	B	C	g
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

- A cada uma das somas chama-se **maxtermo**
- A esta expressão também se chama 2ª forma canónica ou forma canónica normal conjuntiva
- Cada maxtermo corresponde a um dos 0s da função

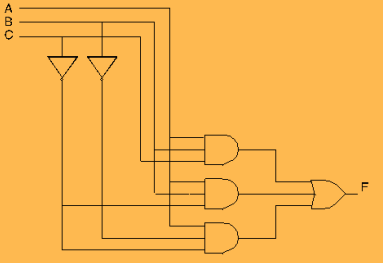
- É usual referir cada maxtermo pelo número correspondente em binário
- Assim: $g = M_0.M_1.M_2.M_3.M_5 \Leftrightarrow f = \prod M(0,1,2,3,5)$

TÉCNICO LISBOA

Especificação da 1ª forma canónica por logigrama

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$f = ABC + AB\overline{C} + A\overline{B}\,\overline{C}$$



- Cada mintermo é representado por uma das portas AND
- Existe uma correspondência entre cada AND, cada um dos 1s da tabela e cada um dos produtos da expressão

Importância das funções NAND e NOR

- Como vimos, qualquer função pode ser representada como uma soma de mintermos. Por exemplo:

$$f = ABC + AB\overline{C} + A\overline{B}\,\overline{C}$$

- Aplicando uma dupla negação à expressão:

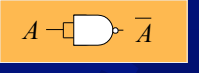
$$f = \overline{\overline{ABC + AB\overline{C} + A\overline{B}\,\overline{C}}}$$

- Aplicando as Leis de Morgan:

$$f = \overline{\overline{ABC} \cdot \overline{AB\overline{C}} \cdot \overline{A\overline{B}\,\overline{C}}}$$

- Obtém-se assim uma expressão em que só surgem NANDs e NOTs. Como um NOT pode ser feito com um NAND, então **qualquer função pode ser implementada só com NANDs**

$$\overline{A \cdot A} = \overline{A}$$



- O mesmo se aplica aos NORs (basta partir da 2ª forma canónica)

TÉCNICO LISBOA

Manipulação e Simplificação de Funções

- A manipulação de uma função pode ser feita para obter uma expressão mais simples ou para obter a expressão em certas formas:
  - Simplificar

$$
\begin{aligned}
f &= \overline{A}\overline{B}C + AC + BC \\
&= \left(\overline{A}\overline{B} + A + B\right)C \\
&= \left(\overline{B} + A + B\right)C \\
&= 1C \\
&= C
\end{aligned}
$$

  - Obter expressão com operadores de 2 variáveis e negações

$$
\begin{aligned}
f &= \overline{A}\overline{B}D + \overline{A}\overline{C}D + ACD + A\overline{B}\overline{D} \\
&= \overline{A}\left(\overline{B}D + \overline{C}D\right) + A\left(CD + \overline{B}\overline{D}\right)
\end{aligned}
$$

  - Obter expressão só com NANDS

$$
\begin{aligned}
f &= \overline{A}\overline{B}D + \overline{A}\overline{C}D + ACD + A\overline{B}\overline{D} \\
&= \overline{\overline{A}\overline{B}D \cdot \overline{A}\overline{C}D \cdot \overline{ACD} \cdot \overline{A\overline{B}\overline{D}}}
\end{aligned}
$$

TÉCNICO LISBOA

Bibliografia

- Arroz,G., Monteiro,J.C., Oliveira,A., "Arquitectura de Computadores, dos Sistemas Digitais aos Microprocessadores", Capítulo 2, 2ª Edição, 2009
- Mano,M., Kime,C. – "Logic and Computer Design Fundamentals", Prentice Hall, Caps 2.1,2.2
- Sêrro,C. – "Sistemas Digitais: Fundamentos Algébricos", IST Press, 2003

TÉCNICO LISBOA