Sistemas Digitais

Álgebra de Boole

Prof. Manoel Eusebio de Lima
Centro de Informática
Universidade Federal de Pernambuco

Postulados básicos da álgebra de chaveamento

- Postulado 1:
  - Uma variável booleana x tem dois valores possíveis 0 e 1. Esses valores são exclusivos:
    - **se x =o então x ≠ 1**
    - **se x =1 então x ≠ 0**
- Postulado 2:
  - A operação NOT é definida como:
    - **$\overline{0} = 1$ $\overline{1} = 0$**
- Postulado 3
  - As operações AND e OR são definidas como:

    0 . 0 = 0 0 + 0 = 0
    0 . 1 = 0 0 + 1 = 1
    1 . 0 = 0 1 + 0 = 1
    1 . 1 = 1 1 + 1 = 1
  - A partir destes postulados podem ser construídos os teoremas que permitem manipular e simplificar expressões lógicas.

Propriedades básicas da álgebra de chaveamento

þ **Leis e Teoremas da Álgebra Booleana:**

- *Operações com 0 and 1:*
  1. X + 0 = X 1D. X • 1 = X
  2. X + 1 = 1 2D. X • 0 = 0
- *Lei da Indepotência:*
  3. X + X = X 3D. X • X = X
- *Lei da Involução:*
  4. $(\overline{\overline{X}}) = X$
- *Lei de Complementação:*
  5. $X + \overline{X} = 1$ 5D. $X \cdot \overline{X} = 0$
- *Lei comutativa:*
  6. X + Y = Y + X 6D. X • Y = Y • X

Propriedades básicas da álgebra de chaveamento

þ *Leis Associativas:*

7. (X + Y) + Z = X + (Y + Z) = X + Y + Z 7D. (X • Y) • Z = X (Y • Z) = X • Y• Z

8. X • (Y+ Z) = (X • Y) + (X •Z) 8D. X + (Y • Z) = (X + Y) • (X + Z)

9. $X \cdot Y + X \cdot \overline{Y} = X$ 9D. $(X + Y) \cdot (X + \overline{Y}) = X$

10. X + X • Y = X 10D. X • (X + Y) = X

11. $(X + \overline{Y}) \cdot Y = X \cdot Y$ 11D. $(X \cdot \overline{Y}) + Y = X + Y$

þ ***Lei de DeMorgan:***

1. $\overline{(X + Y + Z + ...)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot ...$ 1D. $\overline{(X \cdot Y \cdot Z \cdot ...)} = \overline{X} + \overline{Y} + \overline{Z} + ...$
2. {F(X1,X2,...,Xn,0,1,+,•)}' = {F($\overline{X1}$,$\overline{X2}$,...,$\overline{Xn}$,1,0,•,+)}

Propriedades básicas da algebra de chaveamento

- Exemplo:
  - Provar que $x+\overline{x} = 1$
  - Usando a tabela verdade:

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

  - Provar x+(x.y) =x (lei da absorção)
  - Usando a tabela verdade

x	y	x+(x.y)	x
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	1

Propriedades da álgebra de chaveamento

- Provar que $x+(\overline{x}.y) = x+y$
  - Partindo de $x+(\overline{x}.y)$ teremos
  - Usando a lei distributiva $x+(\overline{x}.y) = (x+\overline{x}).(x+y)$
  - Usando a lei de complementação $(x+\overline{x}).(x+y) = 1.(x+y)$
  - Propriedade especial de **1**, 1.(x+y) = x+y
- Exemplo:
  - Simplificar a seguinte função lógica:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

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

Indepotência

Distributiva e Complementação

Distributiva

Exemplos de implementação de funções com minitermos

Soma de Produtos ***Termo produto/Minitermo***

A	B	C	F(A,B,C)	Minitermos
0	0	0	0	$\bar{A}\,\bar{B}\,\bar{C} = m_0$
0	0	1	0	$\bar{A}\,\bar{B}\,C = m_1$
0	1	0	0	$\bar{A}\,B\,\bar{C} = m_2$
0	1	1	1	$\bar{A}\,B\,C = m_3$
1	0	0	1	$A\,\bar{B}\,\bar{C} = m_4$
1	0	1	1	$A\,\bar{B}\,C = m_5$
1	1	0	1	$A\,B\,\bar{C} = m_6$
1	1	1	1	$A\,B\,C = m_7$

F(A,B,C) = Σm(3,4,5,6,7)

= m3 + m4 + m5 + m6 + m7

= A' B C + A B' C' + A B' C

+ A B C' + A B C

Forma canônica/forma mínima

F = A B' (C + C') + A' B C + A B (C' + C)

= A B' + A' B C + A B

= A (B' + B) + A' B C

= A + A' B C

= A + B C = (A+B).(A+C)

Exemplo de implementação de funções com maxtermos

A	B	C	F(A,B,C)	Maxtermos
0	0	0	0	$A+B+C = M_0$
0	0	1	0	$A+B+\bar{C} = M_1$
0	1	0	0	$A+\bar{B}+C = M_2$
0	1	1	1	$A+\bar{B}+\bar{C} = M_3$
1	0	0	1	$\bar{A}+B+C = M_4$
1	0	1	1	$\bar{A}+B+\bar{C} = M_5$
1	1	0	1	$\bar{A}+\bar{B}+C = M_6$
1	1	1	1	$\bar{A}+\bar{B}+\bar{C} = M_7$

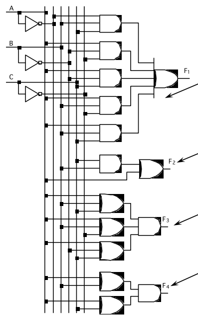
Notação de Maxtermos da Função F

F(A,B,C) = ΠM(0,1,2)

= (A + B + C) (A + B + C') (A + B' + C)
= (AA+AB+AC'+AB+BB+BC'+AC+BC+CC')(A + B' + C)
= (A+AB+AC'+B+BC'+AC+BC)(A + B' + C)
= (A(1+B+C'+C)+B(1+C'+C)+CC')(A + B' + C)
= (A(1+B+C'+C)+B(1+C'))(A + B' + C)
= (A+B)(A + B' + C)
= A(1+B+C+B)+BC

= A+BC = (A+B).(A+C)

Implementação das funções



Forma Canônica de Soma de Produtos

Soma de Produtos Minimizada

Forma Canônica de Produto de Soma

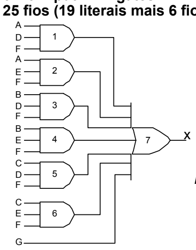
Forma Minimizada de Produto de Soma

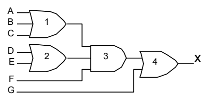
Lógica Multi-Nível

- ***Redução de equação na forma de soma de produtos***

x = A D F + A E F + B D F + B E F + C D F + C E F + G

6 x 3-input AND gates + 1 x 7-input OR gate (não existe)
25 fios (19 literais mais 6 fios internos)





Forma fatorada:

x = (A + B + C) (D + E) F + G

1 x 3-input OR gate, 2 x 2-input OR gates,
1 x 3-input AND gate
10 fios (7 literais mais 3 fios internos)

Redes NAND-NAND e NOR-NOR

- ***Redes NAND-NAND e NOR-NOR***

Lei de DeMorgan's: (A • B)' = A' + B'

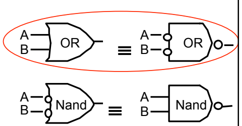
A + B = (A' • B')';

Ou seja,

OR é igual a NAND com entradas complementadas
NAND é igual a OR com entradas complementadas

Equivalência OR/AND

A	$\bar{A}$	B	$\bar{B}$	A + B	$\overline{\bar{A} \cdot \bar{B}}$	$\bar{A} + \bar{B}$	$\overline{A \cdot B}$
0	1	0	1	0	0	1	1
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	0	1	1	0	0



NAND-AND e NAND- NOT

- ***Redes NAND-AND e NAND-NOT***

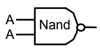
((A • B)')' = A•B (AND)

(A•A)' = NOT A

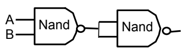
Ou seja,

AND é igual a NAND com saída complementada
NOT é igual a NAND com entradas conectadas

NOT A = A'



AND = A•B



Redes NAND-NAND e NOR-NOR

þ ***Equivalência AND/NOR***

Lei de DeMorgan's: $(A + B)' = A' \cdot B'$;

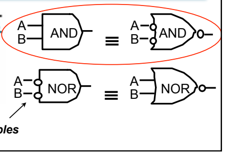
$(A \cdot B) = (A' + B')'$

Ou seja,

AND é igual a NOR com entradas complementadas
NOR é igual a AND com entradas complementadas

É possível fazermos conversão de circuitos com ANDs e ORs para circuitos com NANDs e NORs pela introdução de inversores apropriados ("bubbles")

A	$\overline{A}$	B	$\overline{B}$	$A \cdot B$	$\overline{\overline{A} + \overline{B}}$	$\overline{A} \cdot \overline{B}$	$\overline{A + B}$
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0



NOR-OR e NOR-NOT

þ ***Redes NOR-OR e NOR-NOT***

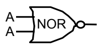
$((A + B)')' = A+B$ (OR)

$(A+A)' = \text{NOT } A$

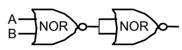
Ou seja,

OR é igual a NOR com saída complementada
NOT é igual a NOR com entradas conectadas

NOT A = A'



OR = A+B



Portas lógicas Universais

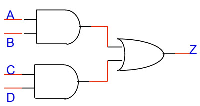
- Se a partir de portas NAND podemos fazer portas NOT, AND e OR
- e, se a partir de portas NOR podemos fazer portas NOT, OR e AND
- Podemos concluir que:

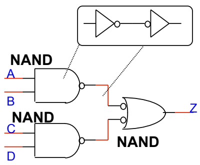
"É possível implementarmos qualquer circuito lógico digital contando apenas com portas lógicas NAND ou apenas com portas lógicas NOR"

Mapeamento AND/OR -> NAND/NAND

þ ***Exemplo: Mapear circuito AND/OR para circuito NAND/NAND***

$Z = (A \cdot B) + (C \cdot D)$





$Z = [(A \cdot B)' \, (C \cdot D)']'$

$= [(A' + B') \, (C' + D')]'$

$= [(A' + B')' \cdot (C' + D')']$

$= (A \cdot B) + (C \cdot D)$

Verificar equivalência das duas formas

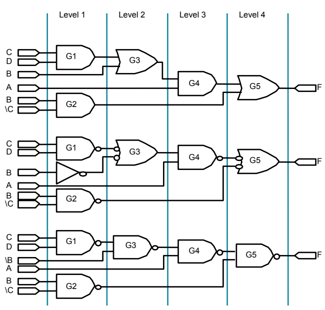
Lógica Multi-Nível

þ **$F = A (B + C D) + B C'$**

Circuito Original com AND-OR

Introdução e Conservação de inversores

Implementação em termos de tradicionais NAND Gates



Exercício - Exemplo de projeto

- Uma companhia instituiu o seguinte controle para o acesso de seus três estacionamentos. Cada empregado tem um cartão que deve ser inserido numa brecha especial que existe em cada portão. O portão só abrirá se o empregado estiver autorizado a usar o estacionamento.

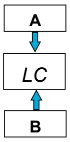
Tipo de empregado	**E_1**	**E_2**	**E_3**
Dirigentes	s	s	s
Administrados	s	s	n
Engenheiros	s	n	s
Secretários	n	s	s
Mecânicos	s	s	n
Eletricistas	s	n	s
Contadores	n	s	n

Tipo de empregado	x_1	x_2	x_3	E_1	E_2	E_3
Nenhuma entrada	0	0	0	0	0	0
Dirigentes	0	0	1	1	1	1
Administradores	0	1	0	1	1	0
Engenheiros	0	1	1	1	0	1
Secretários	1	0	0	0	1	1
Mecânicos	1	0	1	1	1	0
Eletricistas	1	1	0	1	0	1
Contadores	1	1	1	0	1	0

Considere s='1'
n='0'

Exemplo de Projeto

- 2 registradores de 2-bits A e B são usados para armazenar 2 números binários positivos. Projete um circuito que compute o valor absoluto da diferença.



Z= |A-B|

onde Z pode variar de 0 a 3

[A] = [a_1, a_2] Z= [Z_1, Z_2] = [f_1(a_1, a_2, b_1, b_2), f_2(a_1, a_2, b_1, b_2)]

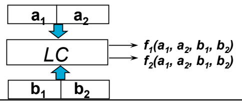
[B] = [b_1, b_2]

O projeto possui 4 entradas e duas saídas

Exemplo: A =2 [1,0] ; B=3[1,1]:

f_1(1,0,1,1)=0

f_2(1,0,1,1)=1   => Z=|A-B| =1

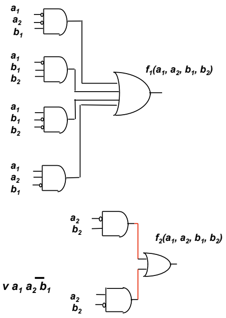


Exemplo de Projeto

a_1 a_2	b_1 b_2	$f_1(a_1, a_2, b_1, b_2)$	$f_2(a_1, a_2, b_1, b_2)$
0 0	0 0	0	0
0 0	0 1	0	1
0 0	1 0	1	0
0 0	1 1	1	1
0 1	0 0	0	1
0 1	0 1	0	0
0 1	1 0	0	1
0 1	1 1	1	0
1 0	0 0	1	0
1 0	0 1	0	1
1 0	1 0	0	0
1 0	1 1	0	1
1 1	0 0	1	1
1 1	0 1	1	0
1 1	1 0	0	1
1 1	1 1	0	0

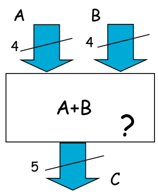
$$f_1(a_1, a_2, b_1, b_2) = \overline{a_1}\,\overline{a_2}\, b_1 \vee \overline{a_1}\, b_1 b_2 \vee a_1 \overline{b_1}\,\overline{b_2} \vee a_1 a_2 \overline{b_1}$$

$$f_2(a_1, a_2, b_1, b_2) = a_2 \overline{b_2} \vee \overline{a_2}\, b_2$$



Problema da decomposição - Exemplo

- Implementar um somador binário de dois vetores de 4 bits:



C = A+B

$c_0 = f_0(a_3, a_2,......a_0 , b_3, b_2, ...b_0) = f_0(a_0, b_0)$

$c_1 = f_1(a_3, a_2,......a_0 , b_3, b_2, ...b_0) = f_1(a_1, b_1)$

....

$c_3 = f_3(a_3, a_2,......a_0 , b_3, b_2, ...b_0) = f_3(a_3, b_3)$

Decomposição de funções complexas

- Sugestões
  - Identificar módulos que se repetem
  - Desenvolver problemas com hierarquia
    - Desenvolver módulos grandes a partir de módulos menores
    - Identificar claramente as entradas e saídas das funções em todos os níveis da hierarquia
      - Tamanho e tipos de dados

Problema de decomposição

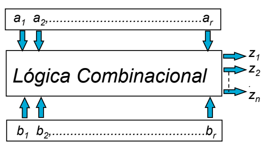
- Z = F(A,B)
- **onde A e B são representados por [a_1, a_2, ...a_r,] e [b_1, b_2, ...b_r,]**
- **A tabela verdade de F(A,B) teria 2^{2r} linhas, assim para r=5 teríamos 1024 linhas de possíveis combinações para a função Z.**
- **Z sendo representado por [z_1, z_2, ...z_r] temos:**

$z_1 = f_1(a_1, a_2,......a_r , b_1, b_2, ...b_r) = f_1(a_1, b_1)$

$z_2 = f_2(a_1, a_2,......a_r , b_1, b_2, ...b_r) = f_2(a_2, b_2)$

....

$z_n = f_n(a_1, a_2,......a_r , b_1, b_2, ...b_r) = f_n(a_n, b_n)$

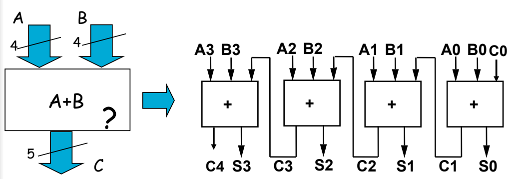


Decomposição visa quebrar o problema em blocos menores, onde cada um destes blocos resolva a função para cada conjunto de variáveis de determinado índice.

O objetivo é usar as características estruturais do problema para decompô-lo em sub-projetos fáceis de serem resolvidos.

Implementação do circuito do somador- Adição binária

- Somador para 4 bits de compondo o problema
  - **Full-adder de 4 bits a partir de módulos de 1 bit conectados em cascata**



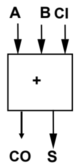
Adição binária

þ *Full Adder (tabela verdade)*

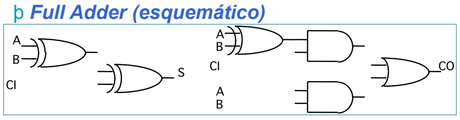
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S = CI xor A xor B

CO = A xor B + A B =
= CI (A ⊕ B) + A B



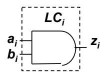
þ *Full Adder (esquemático)*



Decomposição – Operações lógicas

- Projete um circuito lógico que compute:
  - *Z = A • B*
  - ***Onde A e B correspondem as informações contidas dos nos registradores de r-bits e o valor resultante em Z é definido pelo AND bit-a-bit dos conteúdos dos registradores.***
  - ***exemplo:***
    - **A=[1,1,0,1] e B=[0,1,1,0], então Z=[0,1,0,0]**

- $z_1 = a_1 \bullet b_1$
- $z_2 = a_2 \bullet b_2$
- $z_r = a_r \bullet b_r$



$f_i(a_i, a_i) = a_i \bullet b_i$