Sistemas Digitais

Aula 5

1

Postulados básicos da álgebra de chaveamento

2

- 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:
    - $\bar{0} = 1$ $\bar{1} = 0$
- Postulado 3
  - As operações AND e OR são definidas como:
    - $0 \wedge 0 = 0$ $0 \vee 0 = 0$
    - $0 \wedge 1 = 0$ $0 \vee 1 = 1$
    - $1 \wedge 0 = 0$ $1 \vee 0 = 1$
    - $1 \wedge 1 = 1$ $1 \vee 1 = 1$
  - A partir destes postulados podem ser construídos os teoremas que permitem manipular e simplificar expressões lógicas.

Propriedades básicas da álgebra de chaveamento

3

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

- *Operações com 0 and 1:*

1. $X + 0 = X$	1D. $X \cdot 1 = X$
2. $X + 1 = 1$	2D. $X \cdot 0 = 0$

- *Lei da Indepotência:*

3. $X + X = X$	3D. $X \cdot X = X$

- *Lei da Involução:*

4. $(\bar{X}) = X$

- *Lei de Complementação:*

5. $X + \bar{X} = 1$	5D. $X \cdot \bar{X} = 0$

- *Lei comutativa:*

6. $X + Y = Y + X$	6D. $X \cdot Y = Y \cdot X$

Propriedades básicas da álgebra de chaveamento

4

☑ *Leis Associativas:*

7. $(X + Y) + Z = X + (Y + Z) = X + Y + Z$	7D. $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z$
8. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$	8D. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
9. $X \cdot Y + X \cdot \bar{Y} = X$	9D. $(X + Y) \cdot (X + \bar{Y}) = X$
10. $X + X \cdot Y = X$	10D. $X \cdot (X + Y) = X$
11. $(X + \bar{Y}) \cdot Y = X \cdot Y$	11D. $(X \cdot \bar{Y}) + Y = X + Y$

☑ ***Lei de DeMorgan:***

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

Propriedades básicas da algebra de chaveamento

5

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

x	x+$\bar{x}$	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

6

- Provar que $x+(\bar{x}.y) = x+y$
  - Partindo de $x+(\bar{x}.y)$ teremos
  - Usando a lei distributiva $x+(\bar{x}.y) = (x+\bar{x}).(x+y)$
  - Usando a lei de complementação $(x+\bar{x}).(x+y) = 1.(x+y)$
  - 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

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

$= \bar{x}_1.\bar{x}_2.\bar{x}_3 + \bar{x}_1.\bar{x}_2.x_3 + \bar{x}_1.\bar{x}_2.x_3 + x_1.\bar{x}_2.x_3$

$= \bar{x}_1.\bar{x}_2(\bar{x}_3 + x_3) + \bar{x}_2.x_3(\bar{x}_1 + x_1)$

$= \bar{x}_1.\bar{x}_2 + \bar{x}_2.x_3$

$= \boxed{\bar{x}_2.(\bar{x}_1 + x_3)}$

Indepotência

Distributiva e Complementação

Distributiva

7

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)

8

Exemplo de implementação de funções com maxtermos

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

= (A(1+B+C'+C)+B(1+C'))(A + B' + C)

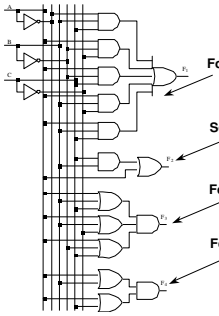
= (A+B)(A + B' + C)

= A(1+B+C+B)+BC

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

9

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

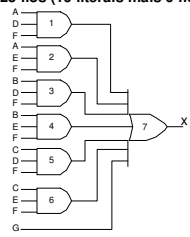
10

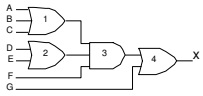
Lógica Multi-Nível

☑ ***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)

11

Redes NAND-NAND e NOR-NOR

☑ ***Redes NAND-NAND e NOR-NOR***

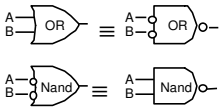
Lei de DeMorgan's: **(A + B)' = A' • B';** **(A • B)' = A' + B'**

Escrito Diferente: **A + B = (A' • B')';** **(A • B) = (A' + B')'**

Ou seja,
OR é igual a NAND com entradas complementadas
AND é igual a NOR com entradas complementadas
NAND é igual a OR com entradas complementadas
NOR é igual a AND 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



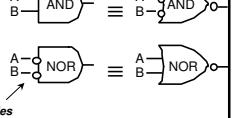
12

Redes NAND-NAND e NOR-NOR

☑ ***Equivalência AND/NOR***

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

A	$\bar{A}$	B	$\bar{B}$	A • B	$\overline{\bar{A} + \bar{B}}$	$\bar{A} \cdot \bar{B}$	$\overline{A + B}$
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



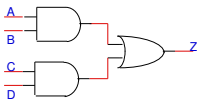
bubbles

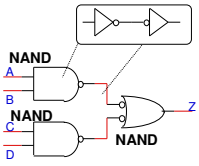
13

Mapeamento AND/OR -> NAND/NAND

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

Z = (A • B) + (C • D)





Verificar equivalência das duas formas

Z = [(A•B)' (C•D)']'

= [(A' + B') (C' + D')]'

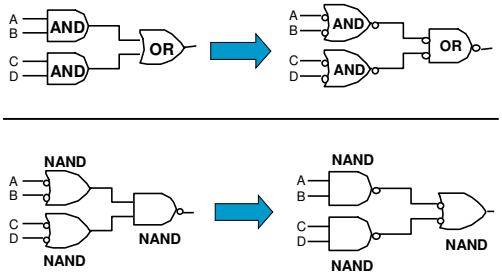
= [(A' + B')' • (C' + D')']

= (A • B) + (C • D)

14

Mapeamento AND/OR -> NAND/NAND

Exemplo: Mapear circuito AND/OR em portas NAND/NAND



15

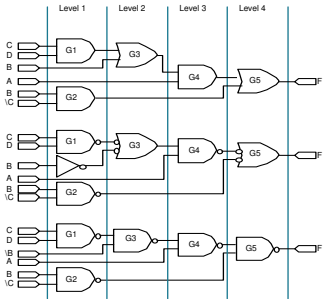
Lógica Multi-Nível

☑ F = A (B + C D) + B C'

Circuito Original com AND-OR

Introdução e Conservação de inversores

Implementação em termos de tradicionais NAND Gates



16

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

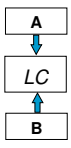
Tipo de empregado	x_1	x_2	x_3	E_1	E_2	E_3
Nenhuma entrada	0	0	0	0	0	0
Dirigentes	0	0	1	1	1	1
Administradores	0	1	0	1	1	0
Engenheiros	0	1	1	1	0	1
Secretários	1	0	0	0	1	1
Mecânicos	1	0	1	1	1	0
Eletricistas	1	1	0	1	0	1
Contadores	1	1	1	0	1	0

Considere s='1'
n='0'

17

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] = [$f_1(a_1, a_2, b_1, b_2)$, $f_2(a_1, a_2, b_1, b_2)$]

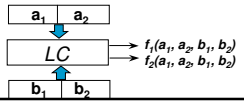
[B] = [b_1, b_2]

O projeto possui 4 entradas e duas saídas

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

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

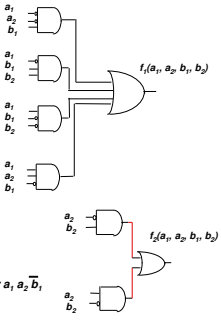
f_2(1,0,1,1)=1 => Z=|A-B| =1



18

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



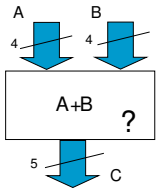
$f_1(a_1, a_2, b_1, b_2) = \overline{a}_1 \overline{a}_2 b_1 v \overline{a}_1 b_1 b_2 v a_1 \overline{b}_1 \overline{b}_2 v a_1 a_2 \overline{b}_1$

$f_2(a_1, a_2, b_1, b_2) = a_2 \overline{b}_2 v \overline{a}_2 b_2$

19

Problema da decomposição - Exemplo

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



C = A+B

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

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

$\ldots\ldots$

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

20

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

21

Problema de decomposição

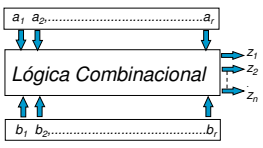
- Z = F(A,B)
- **onde A e B são representados por $[a_1, a_2, \ldots a_r]$ e $[b_1, b_2, \ldots b_r]$**
- **A tabela verdade de F(A,B) teria 2^{2r} linhas, assim para r=5 teríamos 1024 linhas de possíveis combinações para a função Z.**
- **Z sendo representado por $[z_1, z_2, \ldots z_r]$ temos:**

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

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

$\ldots\ldots$

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

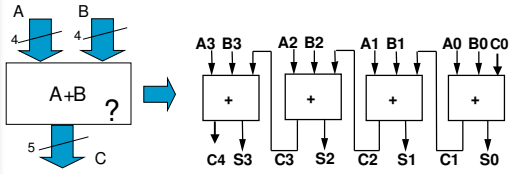


Decomposição visa quebrar o problema em blocos menores, onde cada um destes blocos resolva a função para cada conjunto de variáveis de determinado índice.
O objetivo é usar as características estruturais do problema para decompô-lo em sub-projetos fáceis de serem resolvidos.

22

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**



23

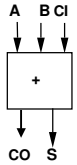
Adição binária

☑ ***Full Adder (tabela verdade)***

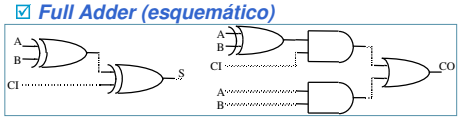
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S = CI xor A xor B

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



☑ ***Full Adder (esquemático)***

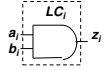


24

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]**

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



$f_i(a_i, a_j) = a_i \cdot b_i$