MC542

Organização de Computadores
Teoria e Prática

2007
Prof. Paulo Cesar Centoducatte
ducatte@ic.unicamp.br
www.ic.unicamp.br/~ducatte

MC542

Circuitos Lógicos

Projeto de Circuitos Combinacionais

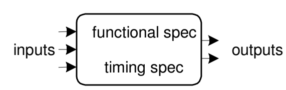
"DDCA" - (Capítulo 2)
"FDL" - (Capítulos 2 e 4)

Título do Capítulo Abordado
Sumário

Introdução

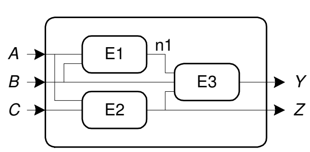
Um circuito lógico é composto de:

- Entradas (inputs)
- Saídas (outputs)
- Especificação Funcional
- Especificação da temporização (timing)



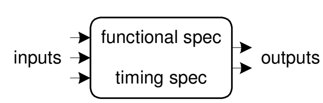
Circuito

- Nodes
  - Inputs: *A, B, C*
  - Outputs: *Y, Z*
  - Interno: n1
- Elementos do Circuito
  - E1, E2, E3



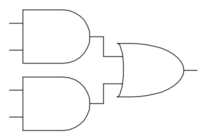
Tipos de Circuitos Lógicos

- **Combinacional**
  - Sem memória
  - As saídas são determinadas pelos valores correntes das entradas
- **Seqüencial**
  - Tem memória
  - As saídas são determinadas pelos valores anteriores e correntes das entradas



Composição de Circuitos Combinacionais

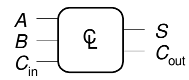
- Todos os elementos do circuito é combinacional
- Todos os nodes do circuito ou é uma entrada do circuito ou está conectado a uma saída de um elemento do circuito
- O circuito não contem ciclos: todo caminho no circuito visita cada node no máximo uma vez
- **Exemplo:**



Equação Booleana

- Especificação funcional das saídas em termos das entradas usando-se operadores booleanos
- Exemplo:

S = F(A, B, Cin)
Cout = F(A, B, Cin)



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Forma Soma-de-Produtos (SOP)

- Tada equação booleana pode ser descrita na forma SOP
- Cada linha da tabela verdade é associada a um **mintermo**
- Um mintermo é um produoto (AND) de literais
- Cada mintermo é TRUE (1) para uma dada linha (e somente para essa linha)
- A função é formada pelo OR dos mintermos para os quais a saída é TRUE (1)
- Assim, a suma (OR) de produtos (termos AND)

A	B	Y	minterm
0	0	0	$\overline{A}\ \overline{B}$
0	1	1	$\overline{A}\ B$
1	0	0	$A\ \overline{B}$
1	1	1	$A\ B$

Y = F(A, B, C) = $\overline{A}B + AB$

Terminologia

- **Literal** - Uma variável complementada ou não em um termo produto (ou termo soma)
- **Implicante** – Um termo produto que implementa um ou mais 1´s da função. Exemplo: um míntermo é um implicante; um produto gerado pela simplificação de uma váriável de dois míntermos é um implicante.
- Implicante Principal – Um implicante que não pode ser simplificado em outro implicante com menos literais.
- **Implicante Essencial** – Implicante Principal que é imprecindivel na realização da função (existe pelo menos um "1" que só é coberto por ele).
- *Cobertura* – Uma coleção de implicantes que implementam a função (implementam todos os 1´s da função).
- *Custo* – número de portas + número de entradas de todas as portas (assumiremos que as entrads primárias estão disponíveis tanto na forma verdadeira quanto complementada).

Forma Produto-de-Somas (POS)

- Tada equação booleana pode ser descrita na forma POS
- Cada linha da tabela verdade é associada a um **maxtermo**
- Um maxtermo é uma suma (OR) de literais
- cada maxtermo é FALSE (0) para uma dada linha (e somente para essa linha)
- A função é formada pelo AND dos maxtermos para os quais a saída é False (0)
- Assim, um produto (AND) de soma (termos OR)

A	B	Y	maxterm
0	0	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	0	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

Y = F(A, B, C) = $(A + B)(\overline{A} + B)$

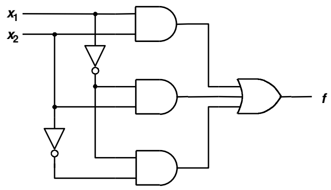
Síntese Usando Portas And, OR e Not

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

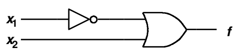
1. Implemente cada 1 da tabela verdade com um AND e Nots
2. Faça um OR dos circuitos criados em 1.
3. Opcional: simplifique a função

Soma de Produtos

Síntese Usando Portas And, OR e Not



Canonical sum-of-products



Minimal-cost realization

Síntese Usando Portas And, OR e Not

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

1. Implemente cada 0 da tabela verdade com um OR e Nots
2. Faça um AND dos circuitos criados em 1.
3. Opcional: simplifique a função

Produto de Somas

Soma-de-Produtos e Produtos-de-Soma
(SoP e PoS)

- **Mintermos e Maxtermos**
  - Mintermo: Implementa um "1" da tabela verdade
  - Maxtermo: Implementa um "0" da tabela verdade

- **Forma canônica:**
  - de Mintermos: a expressão que representa a função possui todos os mintermos (não simplificada)
  - de Maxtermos: a expressão que representa a função possui todos os maxtermos (não simplificada)

Numeração de Mintermos e Maxtermos

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2 x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1 x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1 x_2 x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2 x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1 x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Exemplo

Assuma que temos um salão com três portas e próximo a cada uma delas temos uma chave para acender/apagar a luz. Projete o circuito de controle que acende/apaga a luz do salão.

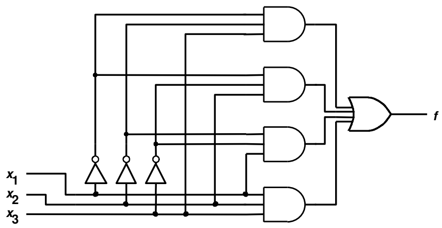
Solução:

- x1; x2 e x3 váriaveis que indicam o estado das chaves 1, 2 e 3 (1 -> fechada; 0 -> aberta)
- Monte a tabela verdade que representa a função desejada, ié: ao acionarmos uma chave (mudar seu estado) se a luz está apagada ela acende e vice-versa
- Sintetize o circuito de controle

Exemplo: tabela verdade

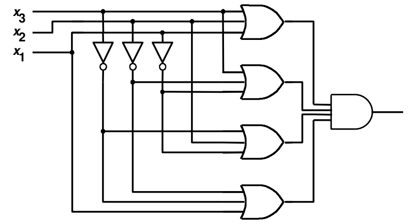
x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Exemplo



Implementação SOP do controlador three-way

Exemplo



Implementação POS do controlador three-way

Álgebra Booleana

- **Conjunto de axiomas e teoremas: usados para simplificar equações Booleanas**
- **Similar à algebra regular, porém mais simples em muitos casos já que as variáveis só podem ter dois valores (1 or 0)**
- **Axiomas e teoremas obedecem aos principios da dualidade:**
  - Trocando-se ANDs por Ors (e vice-versa) e 0's por 1's (e vice-versa)

Axiomas e Teoremas

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

Teoremas

	Theorem		Dual	Name
T6	$B \bullet C = C \bullet B$	T6′	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7′	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + B \bullet D = B \bullet (C + D)$	T8′	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9′	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10′	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = B \bullet C + \overline{B} \bullet D$	T11′	$(B + C) \bullet (\overline{B} + D) \bullet (C + D) = (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = (\overline{B_0} + \overline{B_1} + \overline{B_2} \ldots)$	T12′	$\overline{B_0 + B_1 + B_2 \ldots} = (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2})$	De Morgan's Theorem

Axiomas e Teoremas

- **As Demonstrações podem ser feitas usando-se:**
  - ***Indução Perfeita***
  - ***Gráfica (Diagrama de Venn)***
  - ***Manipulação Algébrica***

Precedência dos Operadores

1. Not
2. AND
3. OR

Prova do Teorema de De Morgan

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

Indução Perfeita

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\overline{x}$	$\overline{y}$	$\overline{x} + \overline{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0
		LHS			RHS	

Prova do Teorema da Distribuição

$$x.(y+z) = x.y + x.z$$

Diagrama de Venn

Manipulação Algébrica

- $Y = \overline{A}B + AB$

$= B(\overline{A} + A)$ T8

$= B(1)$ T5'

$= B$ T1

Manipulação Algébrica

- $Y = B(AB + ABC)$

$= B(AB(1 + C))$ T8

$= B(AB(1))$ T2'

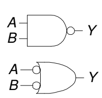
$= B(AB)$ T1

$= A(BB)$ T7

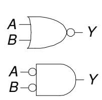
$= AB$ T3

Teorema De Morgan

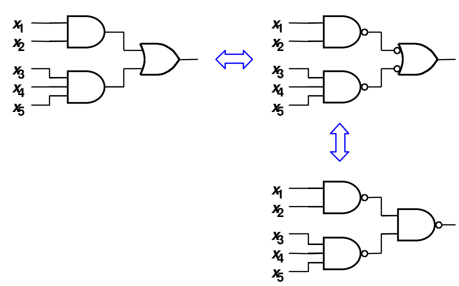
- $Y = \overline{AB} = \overline{A} + \overline{B}$



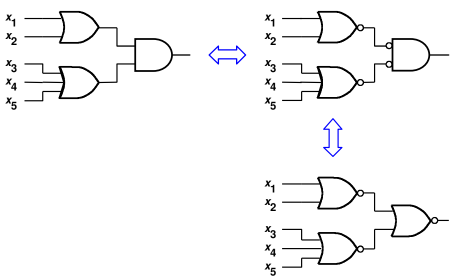
- $Y = \overline{A + B} = \overline{A} \bullet \overline{B}$



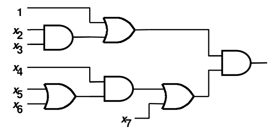
Exemplo de Síntese Só com NANDs



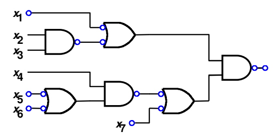
Exemplo de Síntese Só com NORs



Exemplo

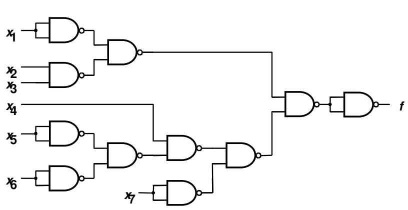


Circuit with AND and OR gates

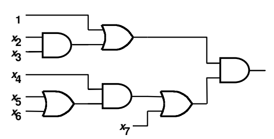


Convertendo para NANDs

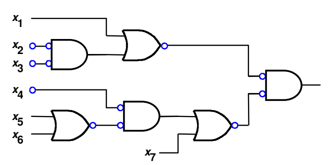
Exemplo (cont.)



Exemplo (cont.)

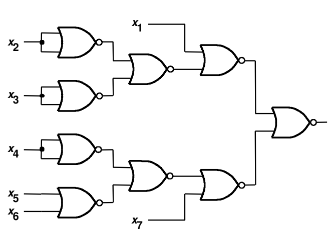


Circuit with AND and OR gates



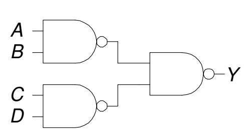
Convertendo para NORs

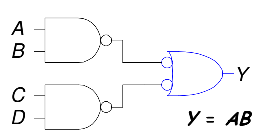
Exemplo (cont.)



Exercício:

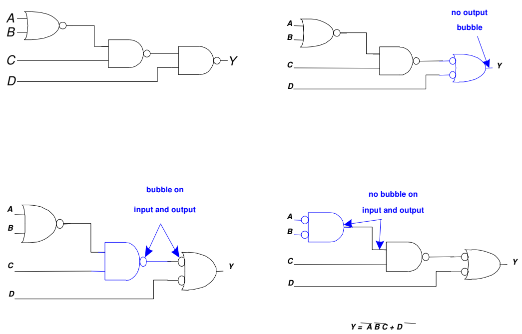
- **Qual é a expressão booleana para o circuito abaixo?**





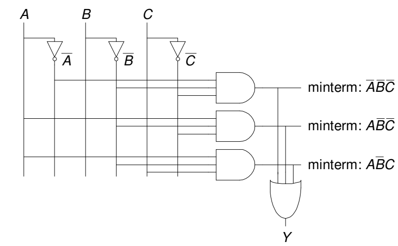
$Y = AB + CD$

Técnica Bubble Pushing

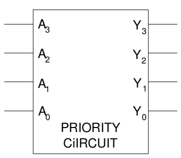


Síntese Lógica

- Lógica em dois níveis: ANDs seguidos de OR
- Exemplo: $Y = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$



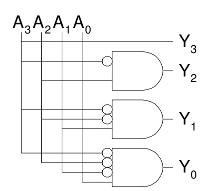
Circuitos Multi-Saídas



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**

Circuitos Multi-Saídas

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**



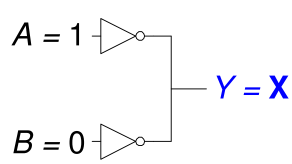
Don't Cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**0**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**0**	**0**
0	1	0	1	**0**	**1**	**0**	**0**
0	1	1	0	**0**	**1**	**0**	**0**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**0**
1	0	0	1	**1**	**0**	**0**	**0**
1	0	1	0	**1**	**0**	**0**	**0**
1	0	1	1	**1**	**0**	**0**	**0**
1	1	0	0	**1**	**0**	**0**	**0**
1	1	0	1	**1**	**0**	**0**	**0**
1	1	1	0	**1**	**0**	**0**	**0**
1	1	1	1	**1**	**0**	**0**	**0**

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	X	**0**	**0**	**1**	**0**
0	1	X	X	**0**	**1**	**0**	**0**
1	X	X	X	**1**	**0**	**0**	**0**

Contenção: X

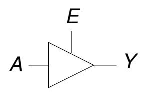
- Contenção (conflito): o circuito tenta colocar a saída em 1 e 0



Alta Impedância: Z

- A saída fica isolada das entradas

Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Simplificação de Funções Lógicas

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$f = \Sigma\, m(0, 2, 4, 5, 6)$

Função Mínima?

$f = \overline{x}_1\, \overline{x}_2$

Como determinar f mínima?

Karnaugh Maps (K-Maps)

- Funções Booleanas podem ser minimizadas combinando-se termos
- K-maps minimiza as expressões graficamente
- *PA + PA = P*

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y: C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Simplificação de Funções Lógicas

- mo e m2 ?

$m_0 = \overline{x}_1\, \overline{x}_2\, \overline{x}_3$ $\quad$ $m_2 = \overline{x}_1\, x_2\, \overline{x}_3$

$\overline{x}_1\, \overline{x}_3$

Simplificação de Funções Lógicas

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

O Mapa de Karnaugh agrupa os mintermos "simplificáveis" de forma gráfica facilitando o processo de duplicação de termos.
($x = x + x$)

Mapa de Karnaugh 2 variáveis

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

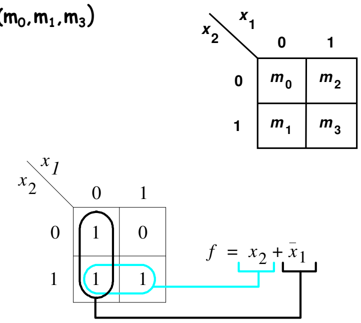
Truth table

x_2 \ x_1	0	1
0	m_0	m_2
1	m_1	m_3

Karnaugh map

Exemplo de uso do Mapa K

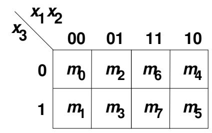
$f = \Sigma(m_0, m_1, m_3)$



Mapa de Karnaugh 3 variáveis

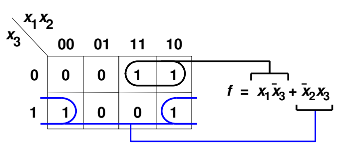
x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

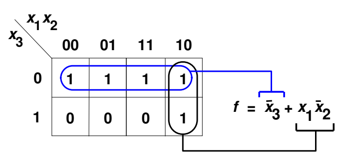
Truth table



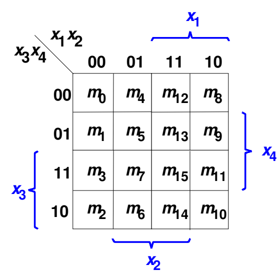
Karnaugh map

Exemplos de Uso do Mapa K para 3 variáveis

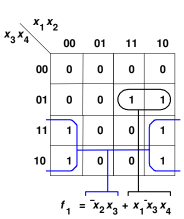


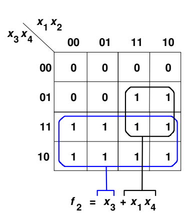


Mapa de Karnaugh 4 variáveis

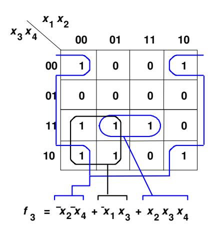


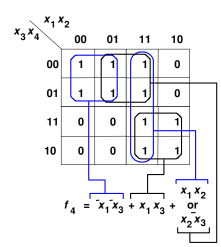
Exemplos de Uso do Mapa K para 4 variáveis



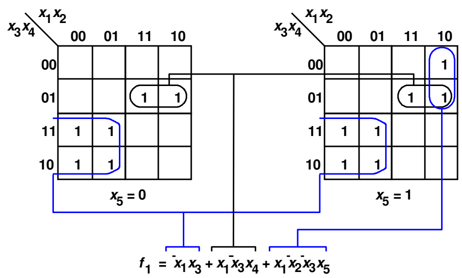


Exemplos de Uso do Mapa K para 4 variáveis





Mapa de Karnaugh 5 variáveis



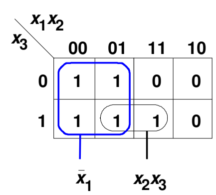
Minimização

- Terminologia:
  - **Literal** - Uma variável complementada ou não em um termo produto
  - **Implicante** - Um termo produto que implementa um ou mais 1´s da função. Exemplo: um mintermo é um implicante; um produto gerado pela simplificação de uma variável de dois mintermos é um implicante.
  - **Implicante Principal** - Um implicante que não pode ser simplificado em outro implicante com menos literais.
  - **Implicante Essencial** - Implicante Principal que é imprecindível na realização da função (existe pelo menos um "1" que só é coberto por ele).
  - **Cobertura** - Uma coleção de implicantes que implementam a função (implementam todos os 1´s da função).
  - **Custo** - número de portas + número de entradas de todas as portas (assumiremos que as entrads primárias estão disponíveis tanto na forma verdadeira quanto complementada).

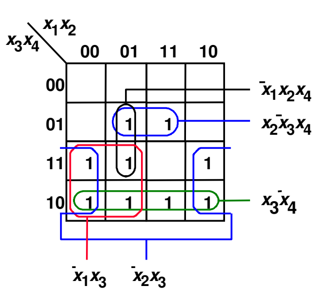
Uso do Mapa K

1. Represente todos mintermos da função no mapa K
2. Determine todos os implicantes principais
3. Determine o conjunto dos Implicantes Essenciais
4. Se o conjunto dos implicantes essenciais cobre todos os valores 1´s da função, então tem-se a função de custo mínimo. Caso contrário, determine o conjunto de custo mínimo, usando os implicantes principais, que cobre os 1´s não cobertos pelo conjunto de implicantes essenciais.

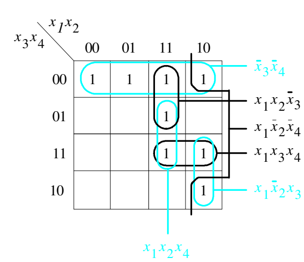
Exemplos



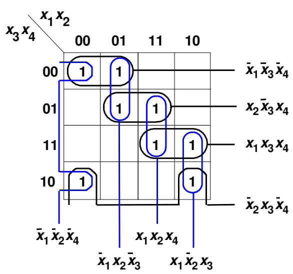
Exemplos



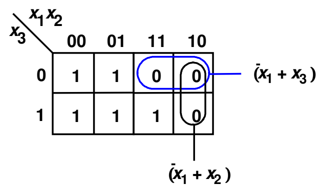
Exemplos



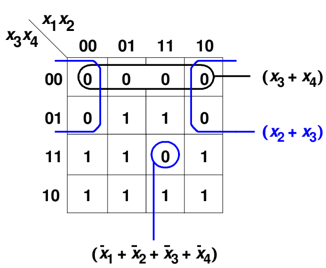
Exemplos



Minimização de Produto-de-Somas



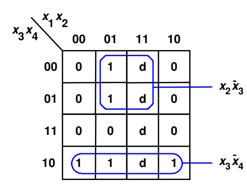
Exemplo



Funções Incompletamente Especificadas

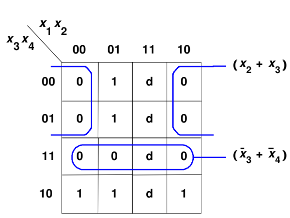
- **Sabe-se, pela natureza do problema, que determinadas combinações dos possíveis valores para as entradas não ocorrem durante a operação do sistema que se deseja projetar.**

$f = \Sigma\, m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$



Funções Incompletamente Especificadas

Implementada como Produto-de-Somas

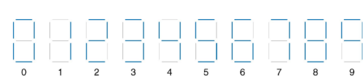


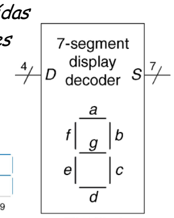
Exercício

- Um decodificador de sete-segmentos tem 4 bits de entrada, $D_{3:0}$, e produz sete saídas que controlam diodos emisores de luz que mostram valores de 0 a 9. As setes saídas, normalmente, são denominadas de segmento ***a*** a ***g***, ou ***S_a-S_g*** *como mostrado na figura abaixo.*

  a) Escreva a tabela verdade para as saídas e use K-map para minimizar as equações booleanas que as implemente.

  b) refaça a) usando don't cares





Blocos Básicos

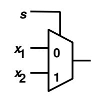
- Multiplexadores (MUX)
- Decodificadores

Multiplexadores (MUX)

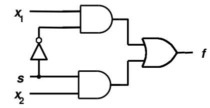
- Seleciona uma de *N* entrada como saída.
- $\log_2 N$-bitde entrada de controle
- Exemplo:

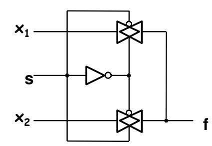
2:1 Mux

$s\,x_1x_2$	$f\,(s, x_1, x_2)$
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	1



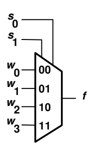
s	$f\,(s, x_1, x_2)$
0	x_1
1	x_2

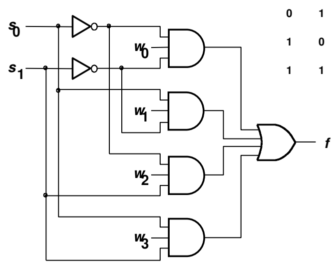




transmission gates

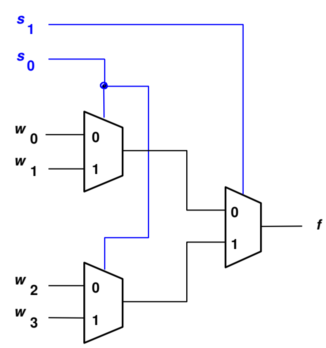
Multiplexador: 4 para 1 (4:1)



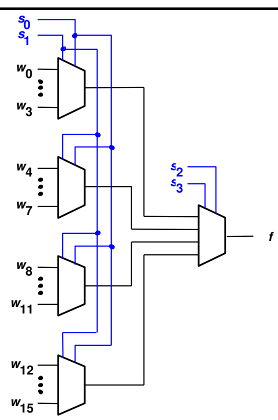


s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

Mux 4:1 a partir de Mux 2:1

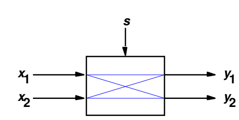


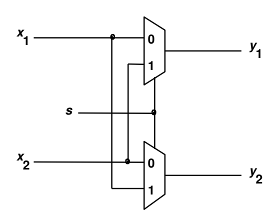
Mux 16:1



Exemplo de Uso de Mux

2x2 crossbar switch



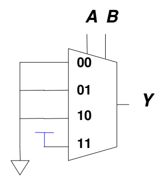


Lógica Usando Mux

- Usand o mux como uma lookup table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



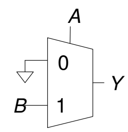
Implementando Funções com Mux

- Reduzindo o tamanho do Mux

$Y = AB$

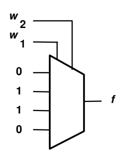
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



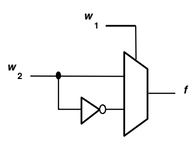
Síntese de Funções Lógicas Usando MUX

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0



w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

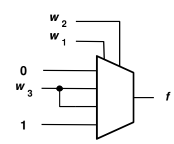
w_1	f
0	w_2
1	$\overline{w}_2$



Exemplo

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

w_1	w_2	f
0	0	0
0	1	w_3
1	0	w_3
1	1	1

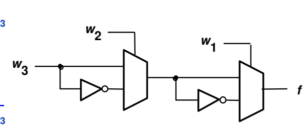


Exemplo

w_1	w_2	w_3	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

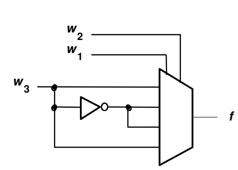
$w_2 \oplus w_3$

$\overline{w_2 \oplus w_3}$



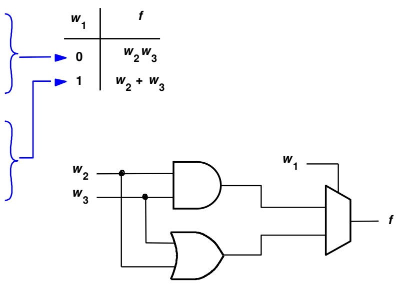
Exemplo

w_1	w_2	w_3	f	
0	0	0	0	w_3
0	0	1	1	
0	1	0	1	$\overline{w}_3$
0	1	1	0	
1	0	0	1	$\overline{w}_3$
1	0	1	0	
1	1	0	0	w_3
1	1	1	1	



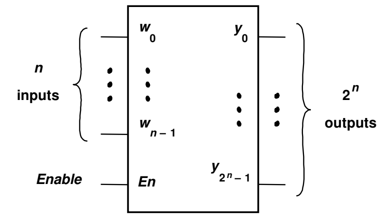
Exemplo
Maioria de uns

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



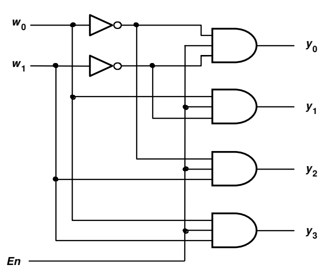
Decodificadores

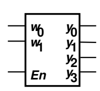
- N inputs, 2^N outputs
- One-hot outputs: somente uma saída HIGH por vez



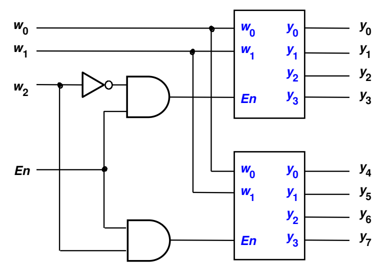
Decodicifador 2:4

En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

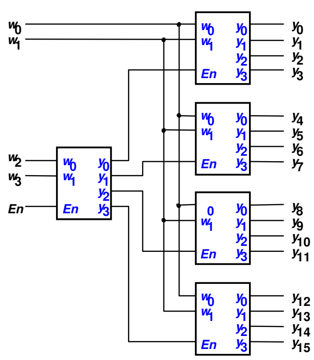




Decodicifador 3:8 Usando 2:4

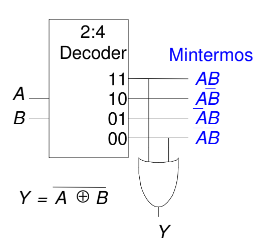


Decodicifador 4:16 Usando 2:4



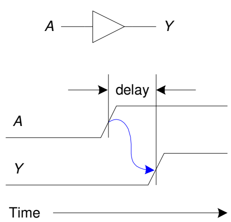
Lógica Usando Decodificadores

- OR de mintermos



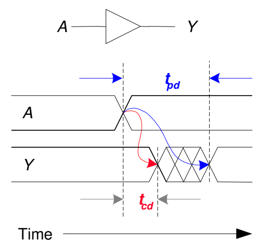
Timing

- Delay: atraso entre a mudança na entrada e na saída
- Um dos maiores desafios em projeto de circuitos: tornar o circuito mais rápido



Delay: Propagação e Contaminação

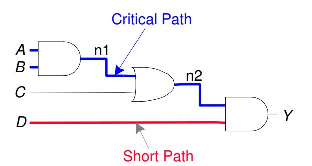
- Propagation delay: t_{pd} = max delay da entrada à saída
- Contamination delay: t_{cd} = min delay da entrada à saída



Delay: Propagação e Contaminação

- *Os atrasos são causados por*
  - *Capacitância e resistências no circuito*
  - *Speed of light limitation*
- Rasões porque t_{pd} and t_{cd} podem ser diferentes:
  - Diferentes tempos de subida (*rising*) e de decida (*falling*)
  - Múltiplas entradas e saídas, algumas podem ser mais rápidas do que as outras
  - Circuito mais lento quando quente e mais rápido quando frio

Caminhos: Críticos e Curtos

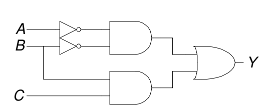


Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$
Short Path: $t_{cd} = t_{cd\_AND}$

Glitches

- Um glitch ocorre quando uma mudança em uma entrada causa múltiplas mudanças na saída
- Glitches não causam problemas se seguirmos as convenções de projetos síncronos
- É importante reconhecer um glitch quando se vê um em uma simulação ou em um osciloscópio

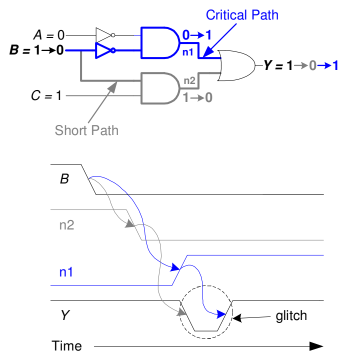
Exemplo de Glitch



C \ AB	00	01	11	10
0	1	0	0	0
1	1	1	1	0

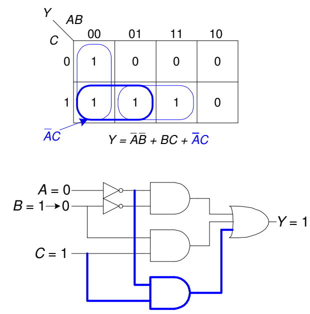
$Y = \overline{A}\,\overline{B} + BC$

Exemplo de Glitch (cont.)



MC542
3.91

Exemplo de Glitch (cont.)



MC542
3.92