

  
Universidade Federal de Santa Catarina

## EEL7020 – Sistemas Digitais

### Aula 4: Circuitos combinacionais - (De-)Multiplexadores e (De-)Codificadores

Prof. Djones Vinicius Lettnin  
lettnin@eel.ufsc.br  
<http://lettnin.paginas.ufsc.br/>

Disclaimer: slides adapted for EEL7020 by D. Lettnin from the original slides made available by the authors E. Batista, J. Güntzel and J. Fraga.

  
Universidade Federal de Santa Catarina

### Revisão

**Soma de mintermos**

$$\begin{array}{|c|c|c|c|} \hline A & B & C & F \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline \end{array}$$

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC$$

$$F = m_2 + m_3 + m_5 + m_6$$

$$F = \sum(2,3,5,6)$$

**Produto de maxtermos**

$$F = (A + B + C) \cdot (A + B + \bar{C}) \cdot (\bar{A} + B + C) \cdot (\bar{A} + \bar{B} + \bar{C})$$

$$F = M_0 \cdot M_1 \cdot M_4 \cdot M_7$$

$$F = \prod(0,1,4,7)$$

© J. Güntzel – Adapted by D. Lettnin

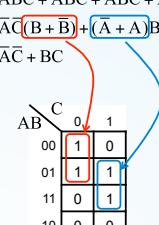
2

  
Universidade Federal de Santa Catarina

### Revisão

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$S = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC$   
 $= \bar{A}\bar{C}(B + \bar{B}) + (\bar{A} + A)BC$   
 $= \bar{A}\bar{C} + BC$



© E. Batista – Adapted by D. Lettnin

3

  
Universidade Federal de Santa Catarina

### Plano de Aula

- Circuitos combinacionais:
  - (De-)Multiplexadores
  - (De-)Codificadores



© J. Güntzel – Adapted by D. Lettnin

4

  
Universidade Federal de Santa Catarina

## Circuitos Combinacionais

### - Características

- São circuitos nos quais as saídas dependem somente das entradas



- Podem conter diversas saídas, cada uma regida por uma equação lógica distinta
- Porém, tais equações podem, eventualmente, compartilhar termos. Neste caso, o compartilhamento de partes do circuito conduz a um circuito de menor custo.

© J. Güntzel – Adapted by D. Lettnin

5

  
Universidade Federal de Santa Catarina

## Circuitos Combinacionais

### - Passos para o Projeto

1. Estudo minucioso do problema e identificação de uma solução, em termos de circuito digital (até aqui, apenas combinacional)
2. Construção da tabela-verdade e do(s) Mapa(s) de Karnaugh
3. Minimização da(s) saída(s) usando o Mapa de Karnaugh. No caso de mais de uma saída, tentar compartilhar termos
4. Realizar o mapeamento tecnológico, considerando a biblioteca de portas disponível

OBS: quando muitas variáveis envolvidas (mais de 5 vars. de entrada) e/ou muitas saídas, utilizar ferramentas de Computer-aided design (CAD) que realizem os passos 2, 3 e 4

© J. Güntzel – Adapted by D. Lettnin

6

**Circuitos Combinacionais**

**- Tipos de Circuitos Combinacionais**

- Um circuito combinacional pode ser classificado segundo sua aplicação:
  - Circuitos de interconexão:** seletores (também chamados de multiplexadores), decodificadores e codificadores
  - Circuitos lógico-aritméticos:** somadores, subtraidores, somadores/subtraidores, multiplicadores, deslocadores, comparadores e ULAs (circuitos que combinam mais de duas operações aritméticas e/ou lógicas).

© J. Guntzel – Adapted by D. Lettmann

7

**Multiplexadores**

- Também conhecido como seletor de dados
- Permite que apenas uma entrada é cada vez disponível na saída
- Aplicações:
  - Seleção de dados
  - Roteamento
  - Conversão paralelo/série
  - Implementação de tabela verdade

© E. Batista – Adapted by D. Lettmann

**Multiplexor (Mux)**

- Mux: Another popular combinational building block
  - Routes one of its N data inputs to its one output, based on binary value of select inputs
    - 4 input mux → needs 2 select inputs to indicate which input to route through
    - 8 input mux → 3 select inputs
    - N inputs →  $\log_2(N)$  selects
  - Like a railyard switch

© Vahid – Adapted by D. Lettmann

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 2:1 (ou seletor 2:1) "Sua função é selecionar uma dentre as duas entradas de dados, fazendo a entrada selecionada aparecer na saída"

sel	A	B	Y
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

© J. Guntzel – Adapted by D. Lettmann

10

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 2:1 Implementação com portas lógicas básicas

Y	$\bar{A}$	$\bar{B}$	A	B	$\bar{A}$	$\bar{B}$	sel
0	0	1	1	0	0	1	0
0	1	1	0	0	1	0	1

$Y = \overline{\text{sel-A}} + \overline{\text{sel-B}}$

© J. Guntzel – Adapted by D. Lettmann

11

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 2:1 (também chamado de seletor 2:1) Outra maneira de enxergar a tabela-verdade do mux 2:1

sel	A	B	Y
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

$$Y = \overline{\text{sel-A}} + \overline{\text{sel-B}}$$

© J. Guntzel – Adapted by D. Lettmann

12

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 4:1 (ou seletor 4:1) "Sua função é selecionar uma dentre quatro entradas"

sel1	sel0	Y
0	0	A
0	1	B
1	0	C
1	1	D

**simbolo**

$$Y = sel1 \cdot sel0 \cdot A + sel1 \cdot sel0 \cdot B + sel1 \cdot sel0 \cdot C + sel1 \cdot sel0 \cdot D$$

© J. Guntzel – Adapted by D. Lettmin

13

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 4:1 (ou seletor 4:1) Implementação como associação de muxes 2:1

sel1	sel0	Y
0	0	A
0	1	B
1	0	C
1	1	D

**A princípio, qualquer estrutura vista para mux 2:1 pode ser usada**

© J. Guntzel – Adapted by D. Lettmin

14

**Circuitos Combinacionais**

**- Multiplexadores (ou seletores)**

- Multiplexador 4:1 (ou seletor 4:1) Implementação com portas lógicas básicas

$$Y = sel1 \cdot sel0 \cdot A + sel1 \cdot sel0 \cdot B + sel1 \cdot sel0 \cdot C + sel1 \cdot sel0 \cdot D$$

© J. Guntzel – Adapted by D. Lettmin

15

**Multiplexador 4:1 em VHDL**

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

Entity mux_4_1_when_else is
  Port(A, B, C, D : IN BIT;
       sel1, sel2 : IN BIT;
       s0 : OUT BIT);
end mux_4_1_when_else;

Architecture teste of mux_4_1_when_else is
begin
  s0 <= A WHEN sel1 = '0' AND sel2 = '0' ELSE
    B WHEN sel1 = '0' AND sel2 = '1' ELSE
    C WHEN sel1 = '1' AND sel2 = '0' ELSE
    D;
end teste;
  
```

© J. Guntzel – Adapted by D. Lettmin

16

**Demultiplexadores**

- Também conhecidos como distribuidores de dados

SELECT code			OUTPUTS							
S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	O <sub>0</sub>	O <sub>1</sub>	O <sub>2</sub>	O <sub>3</sub>	O <sub>4</sub>	O <sub>5</sub>	O <sub>6</sub>	O <sub>7</sub>
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

© E. Batista – Adapted by D. Lettmin

**Demultiplexadores**

- Exemplo: Demux de 8 saídas

SELECT code			OUTPUTS							
S <sub>2</sub>	S <sub>1</sub>	S <sub>0</sub>	O <sub>0</sub>	O <sub>1</sub>	O <sub>2</sub>	O <sub>3</sub>	O <sub>4</sub>	O <sub>5</sub>	O <sub>6</sub>	O <sub>7</sub>
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$O_0 = I \cdot (\bar{S}_2 \bar{S}_1 \bar{S}_0)$

$O_1 = I \cdot (\bar{S}_2 \bar{S}_1 S_0)$

$O_2 = I \cdot (\bar{S}_2 S_1 \bar{S}_0)$

$O_3 = I \cdot (S_2 \bar{S}_1 \bar{S}_0)$

$O_4 = I \cdot (S_2 \bar{S}_1 S_0)$

$O_5 = I \cdot (S_2 S_1 \bar{S}_0)$

$O_6 = I \cdot (S_2 S_1 S_0)$

$O_7 = I \cdot (S_2 S_1 S_0)$

© E. Batista – Adapted by D. Lettmin

**De-Multiplexador 4:1 em VHDL**

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity demux1_4 is
port (
    out0 : out std_logic; --output bit
    out1 : out std_logic; --output bit
    out2 : out std_logic; --output bit
    out3 : out std_logic; --output bit
    sel : in std_logic_vector(1 downto 0);
    bitin : in std_logic --input bit
);
end demux1_4;

architecture Behavioral of demux1_4 is
begin
process (bitin,sel)
begin
case sel is
when "00" => out1 <= bitin; out0 <= '0'; out2 <= '0'; out3 <='0';
when "01" => out1 <= bitin; out0 <= '0'; out2 <= '0'; out3 <='0';
when "10" => out2 <= bitin; out0 <= '0'; out1 <= '0'; out3 <='0';
when others => out3 <= bitin; out0 <= '0'; out1 <= '0'; out2 <='0';
end case;
end process;
end Behavioral;

```

19

**Plano de Aula**

- Circuitos combinacionais:
  - (De-)Multiplexadores
  - (De-)Codificadores

20

**Circuitos Combinacionais**

### - Decodificadores

Decodificador 2:4

- Sua função é ativar uma e somente uma dentre as 4 saídas, de acordo com a combinação de valores das entradas
- Ativar, neste caso, quer dizer diferenciar, destacar
- Existe uma relação entre o número de saídas (ns) e o número de entradas (ne):

$$ns = 2^{ne}$$

21

**Circuitos Combinacionais**

### - Decodificadores

Decodificador 2:4 Tabela-verdade e símbolo

		saídas					
		A1	A0	S0	S1	S2	S3
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0
1	1	0	0	0	1	0	0

símbolo

22

**Circuitos Combinacionais**

### - Decodificadores

Decodificador 2:4

- Cada combinação de entrada pode ser vista como o endereço de uma saída específica

		saídas					
		A1	A0	S0	S1	S2	S3
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0
1	1	0	0	0	1	0	0

endereço da saída  
 (=2 em decimal)

saída S2 ativada...

23

**Circuitos Combinacionais**

### - Decodificadores

Decodificador 2:4

- Cada uma das 4 saídas corresponde a um mintermo diferente.

		saídas					
		A1	A0	S0	S1	S2	S3
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	0
1	1	0	0	0	1	0	0

Implementação independente de tecnologia

24

**Circuitos Combinacionais**

**- Decodificadores**

**Decodificador 2:4**

- Implementação em utilizando NAND e Inversor

		saídas			
		S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>
A <sub>1</sub> A <sub>0</sub>		0 0	1 0	0 1	0 0
0 0	0 0	1	0	0	0
0 1	0 1	0	1	0	0
1 0	1 0	0	0	1	0
1 1	1 1	0	0	0	1

25

© J. Güntzel – Adapted by D. Lettmann

**Circuitos Combinacionais**

**- Decodificadores**

**Decodificador 2:4**

- Com saídas em lógica invertida (ou complementar). A ativação se dá com o valor lógico 0

		saídas			
		S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>
A <sub>1</sub> A <sub>0</sub>		0 0	0 1	1 0	1 1
0 0	0 0	1	0	1	1
0 1	0 1	0	1	0	1
1 0	1 0	1	0	1	0
1 1	1 1	1	1	1	0

26

© J. Güntzel – Adapted by D. Lettmann

**Circuitos Combinacionais**

**- Decodificadores**

**Decodificador 2:4**

- Com saídas em lógica invertida (ou complementar)

		saídas			
		S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>
A <sub>1</sub> A <sub>0</sub>		0 0	0 1	1 1	1 1
0 0	0 0	0	1	1	1
0 1	0 1	1	0	1	1
1 0	1 0	1	1	0	1
1 1	1 1	1	1	1	0

27

© J. Güntzel – Adapted by D. Lettmann

**Circuitos Combinacionais**

**- Decodificadores**

**Decodificador 2:4**

- Acrescentando uma entrada de habilitação (enable)

		saídas			
		S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>
H A <sub>1</sub> A <sub>0</sub>		0 X X	0 0 0	0 0 0	0 0 0
0 X X	0 0 0	0	0	0	0
1 0 0	1 0 0	1	0	0	0
1 0 1	1 0 1	0	1	0	0
1 1 0	1 1 0	0	0	1	0
1 1 1	1 1 1	0	0	0	1

28

© J. Güntzel – Adapted by D. Lettmann

**Decodificador 2:4 em VHDL**

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY dec2to4 IS
  PORT (A : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
        H : IN STD_LOGIC ;
        S : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
END dec2to4 ;

ARCHITECTURE Behavior OF dec2to4 IS
  SIGNAL HA: STD_LOGIC_VECTOR(2 DOWNTO 0) ;
BEGIN
  HA <= H & A ;
  WITH Ena SELECT
    S <=
      "1000" WHEN "100",
      "0100" WHEN "101",
      "0010" WHEN "110",
      "0001" WHEN "111",
      "0000" WHEN OTHERS ;
END Behavior ;

```

29

**Circuitos Combinacionais**

**- Decodificadores**

**Decodificador 2:4**

- Implementação com NAND e Inversor

		saídas			
		S <sub>0</sub>	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>
H A <sub>1</sub> A <sub>0</sub>		0 X X	0 0 0	0 0 0	0 0 0
0 X X	0 0 0	0	0	0	0
1 0 0	1 0 0	1	0	0	0
1 0 1	1 0 1	0	1	0	0
1 1 0	1 1 0	0	0	1	0
1 1 1	1 1 1	0	0	0	1

30

© J. Güntzel – Adapted by D. Lettmann

**Circuitos Combinacionais**

### - Decodificadores

**Decodificador 2:4**

- Com saídas em lógica invertida (ou complementar) e com entrada de habilitação (*enable*)

entradas			saídas			
H	A1	A0	S0	S1	S2	S3
0	X	X	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

© J. Güntzel – Adapted by D. Lettmann

31

**Circuitos Combinacionais**

### - Decodificadores

**Decodificador 2:4**

- Com saídas em lógica invertida Implementação com NANDs e Inversor

entradas			saídas			
H	A1	A0	S0	S1	S2	S3
0	X	X	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

© J. Güntzel – Adapted by D. Lettmann

32

**Circuitos Combinacionais**

### - Decodificadores

- Decodificadores 3:8, 4:16, 5:32 etc
  - Seguem o mesmo princípio dos decodificadores vistos, sempre observando a relação  $n:2^n$  (número de entradas: número de saídas)
  - Também se pode “montar” um decodificador a partir de decodificadores menores, que possuam entrada de habilitação

© J. Güntzel – Adapted by D. Lettmann

33

**Circuitos Combinacionais**

### - Decodificadores

- Um Decodificador 3:8, sem entrada de habilitação

© J. Güntzel – Adapted by D. Lettmann

34

**Codificadores**

- Operação contrária ao dos decodificadores
- Codificadores servem para reduzir o número de bits necessários para a representação de alguma informação (facilitando sua manipulação e seu armazenamento)
- Os principais tipos de codificadores são: binários,e de prioridade.

© E. Batista – Adapted by D. Lettmann

**Codificador Binário 4:2**

**Codificador Binário 4:2**

- Apenas as situações de entrada contendo somente uma posição valendo 1 são consideradas
- As demais situações são tratadas como *don't cares*

entradas				saídas	
A3	A2	A1	A0	S1	S0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

36

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

### DECODIFICADOR PARA 7 SEGMENTOS

- Os decodificadores para 7 segmentos são utilizados em instrumentos digitais que tenham uma saída numérica.
- Ex: Multímetros, frequêncimetros, etc...
- Os displays podem ser LED'S (diodo emissor de luz), LCD'S (crystal líquido), ou lâmpadas incandescente.

LED'S anôdio comum      LED'S catôdo comum

© J. Fraga – Adapted by D. Lettin

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

- Construir um conversor BCD / 7 segmentos com saídas ativo alto

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	0	0	1	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	0	1	1	0	1	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1

$a = m_0 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8 + m_9$   
 $b = m_0 + m_1 + m_2 + m_3 + m_4 + m_7 + m_8 + m_9$   
 $c = m_0 + m_1 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8 + m_9$   
 $d = m_0 + m_2 + m_3 + m_5 + m_6 + m_8 + m_9$   
 $e = m_0 + m_2 + m_6 + m_8$   
 $f = m_0 + m_4 + m_5 + m_6 + m_8 + m_9$   
 $g = m_2 + m_3 + m_4 + m_5 + m_6 + m_8 + m_9$

© J. Fraga – Adapted by D. Lettin

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

- Com base na tabela verdade temos que construir um mapa de Karnaugh para cada uma das saídas do 7 segmentos.

$DC \quad BA$   
 $a = m_0 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8 + m_9$   
 $a = B + D + C.A + \bar{C}.\bar{A}$

© J. Fraga – Adapted by D. Lettin

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

$DC \quad BA$   
 $b = m_0 + m_1 + m_2 + m_3 + m_4 + m_7 + m_8 + m_9$   
 $b = \bar{C} + \bar{B}\bar{A} + BA$

© J. Fraga – Adapted by D. Lettin

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

$c = m_0 + m_1 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8 + m_9$   
 $DC \quad BA$   
 $c = A + \bar{B} + C$

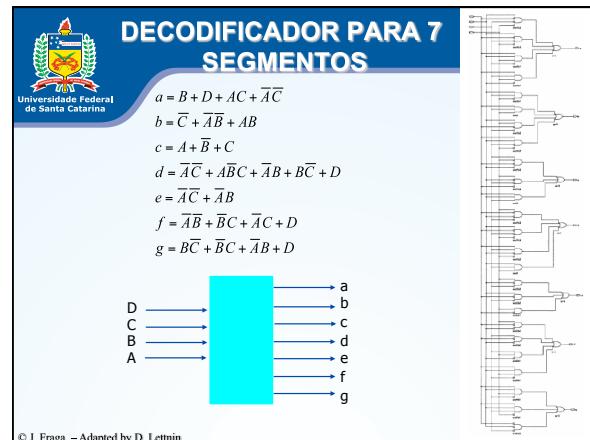
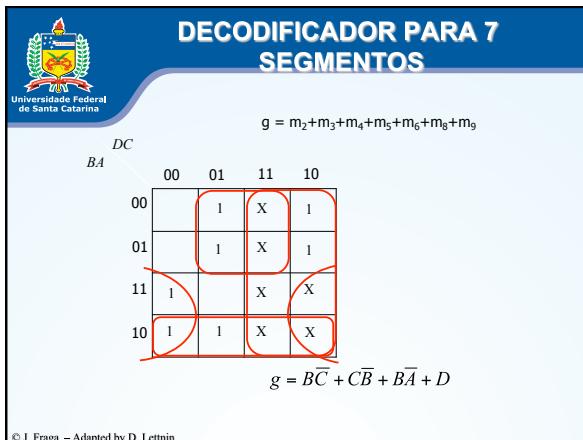
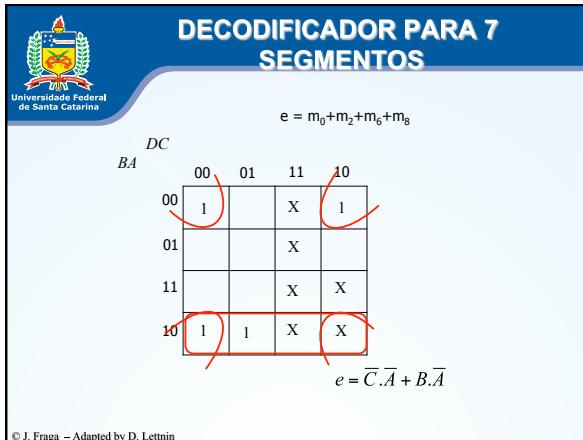
© J. Fraga – Adapted by D. Lettin

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## DECODIFICADOR PARA 7 SEGMENTOS

$DC \quad BA$   
 $d = m_0 + m_2 + m_3 + m_5 + m_6 + m_8 + m_9$   
 $d = \bar{C}.\bar{A} + \bar{C}\bar{B}A + \bar{B}\bar{A} + \bar{C}B + D$

© J. Fraga – Adapted by D. Lettin



**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

## EEL7020 – Sistemas Digitais

### Aula 4: Circuitos combinacionais - (De-)Multiplexadores e (De-)Codificadores

Prof. Djones Vinicius Lettnin  
lettnin@eel.ufsc.br  
<http://lettnin.paginas.ufsc.br/>

Disclaimer: slides adapted for EEL7020 by D. Lettnin from the original slides made available by the authors E. Batista, J. Güntzel and J. Fraga.