

# Lógica e Sistemas Digitais - 4

## Módulos Combinatórios

### MSI

ISEL

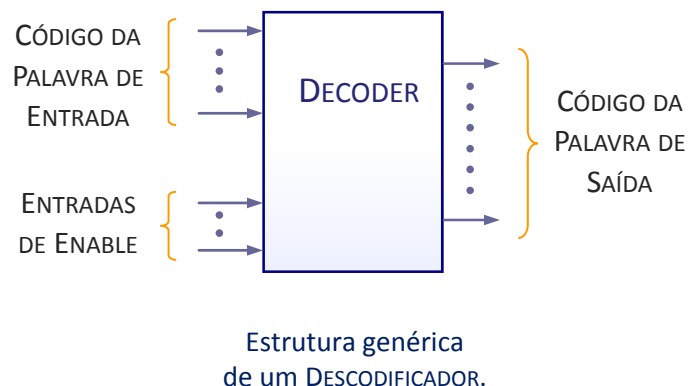
Departamento de Engenharia de Electrónica  
e Telecomunicações e de Computadores  
Lisboa

Mário Araújo

2016-1

## DESCODIFICADOR (DECODER)

4-2



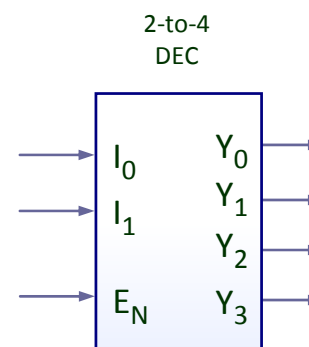
ENTRADAS			SAÍDAS				EQUAÇÕES DAS SAÍDAS
$E_N$	$I_1$	$I_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$	
0	–	–	0	0	0	0	
1	0	0	0	0	0	1	$Y_0 = m_0 E_N$
1	0	1	0	0	1	0	$Y_1 = m_1 E_N$
1	1	0	0	1	0	0	$Y_2 = m_2 E_N$
1	1	1	1	0	0	0	$Y_3 = m_3 E_N$

Tabela de verdade de um 2-to-4 DECODER.

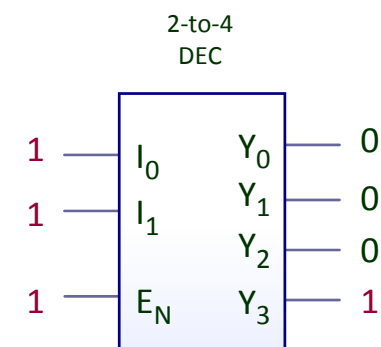
Um DESCODIFICADOR (DECODER) é um circuito combinatório que converte entradas codificadas em saídas codificadas. O número de bits do código de entrada é normalmente inferior ao número de bits do código de saída.

Um decodificador é tipicamente um circuito  $n$ -to- $2^n$  (por exemplo 2-to-4 ou 3-to-8, mas pode ser 4-to-16).

Um decodificador activa **uma e só uma** das  $2^n$  saídas de cada vez perante uma combinação de  $n$ -bits de entrada. A saída activada identifica a palavra codificada aplicada às entradas.

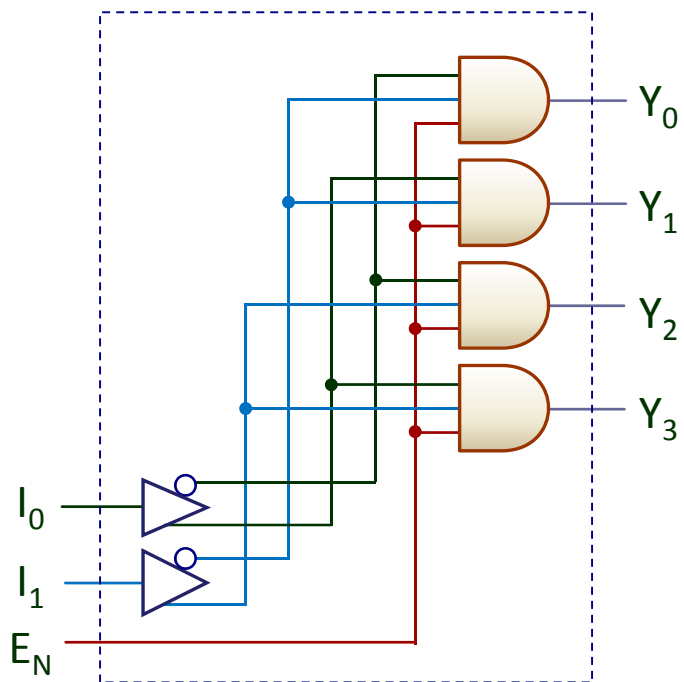
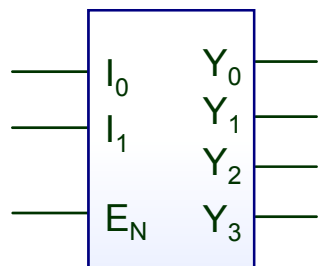


Símbolo lógico de um 2-to-4 DECODER.

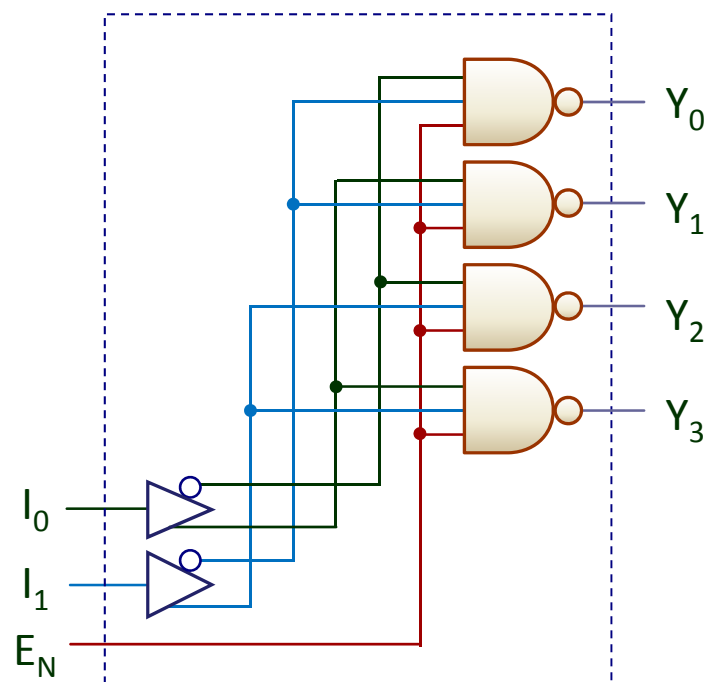
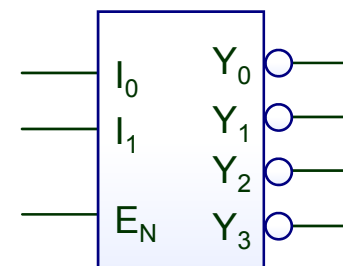


Configuração correspondente à activação da saída  $Y_3$ .

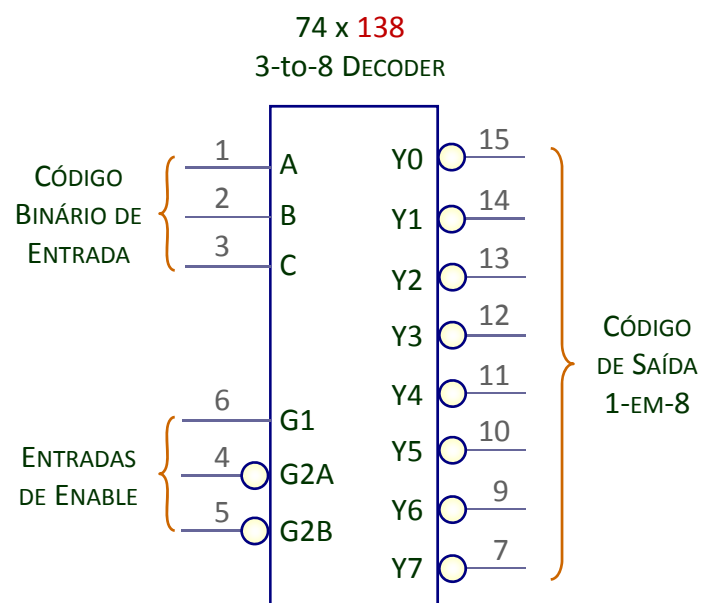




Símbolo lógico (em cima) e estrutura interna de um 2-to-4 DECODER com saídas do tipo ACTIVE-HIGH.



Símbolo lógico (em cima) e estrutura interna de um 2-TO-4 DECODER com saídas do tipo ACTIVE-LOW.



Símbolo e configuração dos pinos de um decodificador de 3 entradas e 8 saídas (16 pinos).

ENTRADAS						SAÍDAS							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	-	-	-	-	-	1	1	1	1	1	1	1	1
-	1	-	-	-	-	1	1	1	1	1	1	1	1
-	-	1	-	-	-	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

Tabela de verdade do decodificador 74 x 138.

As 3 entradas de ENABLE estão ligadas em AND – duas são ACTIVE-LOW (G2A e G2B) e uma é ACTIVE-HIGH (G1).

Basta que uma das entradas de ENABLE não esteja activa para que o decodificador fique com todas as suas saídas desactivadas (Y0\_L a Y7\_L ao valor lógico-1, pois as saídas são do tipo ACTIVE-LOW – o decodificador não funciona nestas condições ).

## DESCODIFICADOR 3-TO-8 EM PAL 22V10

4-5

```

Name DECODER;
...
Device    p22v10;

/* Decoder 3x8 */
/* **** PIN Declaration *** */

PIN      [1..3] = [A0..2] ;
PIN      9 = G1 ;
PIN      10 = !G2A ;
PIN      11 = !G3A ;

PIN [23..16] = [Y0..7] ;
FIELD sel = [A0..2] ;
enable = G1&G2A&G3A ;

/* Functional Declaration * */
Condition {
if enable & sel:0 out !Y0 ;
if enable & sel:1 out !Y1 ;
if enable & sel:2 out !Y2 ;
if enable & sel:3 out !Y3 ;
if enable & sel:4 out !Y4 ;
if enable & sel:5 out !Y5 ;
if enable & sel:6 out !Y6 ;
if enable & sel:7 out !Y7 ;
}
    
```

Troço de código CUPL para um DECODER 3x8 utilizando o comando CONDITION.

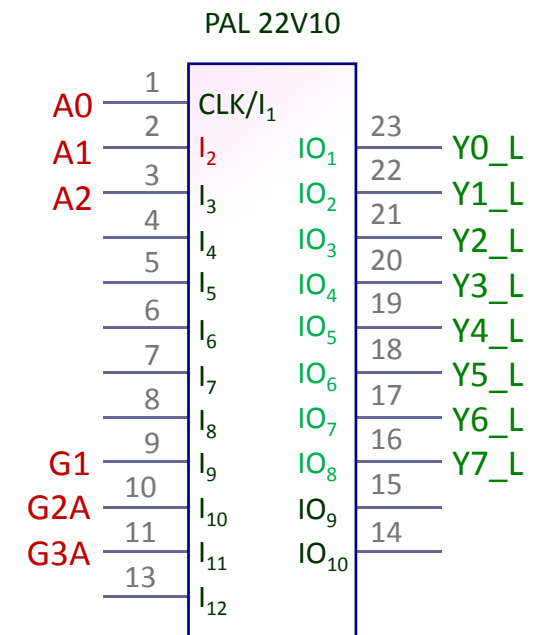


Símbolo e configuração dos pinos do DECODIFICADOR de 3 entradas e 8 saídas (16 pinos).

```

/* ** Functional declaration ** */
$repeat i = [0..7]
    !Y{i} = sel:{i} & enable ;
$repend
    
```

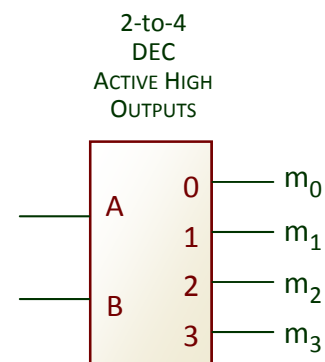
Variante do troço de código CUPL para um DECODER 3x8 utilizando o comando REPEAT.



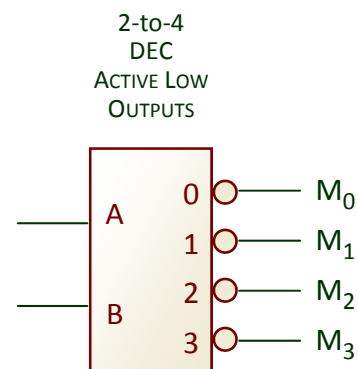
Símbolo lógico da PAL assinalando os pinos de entrada e saída utilizados para a implementação do DECODIFICADOR 3x8 ao lado.

Em alternativa é possível usar uma declaração funcional do DECODIFICADOR mais compacta fazendo uso do comando de pré processamento REPEAT em vez do comando CONDITION.





Gerador de Mintermos.



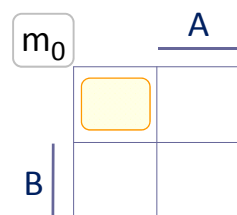
Gerador de Maxtermos.

	<u>A(1)</u>	
	$m_0$	$m_1$
<u>B(2)</u>	$m_2$	$m_3$

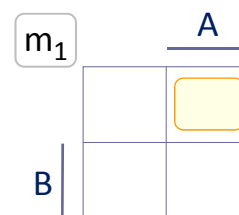
	<u>A(1)</u>	
	$M_0'$	$M_1'$
<u>B(2)</u>	$M_2'$	$M_3'$

m – MINTERMOS  
M – MAXTERMOS

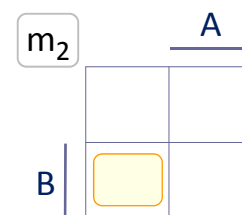
$$m_j = M_j'$$



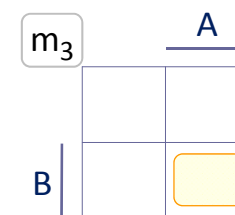
$$m_0 = A'B'$$



$$m_1 = AB'$$

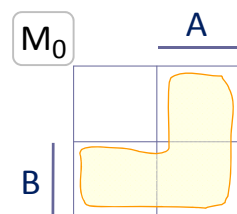


$$m_2 = A'B$$

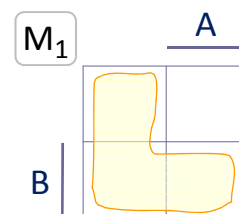


$$m_3 = AB$$

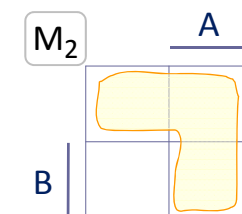
Mapas de Karnaugh correspondentes à geração de Mintermos de um DECODER 2-to-4 com saídas ACTIVE-HIGH.



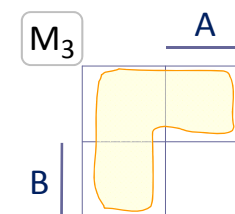
$$M_0 = m_0' = A+B$$



$$M_1 = m_1' = A' + B$$



$$M_2 = m_2' = A+B'$$



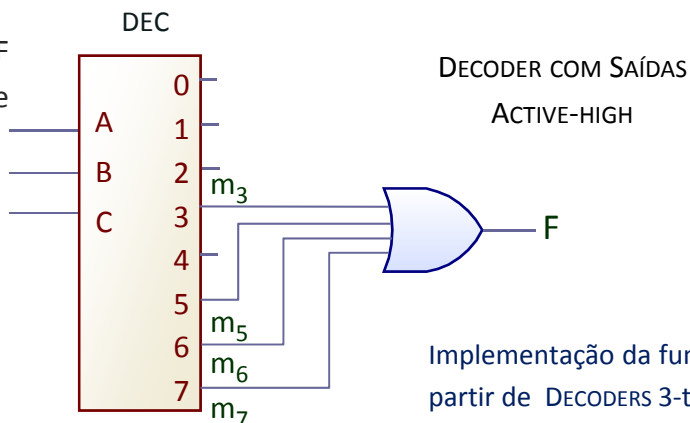
$$M_3 = m_3' = A' + B'$$

Mapas de Karnaugh correspondentes à geração de Maxtermos de um DECODER 2-to-4 com saídas ACTIVE-LOW.

Os decodificadores permitem a implementação de funções combinatórias – cada saída de um decodificador corresponde a um mintermo se for do tipo ACTIVE-HIGH, e a um maxtermo se for do tipo ACTIVE-LOW.

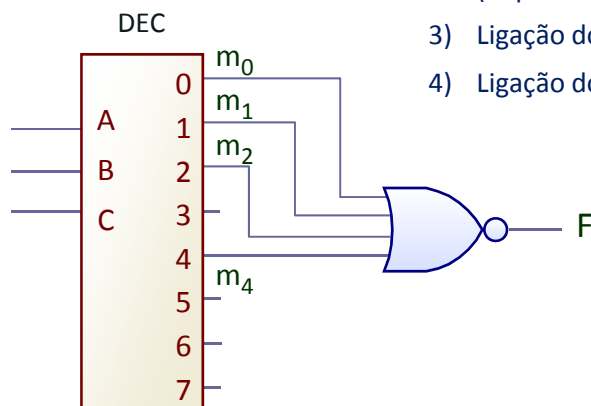
Implementação de F através da soma de **mintermos**.

1



$$F = \sum m(3, 5, 6, 7) = m_3 + m_5 + m_6 + m_7$$

2

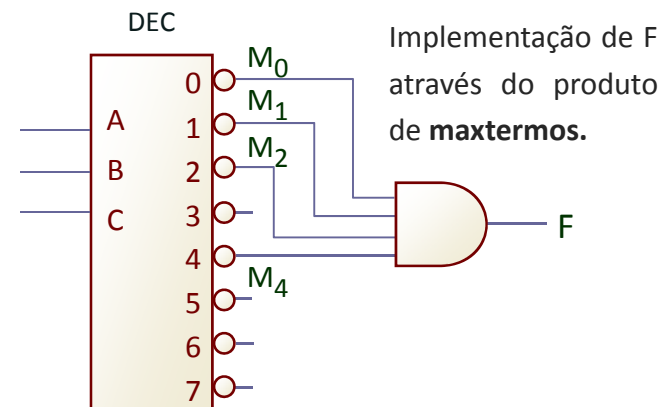


$$F = \{ \sum m(0, 1, 2, 4) \}' = (m_0 + m_1 + m_2 + m_4)'$$

Implementação de F através de  $(F')'$  – soma negada dos mintermos que não aparecem em F.

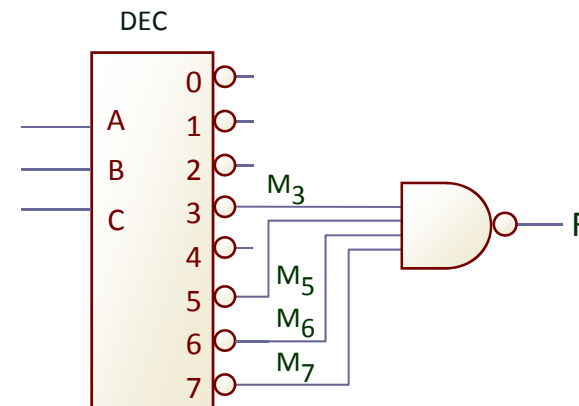
DECODER COM SAÍDAS ACTIVE-LOW

3



$$F = \prod M(0, 1, 2, 4) = M_0 \cdot M_1 \cdot M_2 \cdot M_4$$

4



$$F = \{ \prod M(3, 5, 6, 7) \}' = (M_3 \cdot M_5 \cdot M_6 \cdot M_7)'$$

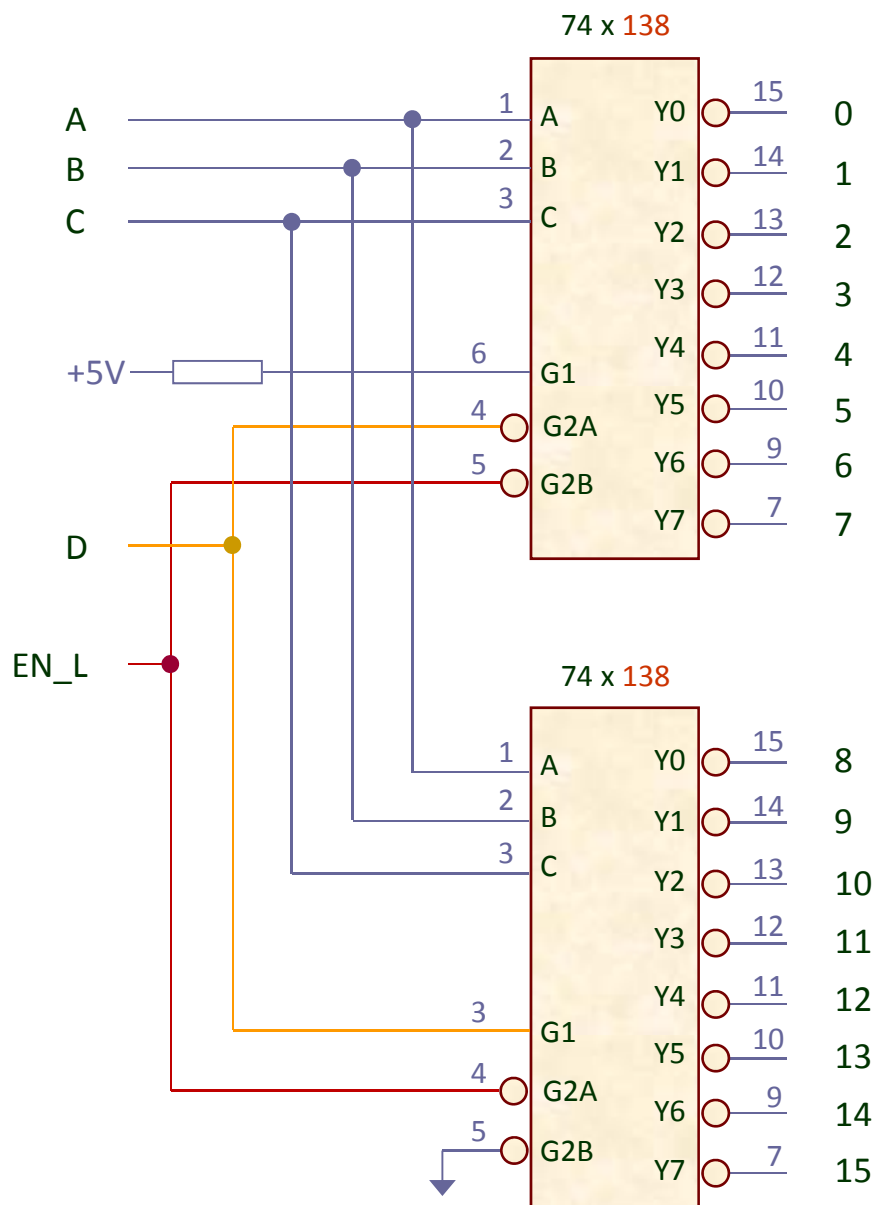
Implementação de F através de  $(F')'$  – produto negado dos maxtermos que não aparecem em F.

Implementação da função Maioria a 3 variáveis a partir de DECODERS 3-to-8:

- 1) Ligação dos mintermos em OR.
- 2) Ligação dos mintermos de  $F'$  em NOR (implementação de F a partir de  $(F')'$ ).
- 3) Ligação dos maxtermos em AND.
- 4) Ligação dos maxtermos de  $F'$  em NAND.

## DESCODIFICADOR DE 16 SAÍDAS – EXPANSÃO A PARTIR DE DESCODIFICADORES DE 8 SAÍDAS

4-8



Vários decodificadores podem ser concatenados de forma hierárquica para a descodificação de palavras de código de maior comprimento.

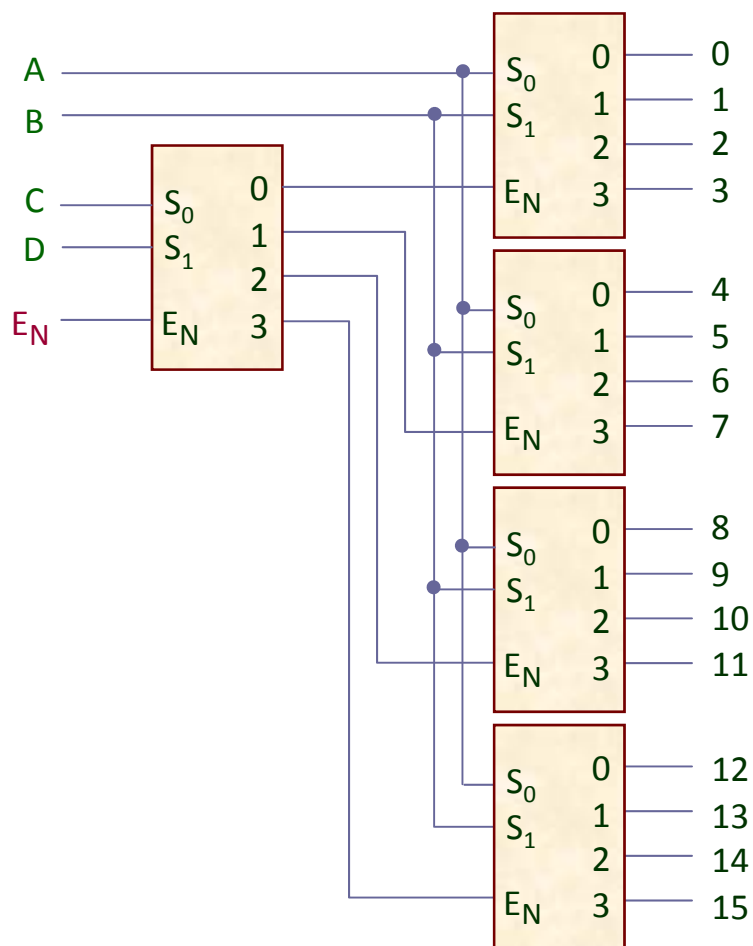
O decodificador em cima fica activo (ENABLED) quando D=0 (números de 0 a 7), e o decodificador em baixo fica activo quando D=1 (números de 8 a 15).

DECODER 4-TO-16 construído a partir de dois DECODERS 3-TO-8.



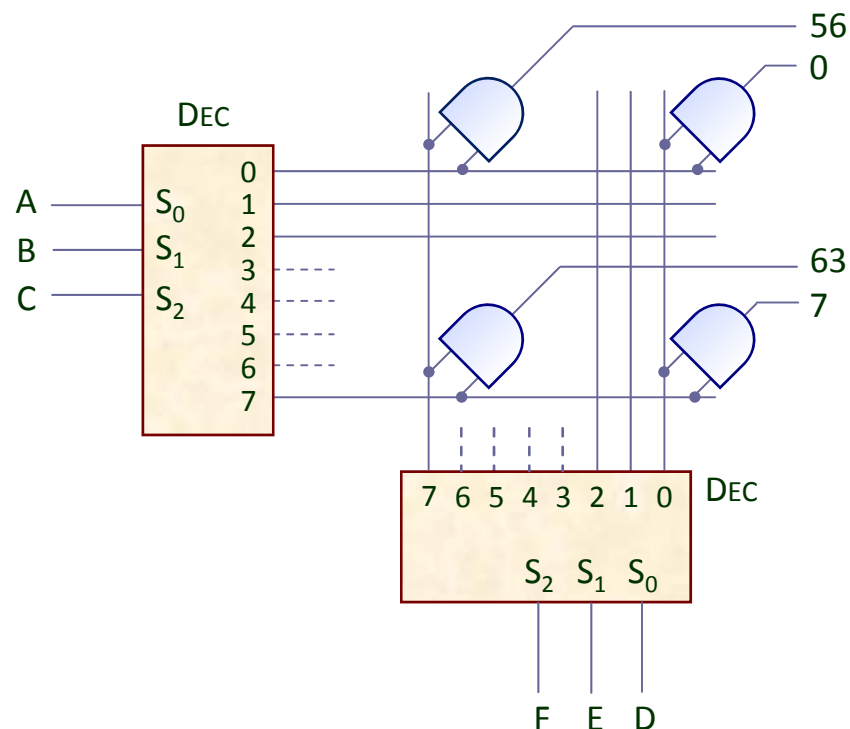
## DESCODIFICADORES DE 16 E DE 64 SAÍDAS – FORMAS DE EXPANSÃO

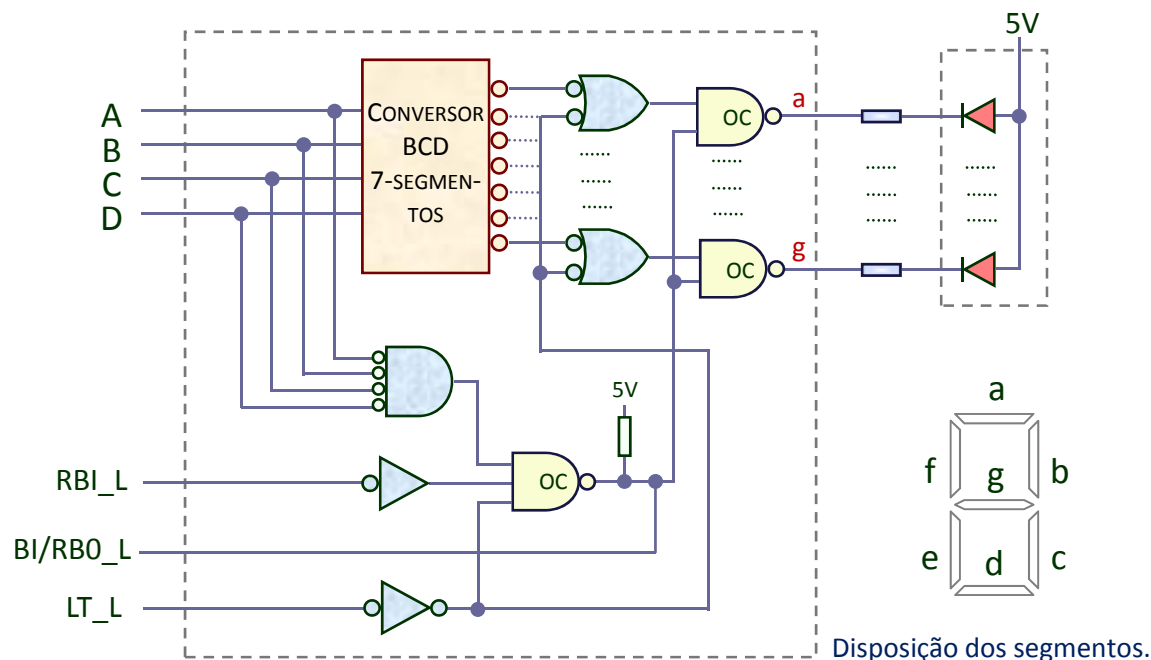
4-9



DECODER 4-TO-16 construído a partir de  
DECODERS 2-TO-4 fazendo uso dos ENABLES.

Um decodificador 4-to-16 pode também ser construído a partir de  
descodificadores 2-to-4 em vez de 3-to-8 como no slide anterior.



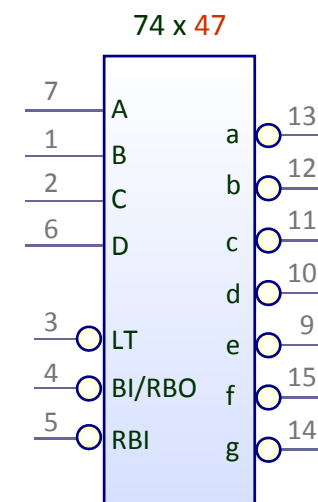


Estrutura interna de um DECODER/DRIVER típico como o 74x47, individualizando a conversão BCD-7 segmentos da lógica associada aos sinais de controlo.

**LT\_L:** LAMP TEST (ilumina todos os segmentos).

**BI/RBO\_L:** BLANKING INPUT/RIPPLE BLANKING OUTPUT (extingue os segmentos se estiver 0 presente à entrada).

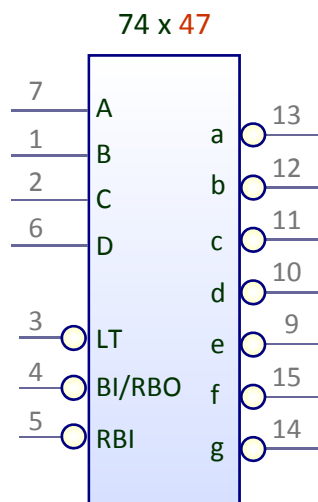
**RBI\_L:** RIPPLE BLANKING Input.



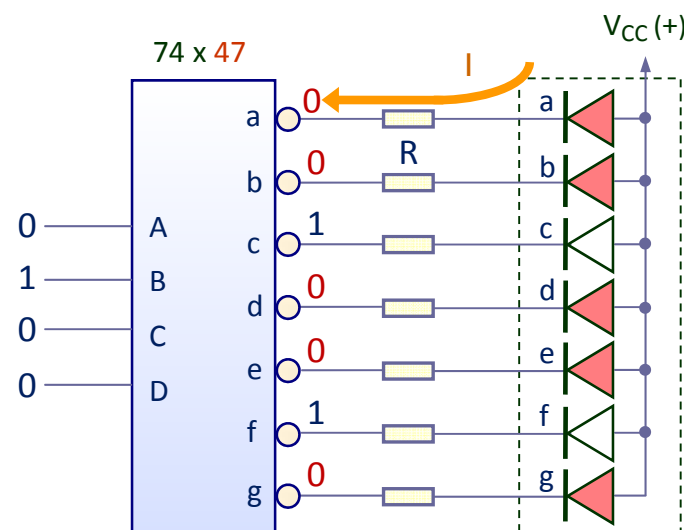
Símbolo lógico e configuração dos pinos de um BCD-to-7-Segment DECODER/DRIVER com saídas do tipo ACTIVE-LOW (16 pinos).

Um decodificador de 7-segmentos tem uma entrada de 4-bits A, B, C e D em código BCD e uma saída de 7-segmentos (pelo que é designado também **CONVERSOR DE CÓDIGO**), e ainda sinais de controlo vocacionados para a concatenação como os descritos à esquerda.

A referência OC significa que as portas NAND são na família TTL de tipo OPEN COLLECTOR.

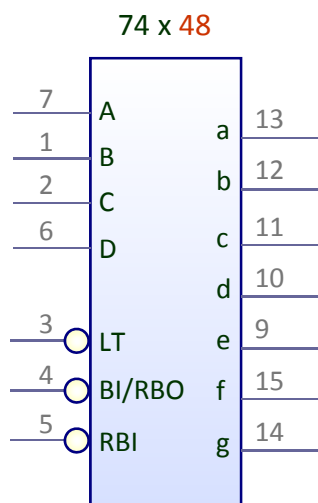


BCD-to-7-Segment  
**DECODER/DRIVER**  
(ACTIVE-LOW outputs)

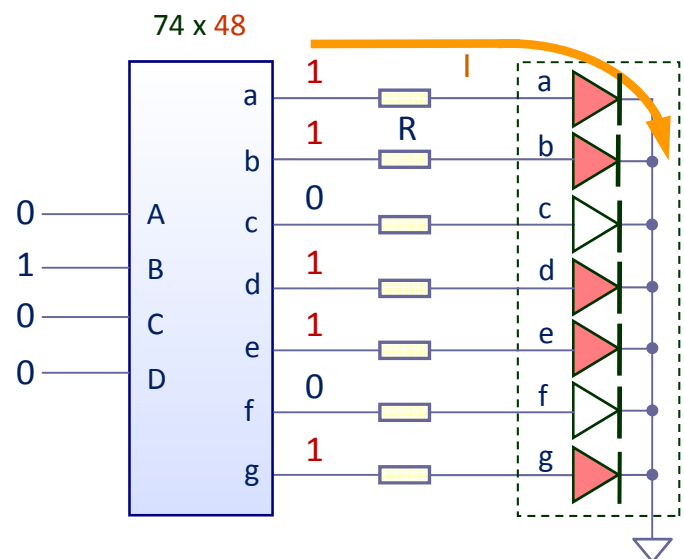


Modo de ligação de um DISPLAY  
de 7-segmentos ACTIVE-LOW  
(ÂNODO COMUM).

COMMON ANODE  
7-segment display



BCD-to 7-Segment  
**DECODER/DRIVER**  
(ACTIVE-HIGH outputs)

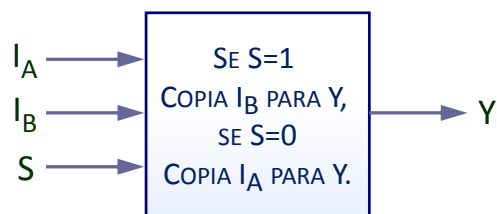


A combinação de LEDS iluminados (a  
vermelho) representa o número 2  
colocado à entrada em código BCD.

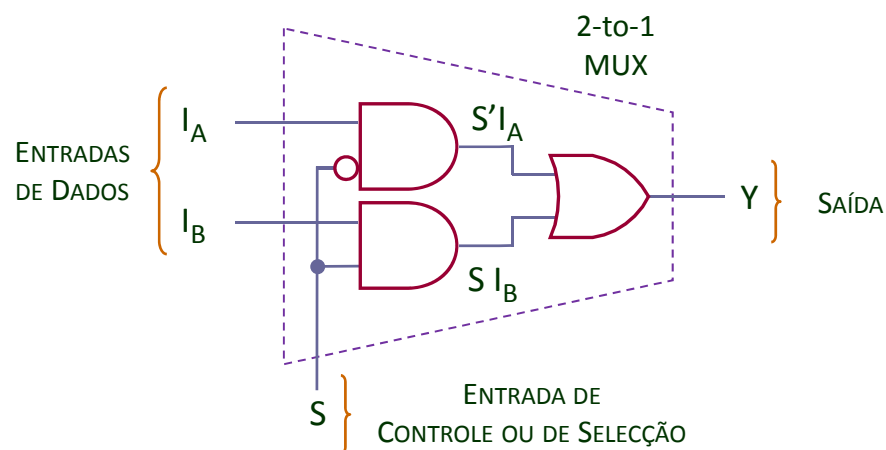
Modo de ligação de um DISPLAY  
de 7-segmentos ACTIVE HIGH  
(CÁTODO COMUM).

COMMON CATHODE  
7-segment display

Ligações de um DECODER/DRIVER a um display de 7-segmentos.



Especificação de um MULTIPLEXER 2-TO-1 em Diagrama de Blocos.



Uma implementação possível para o MULTIPLEXER 2-TO-1.

O MULTIPLEXER é um circuito combinatório que permite, através da especificação de sinais de selecção, efectuar o encaminhamento de uma de n ENTRADAS DE DADOS para uma única SAÍDA.

MINTERMO	S	I <sub>B</sub>	I <sub>A</sub>	Y	
m <sub>0</sub>	0	0	0	0	$S' I_B' I_A$
m <sub>1</sub>	0	0	1	1	
m <sub>2</sub>	0	1	0	0	
m <sub>3</sub>	0	1	1	1	$S' I_B I_A$
m <sub>4</sub>	1	0	0	0	
m <sub>5</sub>	1	0	1	0	$S I_B I_A'$
m <sub>6</sub>	1	1	0	1	
m <sub>7</sub>	1	1	1	1	

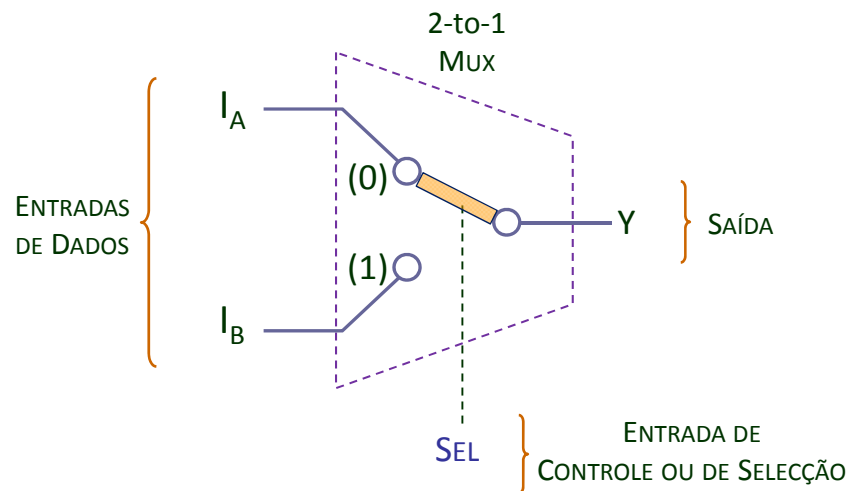
Tabela de Verdade da função Y.

$$Y = m_1 + m_3 + m_6 + m_7$$

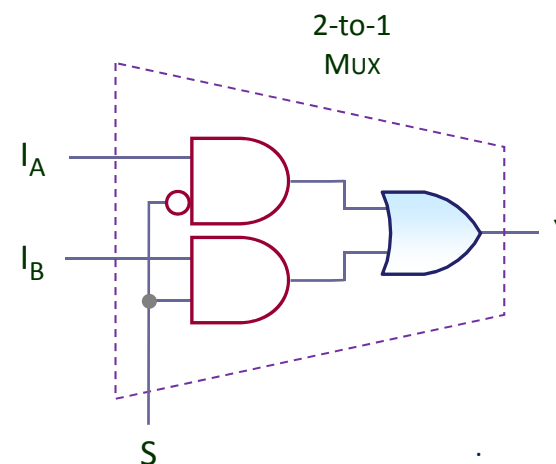
$$Y = S' I_B' I_A + S' I_B I_A + S I_B I_A' + S I_B I_A$$

$$Y = S' I_A + S I_B$$

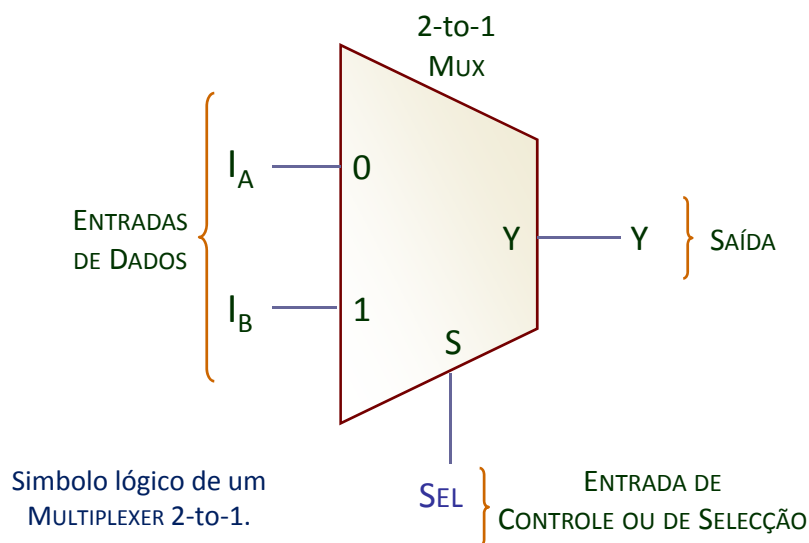
Expressão algébrica em AND-OR e forma canónica abreviada.



Analogia mecânica de um MULTIPLEXER 2-to-1 (comutador digital).



Estrutura interna de um MULTIPLEXER 2-to-1.



Símbolo lógico de um MULTIPLEXER 2-to-1.

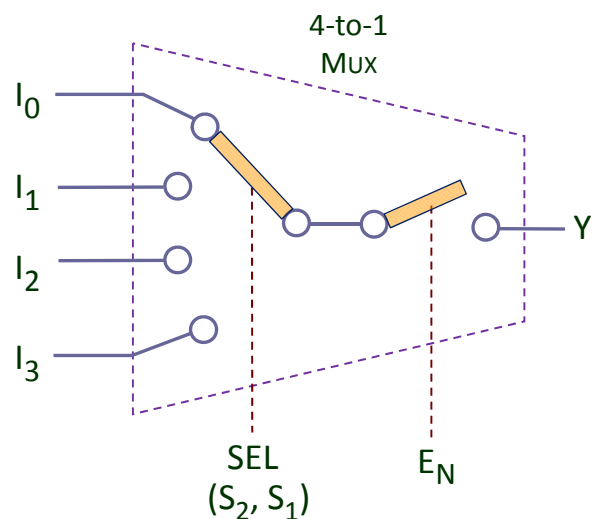
A função de multiplexagem consiste no encaminhamento de um sinal de uma de  $n$  entradas para uma única saída  $Y$ , em exclusão.

Para determinar qual das  $n$  entradas é canalizada para a saída existe uma palavra de selecção **SEL** constituída por um número de bits igual a  $\log_2 n$ , que codifica o número da entrada seleccionada.

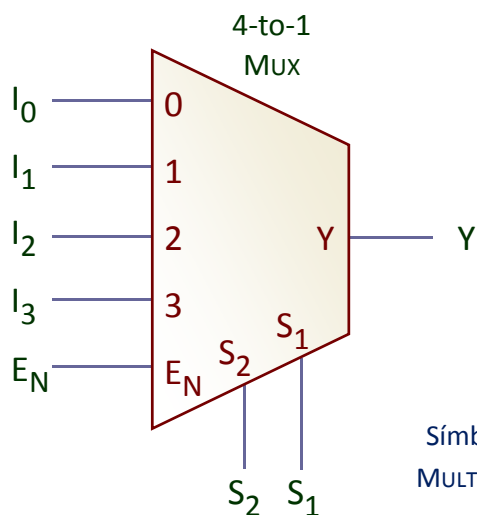
A implementação (em cima) baseia-se na intercepção individual de cada uma das entradas e, posteriormente na sua união numa única saída.

## MULTIPLEXER DE 4 ENTRADAS

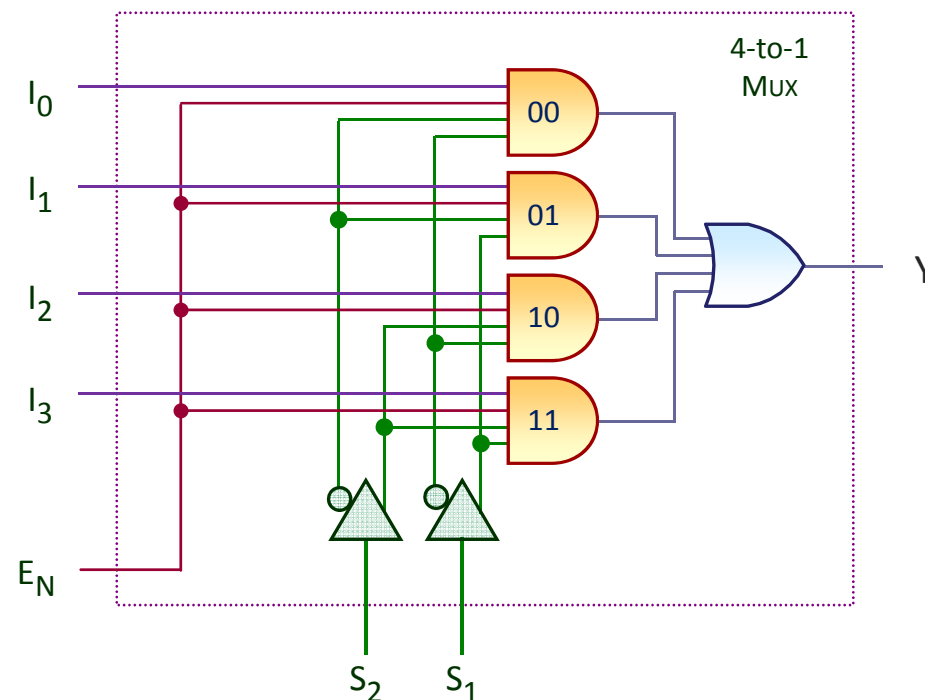
4-14



Analogia mecânica da função de selecção **SEL** (a 2 bits) de uma de 4 entradas  $I_i$  realizada por um **MULTIPLEXER** com **ENABLE** ( $E_N$ ).



Símbolo lógico de um  
MULTIPLEXER 4-to-1 com  
ENABLE.

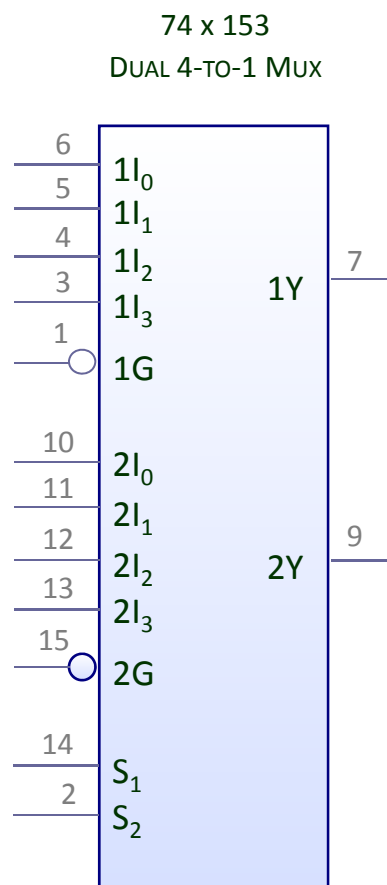


Estrutura interna de um MULTIPLEXER 4-to-1 com ENABLE.

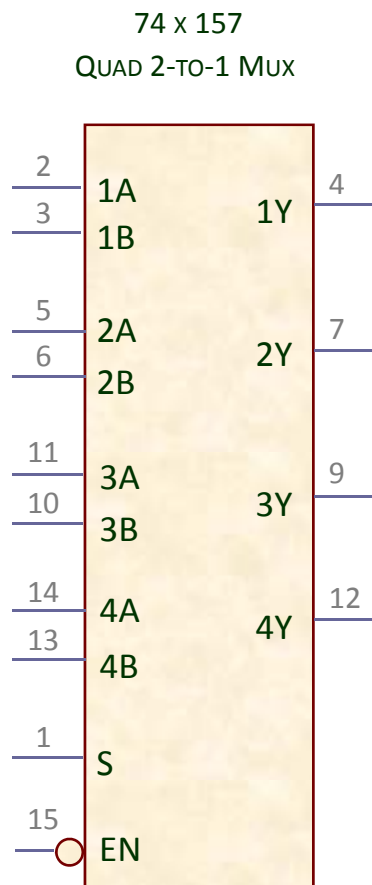
Um MULTIPLEXER gera internamente uma soma de produtos de todas as variáveis de selecção – uma **soma de mintermos**.

## MULTIPLEXERS 74x153 E 74x157

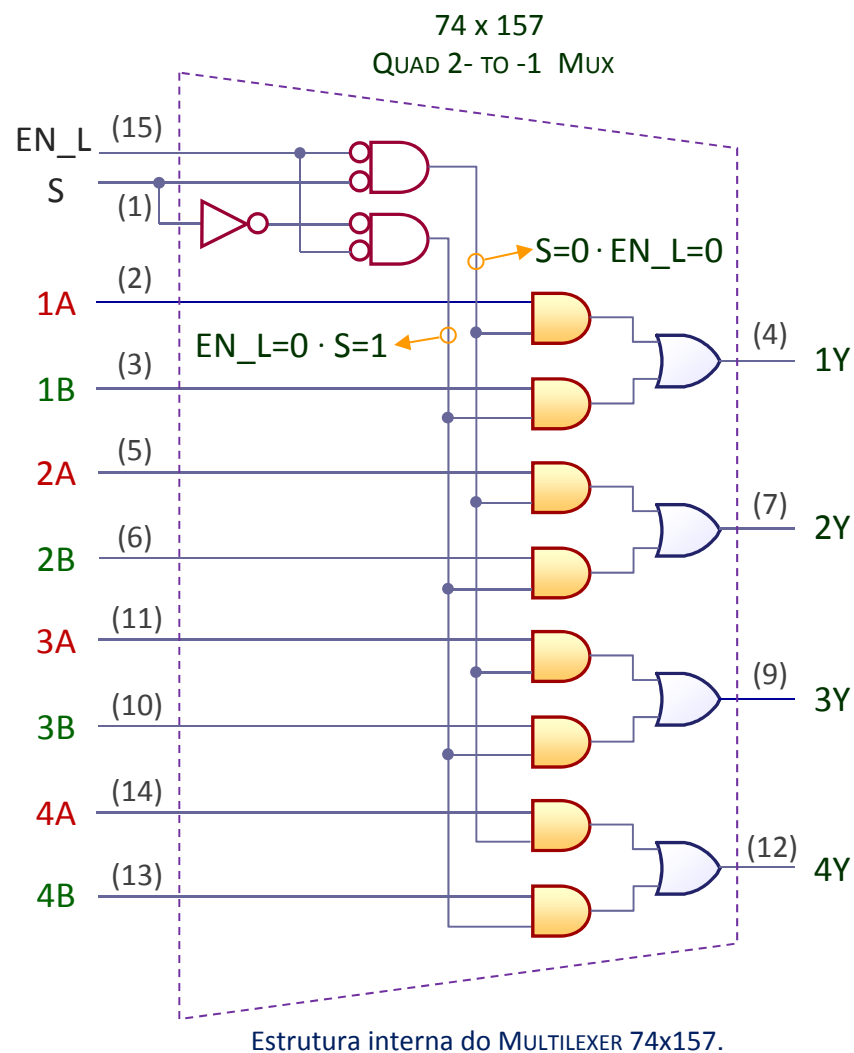
4-15



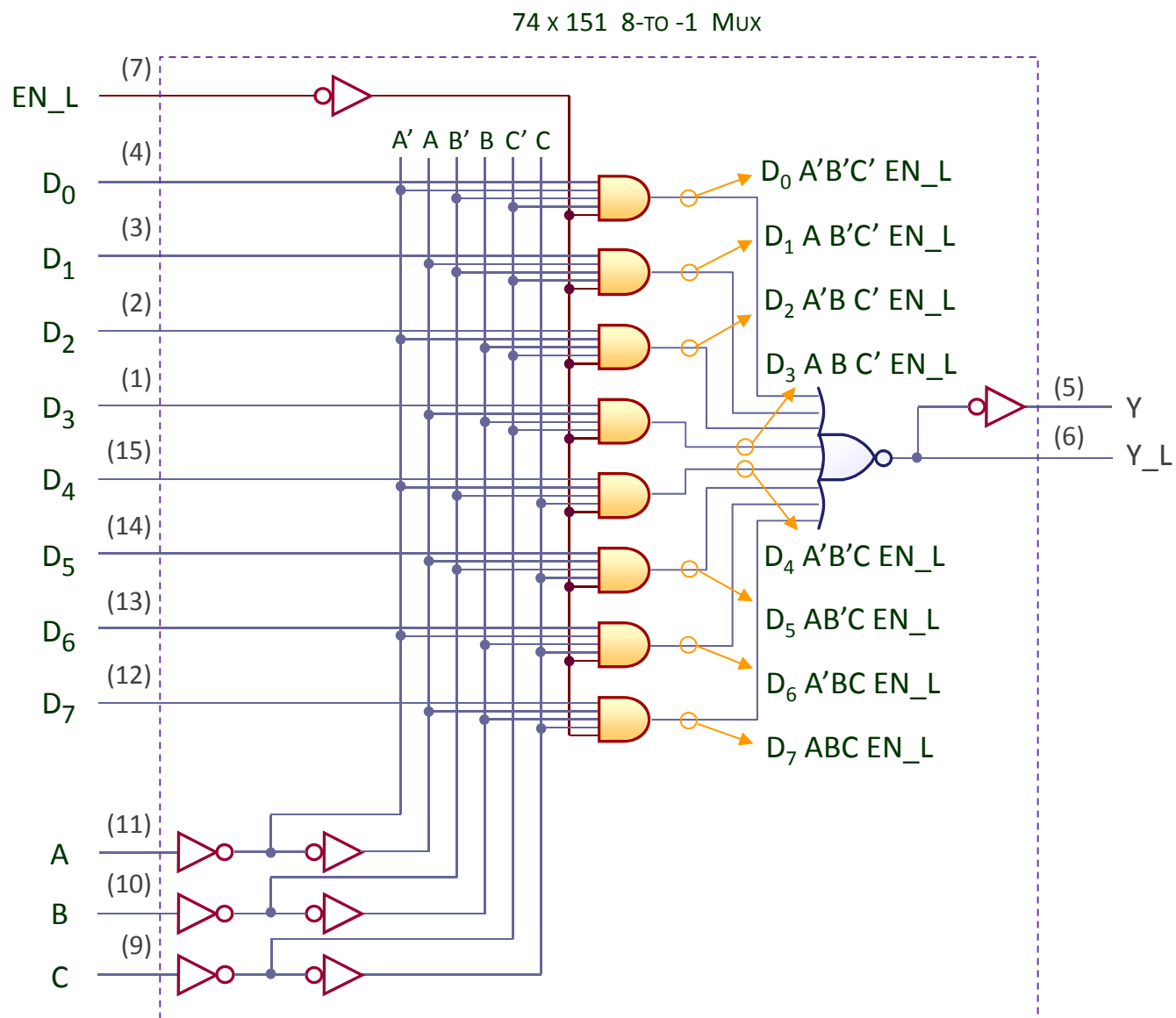
Símbolo lógico e configuração dos pinos do MULTIPLEXER 74x153.



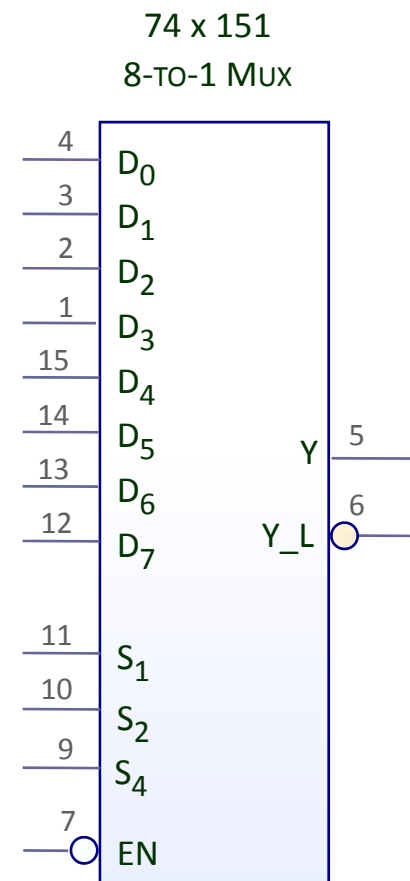
Símbolo lógico e configuração dos pinos do MULTIPLEXER 74x157.



O circuito integrado 74x153 é um MULTIPLEXER duplo de 4 entradas para 1 saída. O circuito integrado 74x157 é um MULTIPLEXER quádruplo de 2 entradas para 1 saída. Em ambos a entrada de ENABLE é do tipo ACTIVE-LOW, normalmente designada por G ou EN. No 74x157 ela é comum aos módulos constituintes. As entradas de selecção dos MULTIPLEXERS são designadas por  $S_1$ ,  $S_2$ , ou alternativamente por  $S_0$ ,  $S_1$ , ou ainda por A, B.

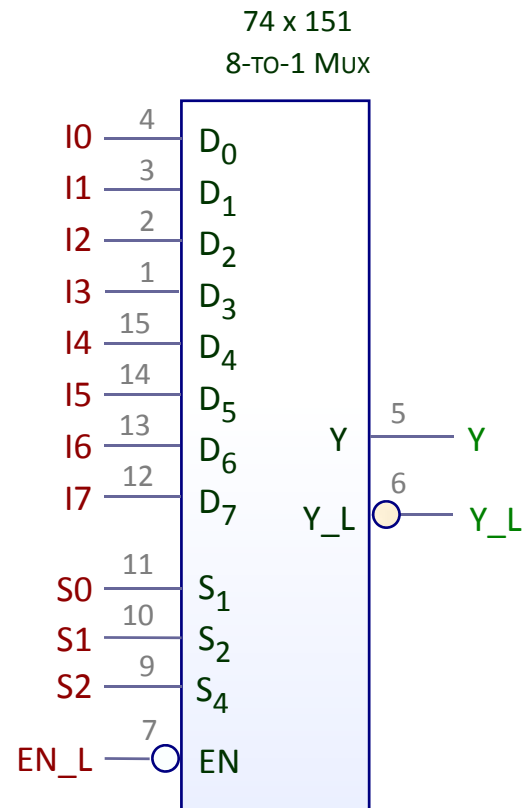


Estrutura interna do MULTIPLEXER 74x151.

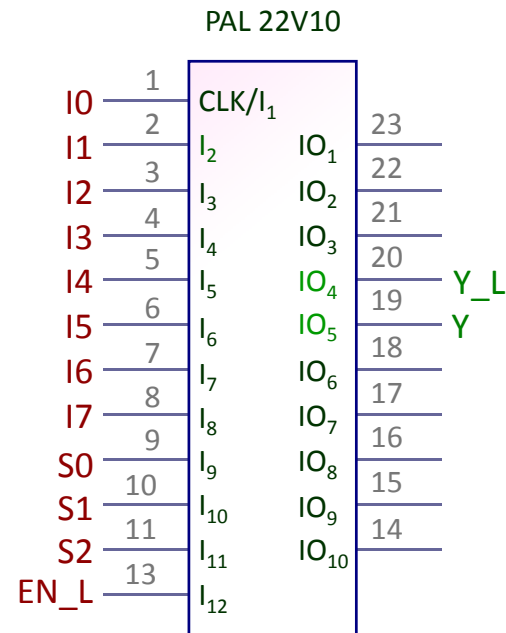


Símbolo lógico e configuração dos pinos do MULTIPLEXER 74x151.





Símbolo lógico e configuração dos pinos do MULTIPLEXER do tipo 74x151.



Símbolo lógico da PAL assinalando os pinos de entrada e saída utilizados para a implementação do MULTIPLEXER 8x1 ao lado.

```

/***** Input PINS *****/
PIN [1..8] = [I0..7] ;
PIN [9..11] = [S0..2] ;
PIN 13      = EN_L ;

/***** Output PINS *****/
PIN 19 = Y ;
PIN 20 = Y_L ;

/***** Body *****/
FIELD sel = [S0..2];

Y =      I0 & !EN_L & sel:0 #
        I1 & !EN_L & sel:1 #
        I2 & !EN_L & sel:2 #
        I3 & !EN_L & sel:3 #
        I4 & !EN_L & sel:4 #
        I5 & !EN_L & sel:5 #
        I6 & !EN_L & sel:6 #
        I7 & !EN_L & sel:7 ;

Y_L = !Y ;
    
```

Troço de código CUPL para um MULTIPLEXER 8x1.

## IMPLEMENTAÇÃO DE FUNÇÕES COM MULTIPLEXERS (EX. 4-1-1)

4-18

### Exemplo 4-1

Um MULTIPLEXER é construído internamente como uma soma de produtos de todas as variáveis de selecção - uma soma de mintermos.

Esta característica pode ser utilizada para a implementação de funções.

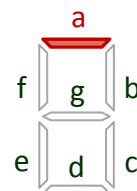
Com um MULTIPLEXER de  $n$  variáveis de selecção pode-se construir qualquer função de  $n$  variáveis.

Mostra-se como exemplo a implementação da saída correspondente ao segmento 'a' de um decodificador BCD-7-segmentos com um MULTIPLEXER de 16 entradas.

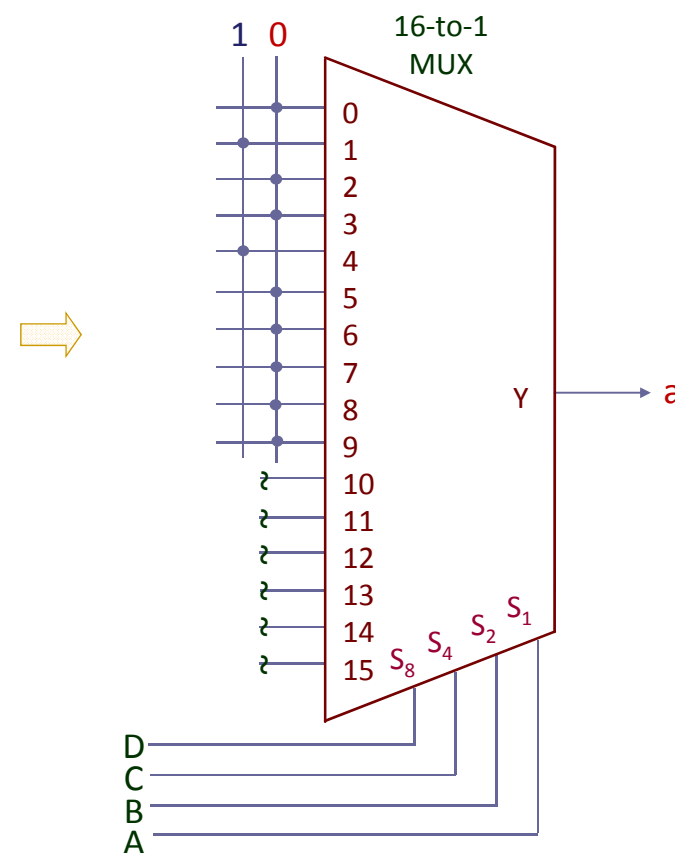
Ligam-se as variáveis da função às variáveis de selecção do MULTIPLEXER pela mesma ordem de pesos. O MULTIPLEXER apresentará nas saídas 0 ou 1 consoante o valor presente em cada uma das suas entradas que deverá coincidir com o valor da função 'a' para cada mintermo da tabela.

MINTERMO	D	C	B	A	a
$m_0$	0	0	0	0	0
$m_1$	0	0	0	1	1
$m_2$	0	0	1	0	0
$m_3$	0	0	1	1	0
$m_4$	0	1	0	0	1
$m_5$	0	1	0	1	0
$m_6$	0	1	1	0	0
$m_7$	0	1	1	1	0
$m_8$	1	0	0	0	0
$m_9$	1	0	0	1	0

Tabela de verdade para o segmento 'a' de um decodificador BCD-7-segmentos ACTIVE LOW.



Segmento 'a' iluminado.

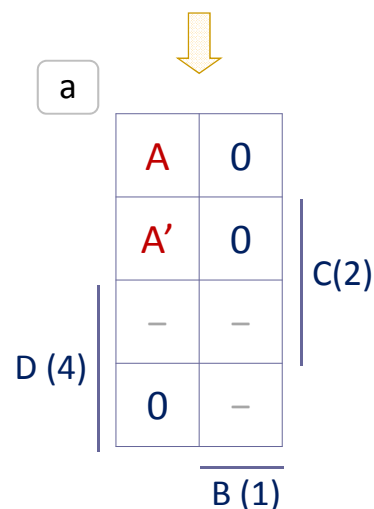
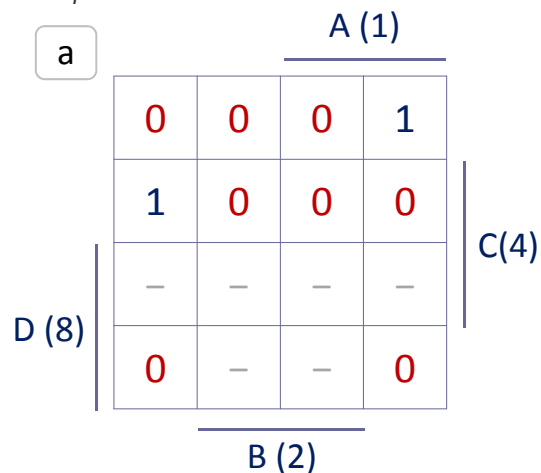


Implementação do segmento 'a' com um MULTIPLEXER de 16 entradas.

# IMPLEMENTAÇÃO DE FUNÇÕES COM MULTIPLEXERS (EX. 4-1-2)

4-19

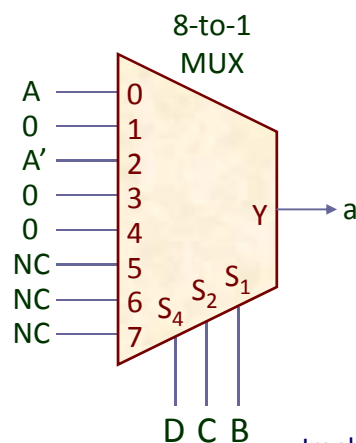
Exemplo 4-1



Mapas de Karnaugh a 3 e 4 variáveis para o segmento 'a'.

Pode implementar-se a mesma função com um MULTIPLEXER de 8 entradas, em vez de 16.

Uma das variáveis de entrada não poderá ligar-se às entradas de selecção do MULTIPLEXER. Pode escolher-se qualquer uma como mostrado neste e nos slides seguintes. Pode ser necessário dispor também das variáveis complementadas, mas isto nem sempre acontece.



MINTERMO	D	C	B	A	a
m <sub>0</sub>	0	0	0	0	0
m <sub>1</sub>	0	0	0	1	1
m <sub>2</sub>	0	0	1	0	0
m <sub>3</sub>	0	0	1	1	0
m <sub>4</sub>	0	1	0	0	1
m <sub>5</sub>	0	1	0	1	0
m <sub>6</sub>	0	1	1	0	0
m <sub>7</sub>	0	1	1	1	0
m <sub>8</sub>	1	0	0	0	0
m <sub>9</sub>	1	0	0	1	0

Tabela de verdade para o segmento 'a' de um decodificador BCD-7-segmentos ACTIVE LOW.

Neste caso foi arbitrariamente escolhida a variável A para não figurar nas entradas de selecção S<sub>1</sub> a S<sub>4</sub> do MULTIPLEXER, o que significa que as suas entradas de dados poderão tomar agora não só os valores 0 e 1 mas também A ou A'.

Implementação do segmento 'a' com um MULTIPLEXER de 8 entradas após a inserção da variável A no mapa de Karnaugh.



## IMPLEMENTAÇÃO DE FUNÇÕES COM MULTIPLEXERS (EX. 4-1-3)

4-20

### Exemplo 4-1

Generalizando: com um MULTIPLEXER  $2^n$ -to-1 é possível implementar qualquer função de  $n$  variáveis directamente a partir da tabela de verdade da função. As variáveis da função são as variáveis de selecção do MULTIPLEXER, e as suas entradas serão sempre 0 ou 1.

É também possível implementar a mesma função com um Multiplexer  $2^{n-1}$ -to-1, com  $n-1$  variáveis de selecção e metade do número de entradas. A variável  $V$  da função que não entra nas variáveis de selecção poderá ter de figurar nas entradas do Multiplexer.

Cada entrada do Multiplexer poderá assumir neste caso os valores 0, 1,  $V$  ou  $V'$ .

a

	A (1)			
	C	0	0	C'
D(4)	0	-	-	0
	B (2)			

a

	A (1)			
	0	0	0	D'
D(4)	1	0	0	0
	B (2)			

a

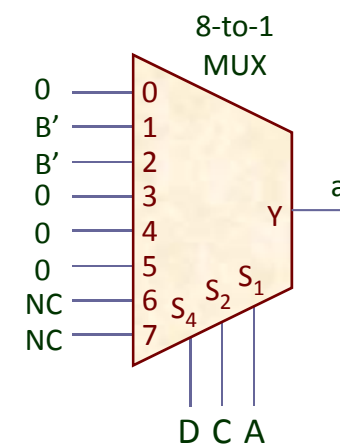
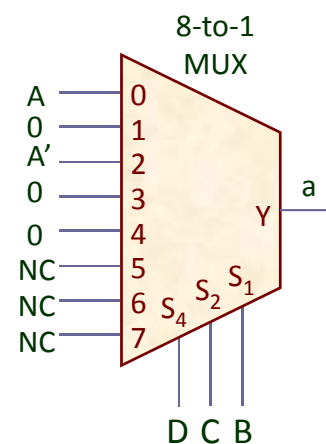
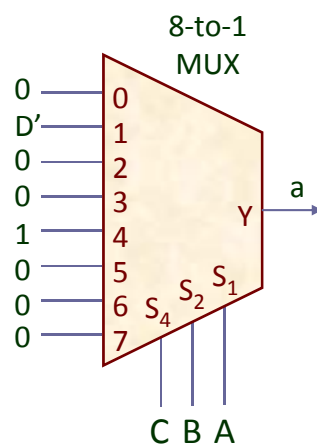
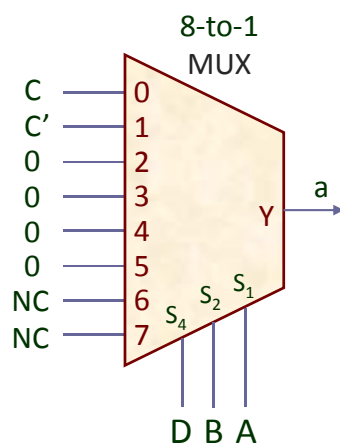
	A	0
	A'	0
	-	-
D(4)	0	-
	B (1)	

C(2)

a

	A (1)	
	0	B'
	B'	0
	-	-
D(4)	0	0
	B (1)	

C(2)



Quatro implementações possíveis para o segmento 'a' com MULTIPLEXER de dimensão inferior com metade (8) do número de entradas.

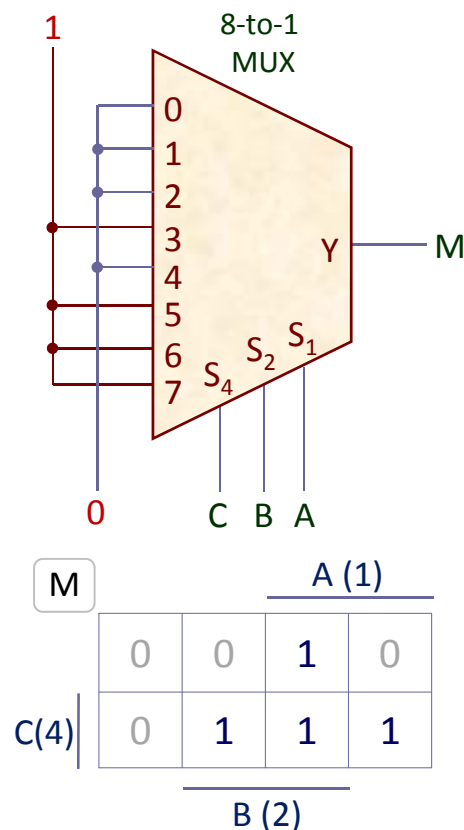


# IMPLEMENTAÇÃO DE FUNÇÕES COM MULTIPLEXERS (EX. 4-2-1)

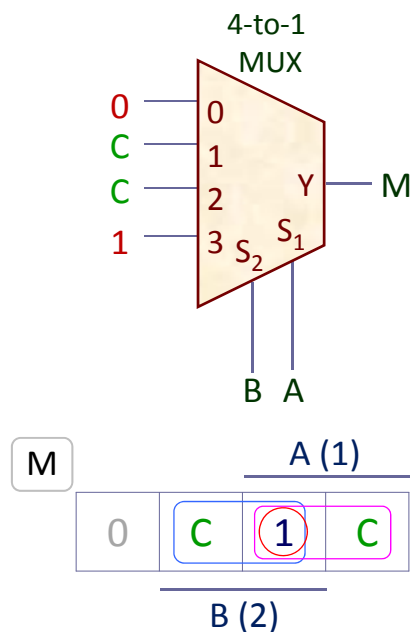
4-21

Exemplo 4-2

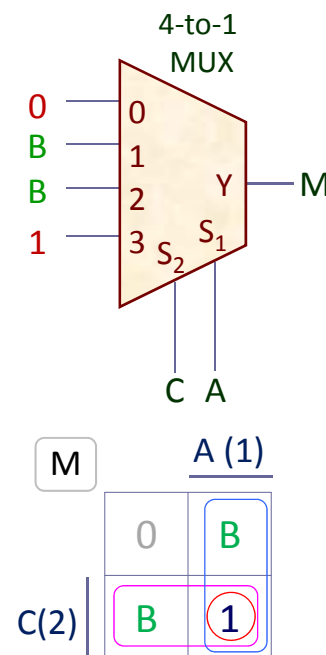
Este outro exemplo mostra a implementação da função Maioria a 3 variáveis: primeiro com um MULTIPLEXER de 8 entradas (3 variáveis de selecção), depois com um MULTIPLEXER de 4 entradas (2 variáveis de selecção) em 3 variantes.



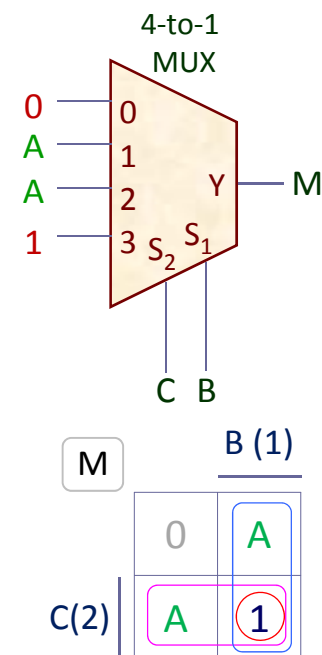
$$M = \sum (3, 5, 6, 7) = AB + BC + AC$$



$$M = AB + BC + AC$$



$$M = AC + BC + AB$$



$$M = BC + AC + AB$$

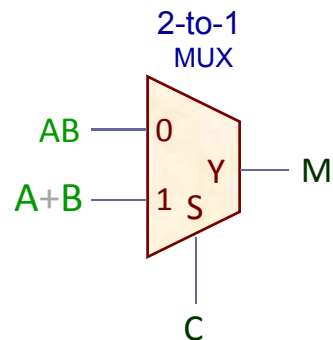
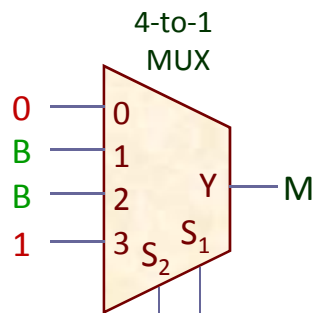
Mapas de Karnaugh com variáveis inseridas e equações lógicas para implementação da função Maioria a 3 variáveis com MULTIPLEXERS 8-to-1 e 4-to-1.



## IMPLEMENTAÇÃO DE FUNÇÕES COM MULTIPLEXERS (EX. 4-2-2)

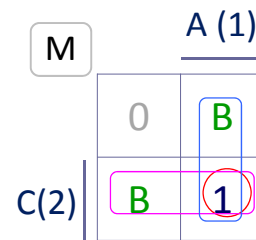
4-22

Exemplo 4-2

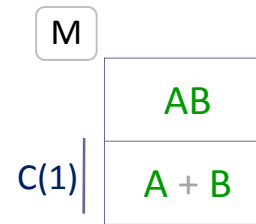
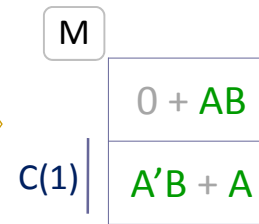


$$M = ABC' + (A+B)C = ABC' + AC + BC = A(BC' + C) + BC = AB + AC + BC$$

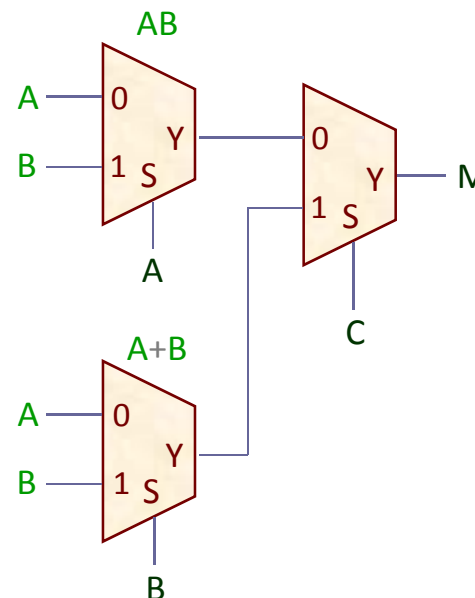
Mapas de Karnaugh com variáveis inseridas e equações lógicas para implementação da função Maioria a 3 variáveis com MULTIPLEXERS 4-to-1 e 2-to-1.



$$M = AB + BC + AC$$



$$M = ABC' + BC + AC = AB + BC + AC$$



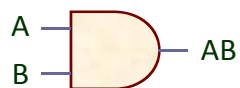
Os exemplos anteriores requereram a inserção de variáveis no mapa de Karnaugh.

A simplificação duma função a partir de um mapa de Karnaugh com variáveis inseridas ocorre da seguinte forma:

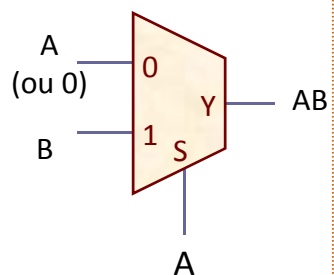
- Extraem-se os uns (1s) presentes no mapa, considerando como 0 as variáveis inseridas e aproveitando as eventuais indiferenças.
- Extraem-se as variáveis inseridas, considerando os 1s também como indiferenças.



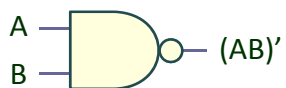
## AND



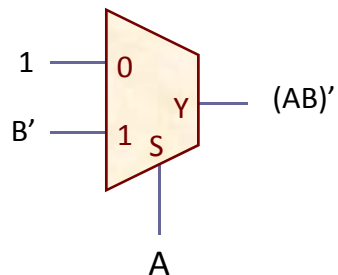
B	A	AB
0	0	0
0	1	0
1	0	0
1	1	1



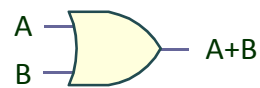
## NAND



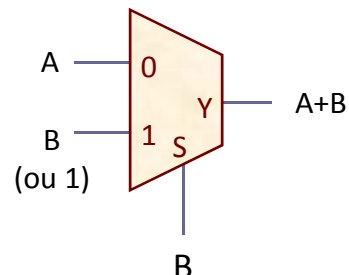
B	A	(AB)'
0	0	1
0	1	1
1	0	1
1	1	0



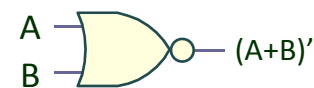
## OR



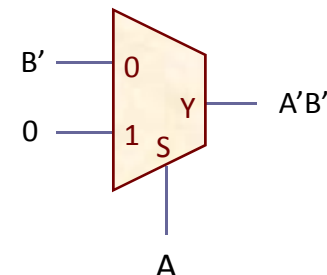
B	A	A+B
0	0	0
0	1	1
1	0	1
1	1	1



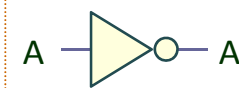
## NOR



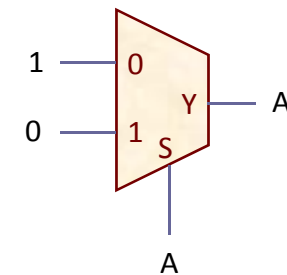
B	A	(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0



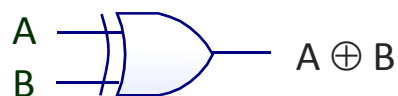
## NOT



A	A'
0	1
1	0

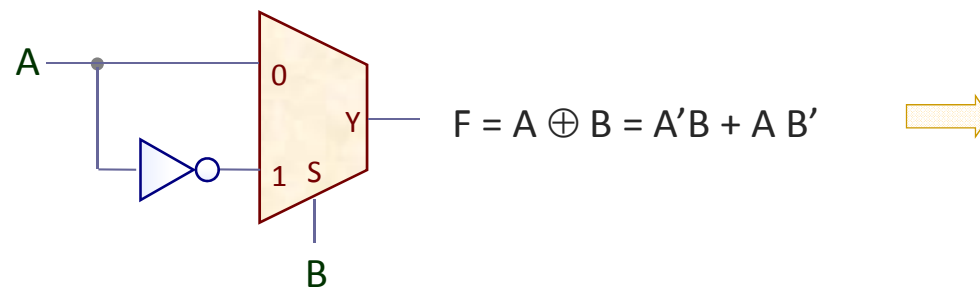


Símbolos lógicos, tabelas de verdade e realização de portas lógicas através de MULTIPLEXERS 2-to-1.

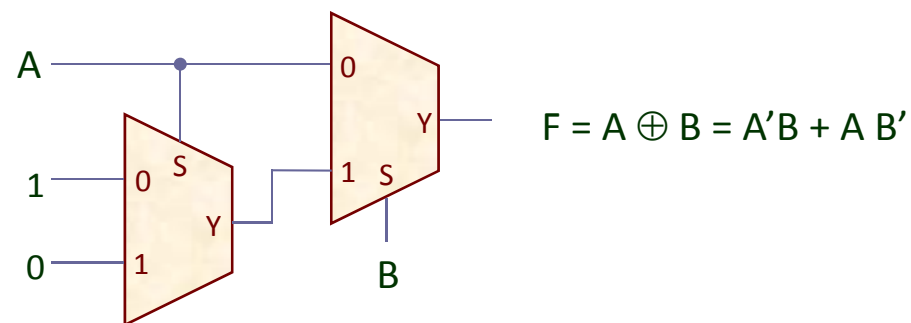


B	A	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Símbolo lógico e tabela de verdade do XOR.



Realização da porta XOR através de um inversor e um MULTIPLEXER 2-to-1.



Realização da porta XOR através de dois MULTIPLEXERS 2-to-1.



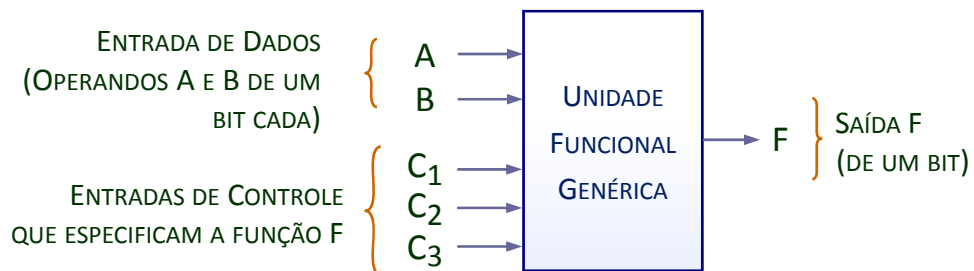
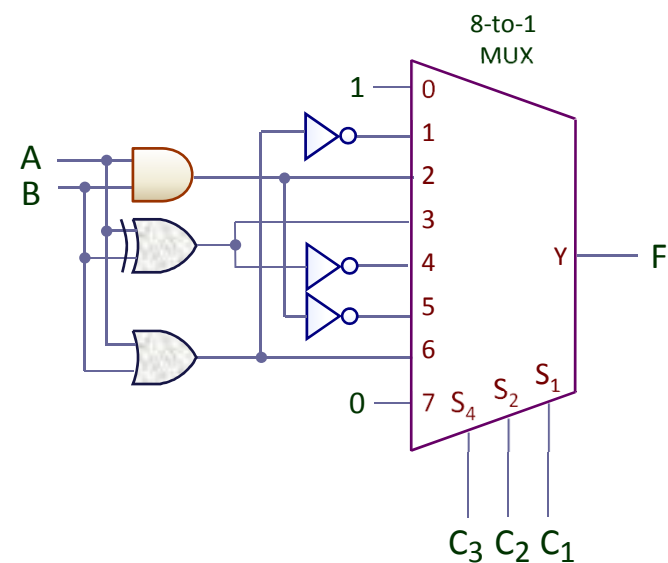


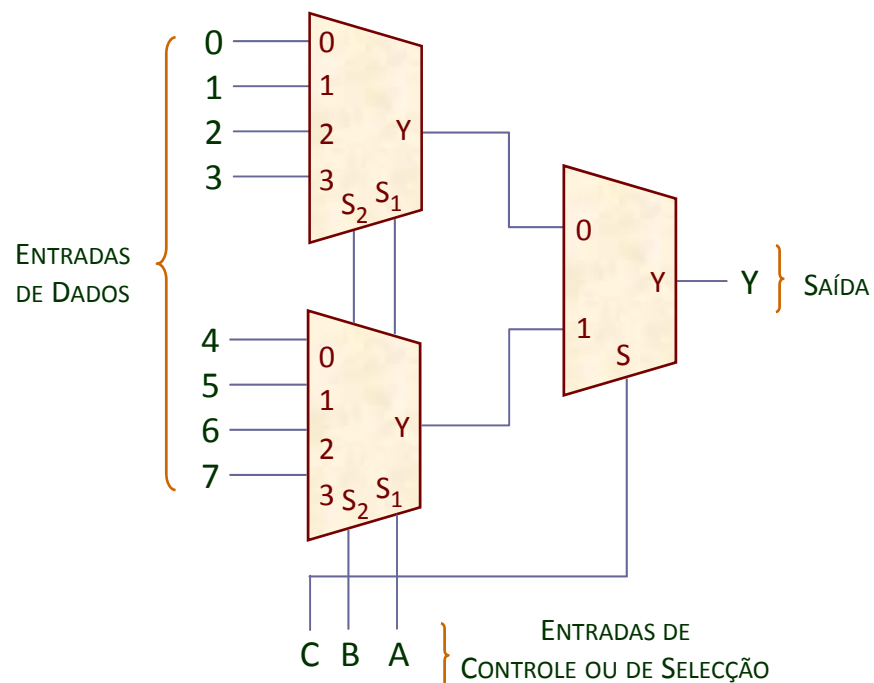
Diagrama de uma unidade funcional genérica com entradas de controle **C** a 3 bits e entradas de dados (operandos) **A** e **B** a um bit. As entradas de controle especificam a função **F** de 1 bit (mesma largura de bits que os operandos).

C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	FUNÇÃO F	
0	0	0	1	SEMPRE 1
0	0	1	$(A + B)'$	NOR
0	1	0	$A \cdot B$	AND
0	1	1	$A \text{ XOR } B$	XOR
1	0	0	$A \text{ XNOR } B$	XNOR
1	0	1	$(A \cdot B)'$	NAND
1	1	0	$A + B$	OR
1	1	1	0	SEMPRE 0

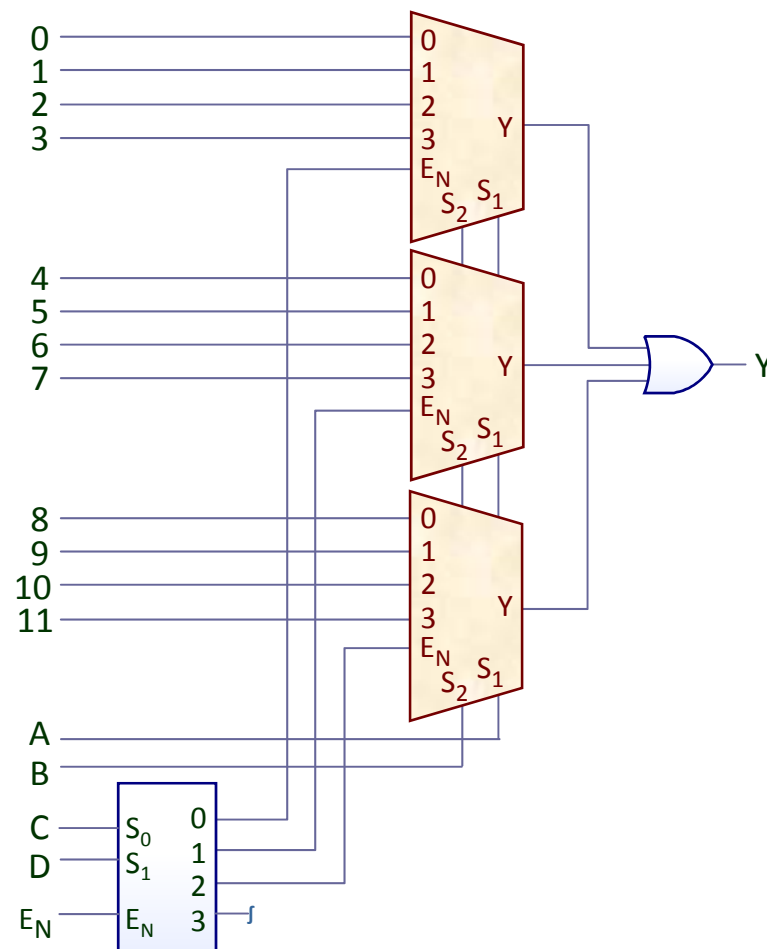
Tabela funcional de F.



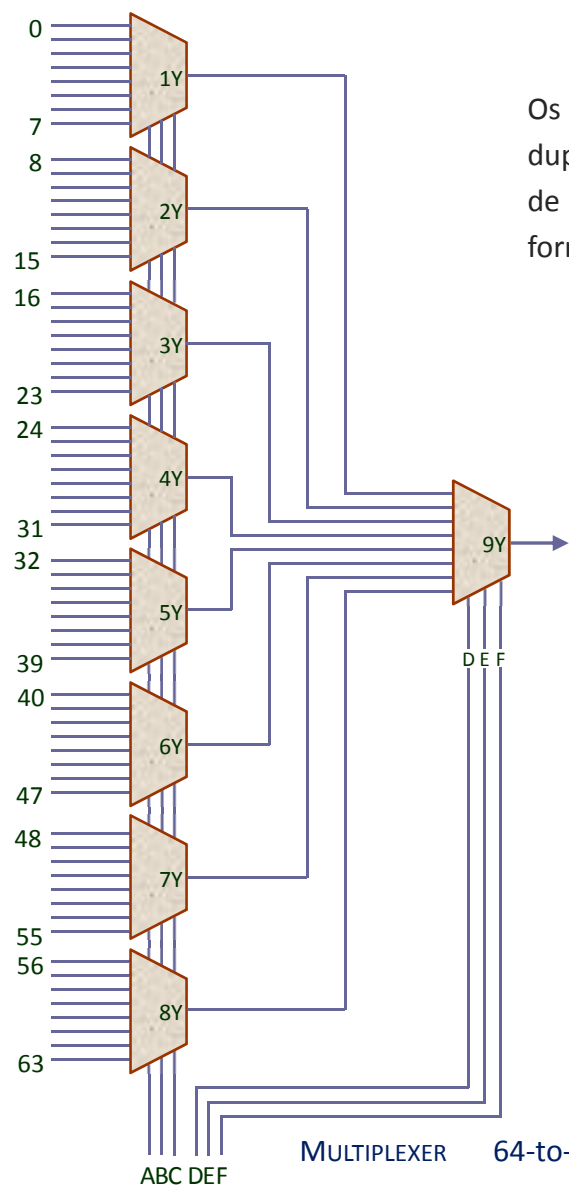
Implementação de F com um MULTIPLEXER de 8 entradas e portas lógicas directamente a partir da tabela funcional.



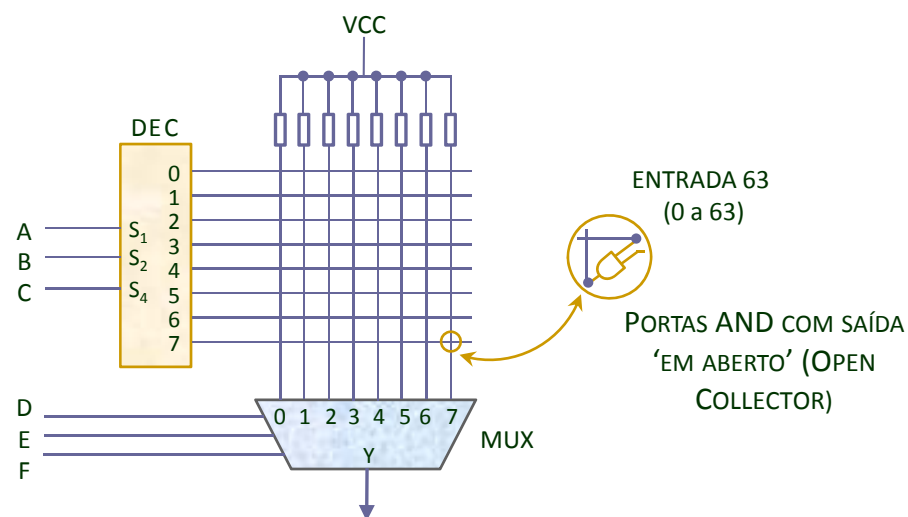
MULTIPLEXER 8-to-1 sintetizado a partir de 2 MULTIPLEXERS 4-to-1 e ainda de um MULTIPLEXER 2-to-1 sem tirar partido das entradas de ENABLE.



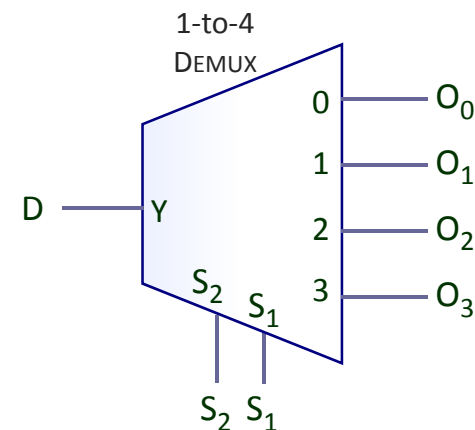
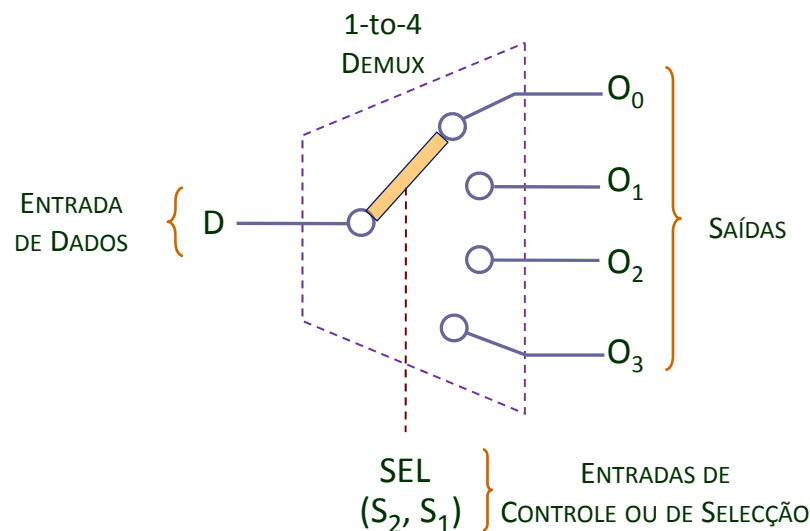
MULTIPLEXER 12-to-1 sintetizado a partir de 4 MULTIPLEXERS 4-to-1 e de um DECODER 2-to-4 tirando partido das entradas de ENABLE.



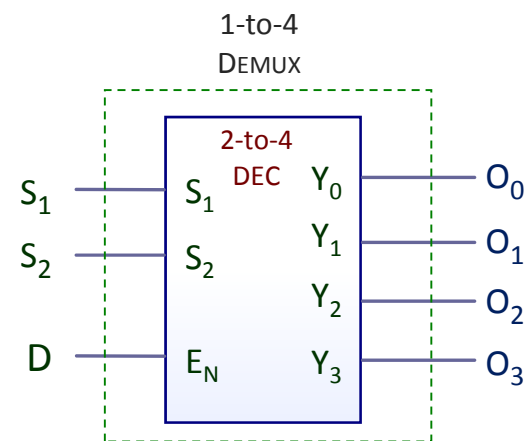
Os Multiplexers podem organizar-se arborescentemente em termos de irem sucessivamente duplicando o número de entradas. Mostram-se 2 Multiplexers de 64 entradas, um organizado de forma unidimensional só a partir de outros Multiplexers de menor capacidade, outro de forma bidimensional .



MULTIPLEXER 64-to-1 sintetizado de forma bidimensional a partir de um MULTIPLEXER 8-to-1, de um DECODER 3-to-8 e de 64 portas AND.



Símbolo lógico de um DEMULTIPLEXER 1-to-4.

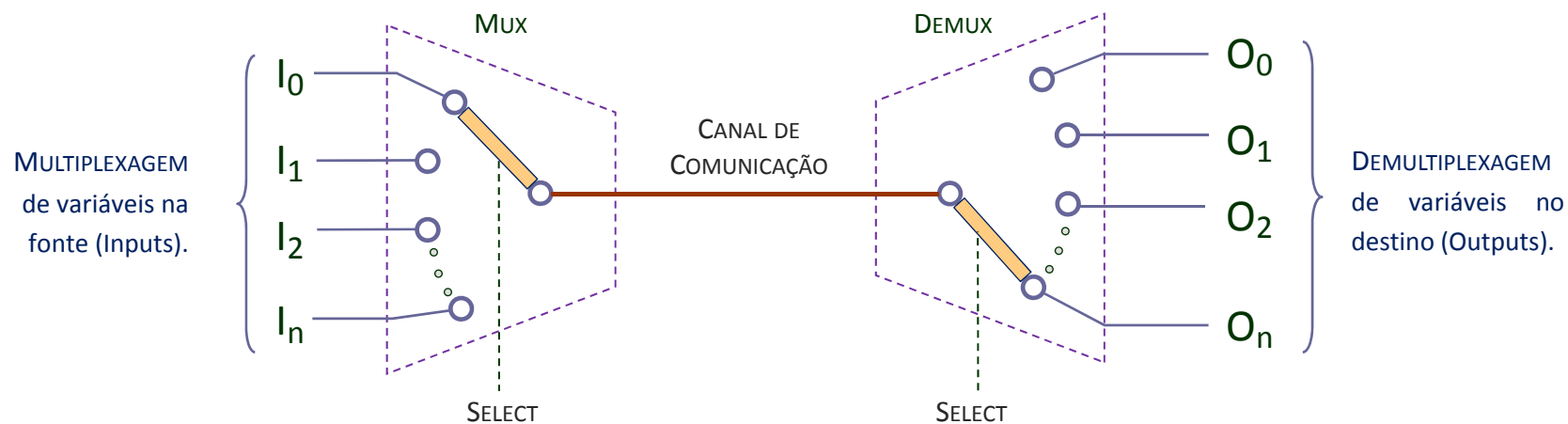


DEMULTIPLEXER de 1 entrada  $D$  para 4 saídas  $O_i$  entendido como um DECODER com ENABLE.

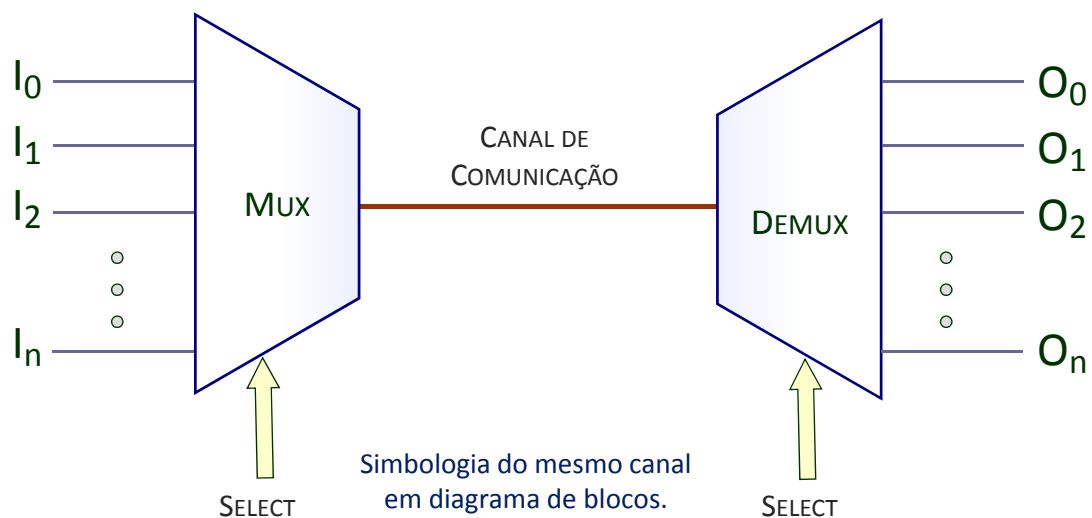
Analogia mecânica da função de selecção SEL (a 2 bits) de um DEMULTIPLEXER 1-TO-4,

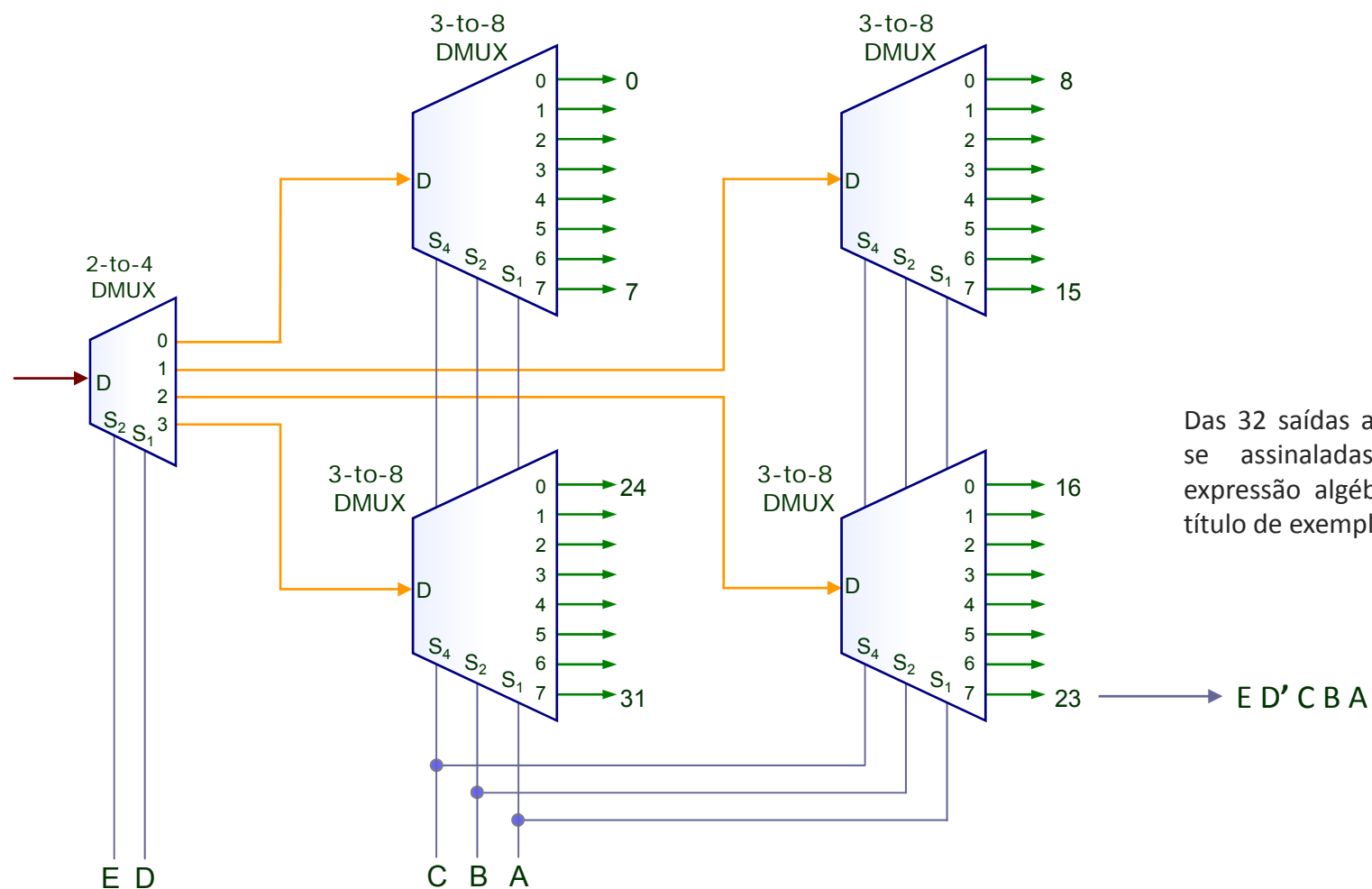
A função de DESMULTIPLEXAGEM consiste no encaminhamento de um sinal de uma única entrada para uma de  $n$  saídas. A entrada a encaminhar é especificada através de  $\log_2 n$  entradas selectoras. A funcionalidade de um DEMULTIPLEXER é oposta à de um MULTIPLEXER.

A estrutura de um DEMULTIPLEXER é idêntica à de um DECODER com ENABLE: a entrada de ENABLE do DESCODIFICADOR funcionará como a entrada de dados de DEMULTIPLEXER.

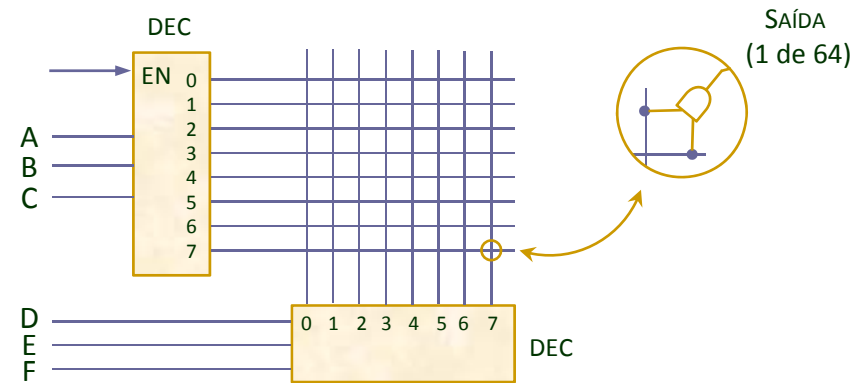
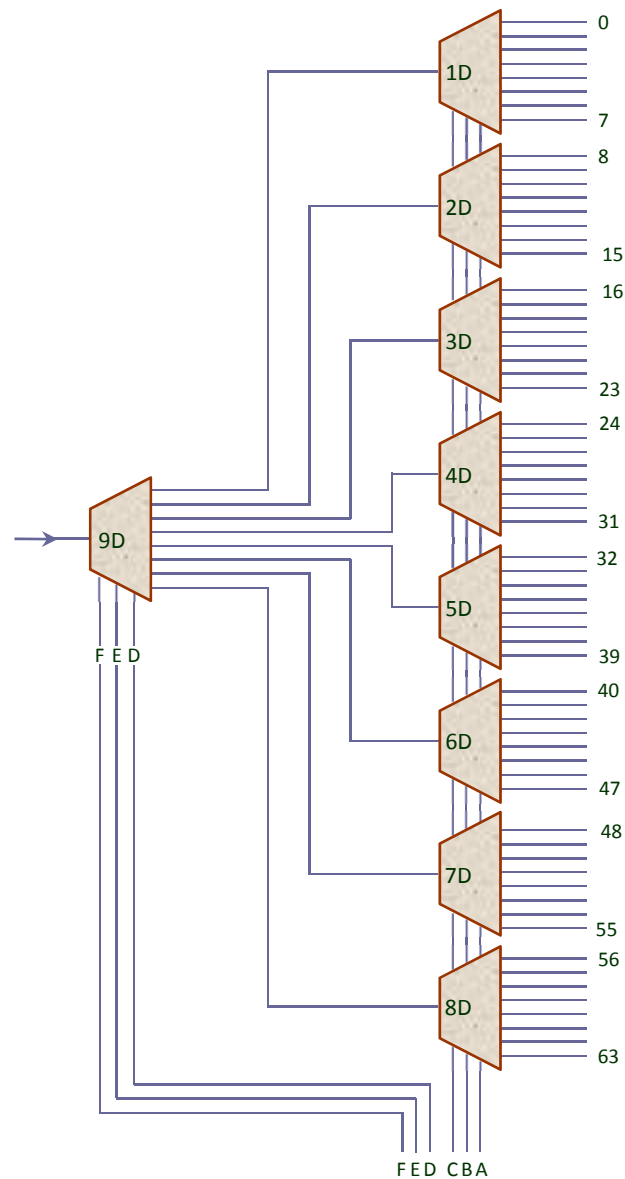


Analogia mecânica da função de MULTIPLEXAGEM na fonte e DEMULTIPLEXAGEM no destino de um canal de comunicação (Bus).



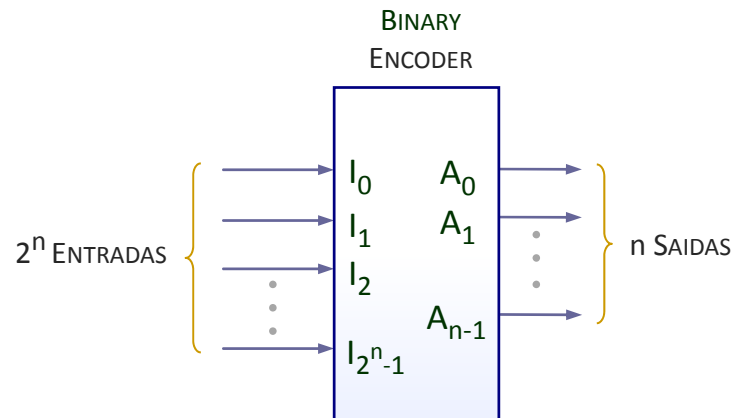


DEMÚLTIPLEXER 1-to-32 sintetizado a partir de quatro DEMÚLTIPLEXERS 3-to-8 e um DEMÚLTIPLEXER 2-to-4 .

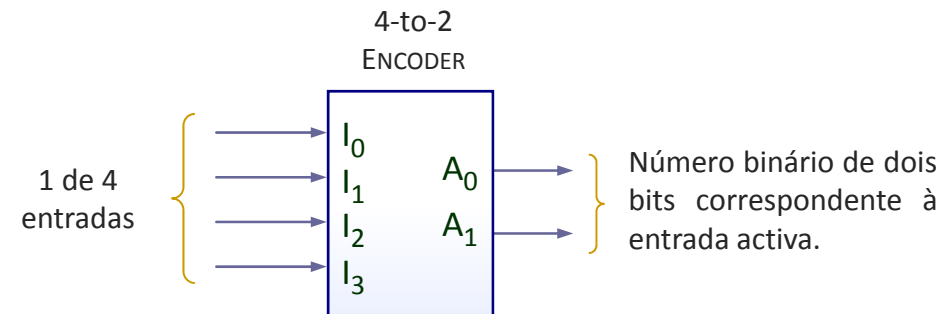


DEMULTIPLEXER 1-to-64 sintetizado de forma bidimensional a partir de dois DECODERS 3-to-8 e de 64 portas AND.

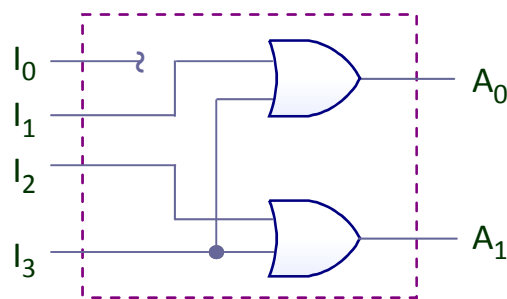
DEMULTIPLEXER 1-to-64 sintetizado de forma unidimensional por concatenação de 9 DEMULTIPLEXERS.



Estrutura genérica de um ENCODER.



Símbolo lógico de um 4-to-2 ENCODER.



Estrutura interna de um 4-to-2 ENCODER simples.

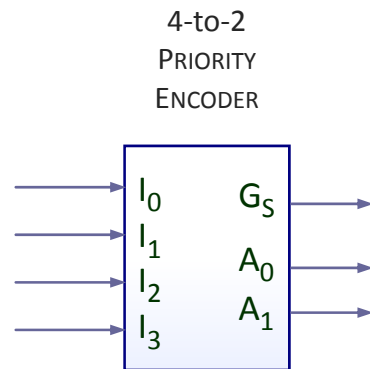
Um CODIFICADOR (ENCODER) é um circuito combinatório que, perante a activação de 1 de  $2^n$  entradas, gera nas  $n$  saídas a **palavra de código** correspondente a essa entrada. Um ENCODER tem uma função oposta à de um DECODER.

Os CODIFICADORES DE PRIORIDADE (PRIORITY ENCODERS) permitem a activação simultânea de mais de uma entrada gerando a codificação correspondente à entrada activa de maior prioridade.

O codificador mais simples é o codificador binário  $2^n$ -to- $n$ , e assume que só uma entrada está activa de cada vez. Realiza a função oposta à de um decodificador. Mas, se houver várias entradas activas em simultâneo, o circuito deixa de funcionar.

É preciso atribuir uma prioridade às linhas de entrada, devendo o codificador produzir o código correspondente à entrada de maior prioridade.

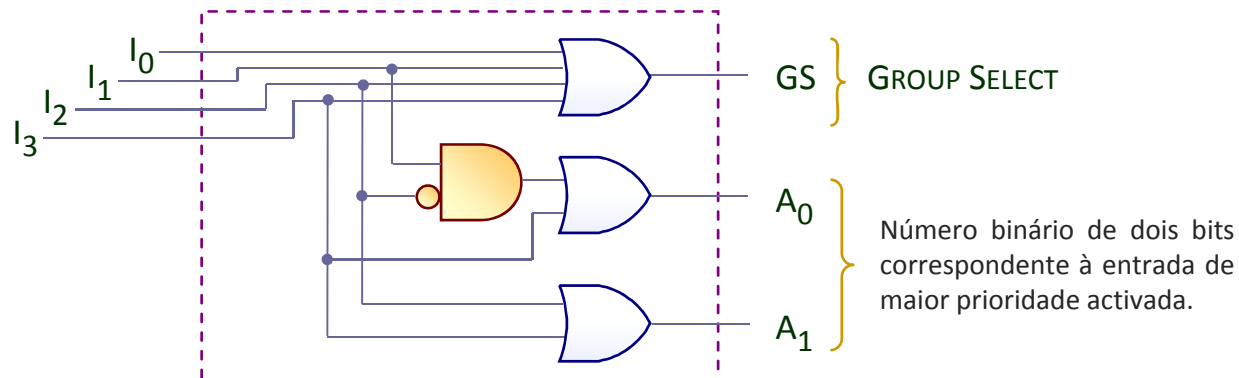




Símbolo lógico.

$I_3$	$I_2$	$I_1$	$I_0$	$A_1$	$A_0$	$G_S$
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	-	0	1	1
0	1	-	-	1	0	1
1	-	-	-	1	1	1

Tabela de verdade simplificada considerando a prioridade atribuída às entradas segundo a ordem crescente da sua significância numérica.



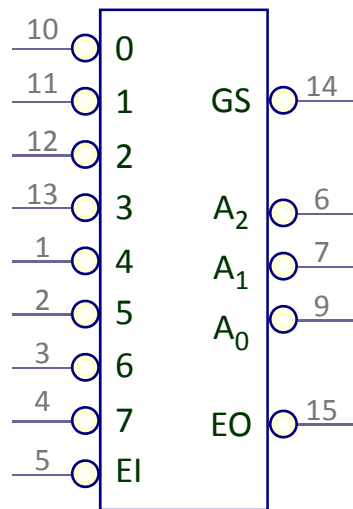
Estrutura interna de um 4-to-2 PRIORITY ENCODER.

$$G_S = I_0 + I_1 + I_2 + I_3$$

$$A_0 = I_1 I_2' I_3' + I_3 = I_1 I_2' + I_3$$

$$A_1 = I_2 I_3' + I_3 = I_2 + I_3$$

Expressões algébricas das saídas obtidas a partir da tabela de verdade.



Símbolo lógico e configuração dos pinos de um circuito 8-LINE to 3-LINE PRIORITY ENCODER 74x148.

Quando se efetuam múltiplos pedidos (várias entradas ativas em simultâneo), o PENC coloca na saída A<sub>2</sub>..A<sub>0</sub> o número do pedido (entrada) mais prioritário. I7 é a entrada que possui maior prioridade. IO é a entrada que possui menor prioridade.

A saída EO\_L (equivalente a IDLE) é ativada se não houver entradas ativadas, isto é se nenhuma das entradas estiver a 1:

$$(EO\_L)' = (EI\_L)' (7\_L \cdot 6\_L \cdot 5\_L \cdot 4\_L \cdot 3\_L \cdot 2\_L \cdot 1\_L \cdot 0\_L)$$

A saída EO\_L é destinada à concatenação. Deve ser ligada à entrada EI\_L de um outro PENC 74x148 que toma conta de um conjunto das 8 entradas de prioridade mais baixa. Só fica ativa se EI\_L estiver ativa mas não houver ativação de nenhuma das entradas li\_L para que um 74x148 de prioridade mais baixa possa ficar ENABLED.

A entrada EI\_L deve estar ativada (a zero) para que qualquer das saídas A<sub>0</sub>..A<sub>2</sub> possa também ser ativada. A tabela de verdade do 74x148 está no slide seguinte. Todas as entradas e saídas são do tipo ACTIVE-LOW.

A saída GS\_L fica ativa quando o PENC está ENABLED e possui pelo menos uma das entradas ativas.

**GS\_L** (GROUP SELECT ou 'GOT SOMETHING'):  
saída que **está** activa quando a entrada de controle EI\_L está activa e uma ou mais entradas de dados estão activas.

**EO\_L** (ENABLE OUTPUT):  
saída vocacionada para ser ligada à entrada EI\_L (ENABLE INPUT) de outro 74x148 que agrupe entradas de mais baixa prioridade.

A saída **EO\_L** está activa se se a entrada de controle EI\_L está activa mas nenhuma entrada de dados está activa. Qualquer saída só pode estar activa se a entrada EI\_L estiver activa.

EI_L	i_L	GS_L	EO_L	CONDIÇÃO
0	0	0	1	Existe pelo menos 1 entrada activa
0	1	1	0	Não existe nenhuma entrada activa
1	–	1	1	Circuito DISABLED

Tabela funcional simplificada do circuito 74x148.

ENTRADAS									SAÍDAS				
EI_L	0_L	1_L	2_L	3_L	4_L	5_L	6_L	7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	–	–	–	–	–	–	–	–	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	–	–	–	–	–	–	–	0	0	0	0	0	1
0	–	–	–	–	–	–	0	1	0	0	1	0	1
0	–	–	–	–	0	1	1	1	0	1	0	0	1
0	–	–	–	0	1	1	1	1	0	1	1	0	1
0	–	–	0	1	1	1	1	1	1	0	0	0	1
0	–	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Tabela de verdade do circuito PRIORITY ENCODER 74x148.

Reutilizando a expressão de EO\_L fica:

$$(GS\_L)' = (EI\_L)' \cdot EO\_L$$

Designam-se genericamente por  $I_i$  as 8 entradas, por  $A_i$  as 3 saídas (por simplificação todas ACTIVE-HIGH), e por  $H_i$  8 variáveis intermédias definidas de modo a que  $H_i = 1$  sse  $I_i$  for a entrada de maior prioridade activa:

$$H_7 = I_7$$

$$H_6 = I_6 \cdot I_7'$$

$$H_5 = I_5 \cdot I_6' \cdot I_7'$$

$$H_4 = I_4 \cdot I_5' \cdot I_6' \cdot I_7'$$

$$\dots$$

$$H_0 = I_0 \cdot I_1' \cdot I_2' \cdot I_3' \cdot I_4' \cdot I_5' \cdot I_6' \cdot I_7'$$

Vem então:

$$A_2 = (H_4 + H_5 + H_6 + H_7) \cdot EI$$

$$A_1 = (H_2 + H_3 + H_6 + H_7) \cdot EI$$

$$A_0 = (H_1 + H_3 + H_5 + H_7) \cdot EI$$

EI é o ENABLE INPUT, também ACTIVE-HIGH. Estas equações estão na base da implementação em CUPL e PAL do slide seguinte.

```

Name    PENC ;
Device  p22v10 ;

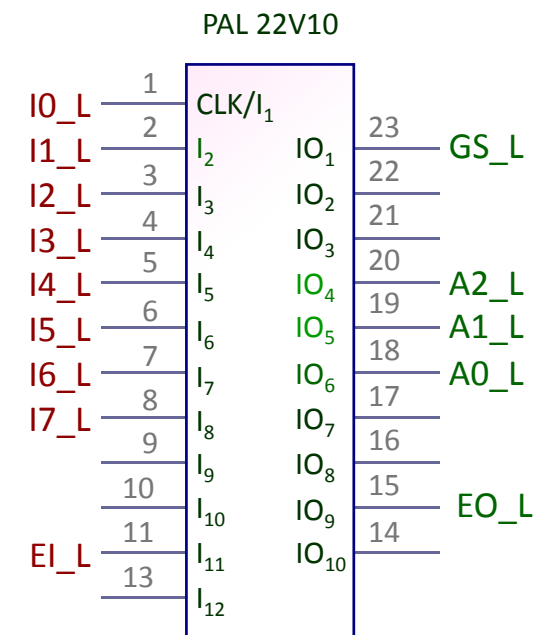
...
/***** Input Pins *****/
PIN [1..8]  = ![I0..7] ;
PIN 11      = !ENI ;

/***** Output PINS *****/
PIN 23      = !GS ;
PIN [18..20] = ![A0..2] ;
PIN 15      = !ENO ;

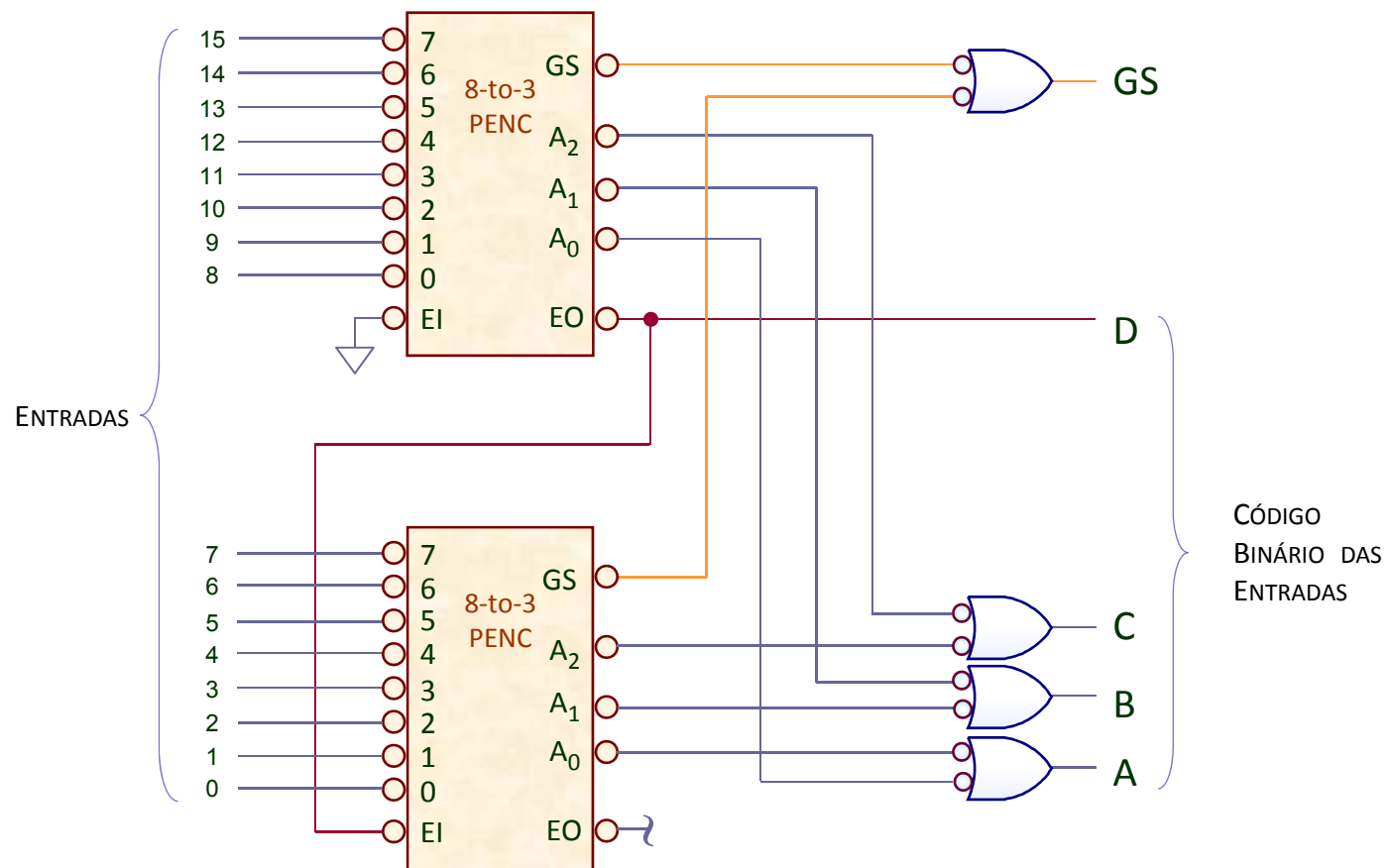
/***** Body *****/

h = !I0 & !I1 & !I2 & !I3 & !I4 & !I5 & !I6 & !I7;
ENO = h & ENI;
GS = ENI & !ENO;
A0 = (I7 # I5 & !I6 # I3 & !I4 & !I6 # I1 & !I2 & !I4 & !I6) & ENI;
A1 = (I7 # I6 # I3 & !I4 & !I5 # I2 & !I4 & !I5 )& ENI;
A2 = (I7 # I6 # I5 # I4)& ENI;
    
```

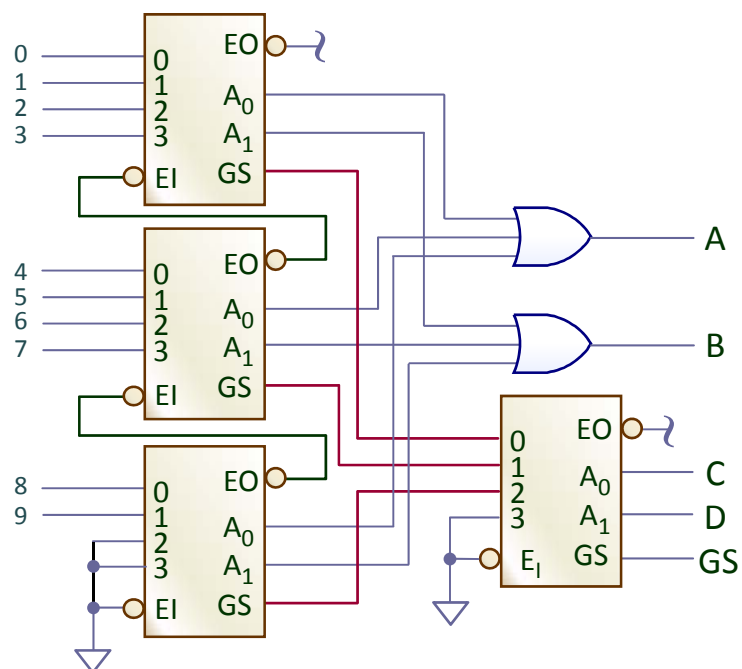
Troço de código CUPL para a implementação de um  
PRIORITY ENCODER 74x148 EM PAL 22V10.



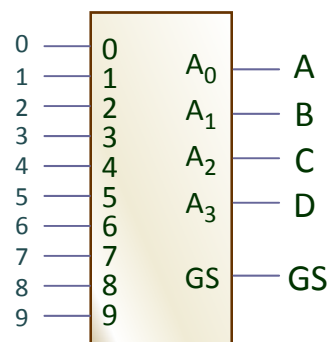
Símbolo lógico da PAL assinalando os pinos de entrada e saída utilizados para a implementação do PRIORITY ENCODER 74x148.



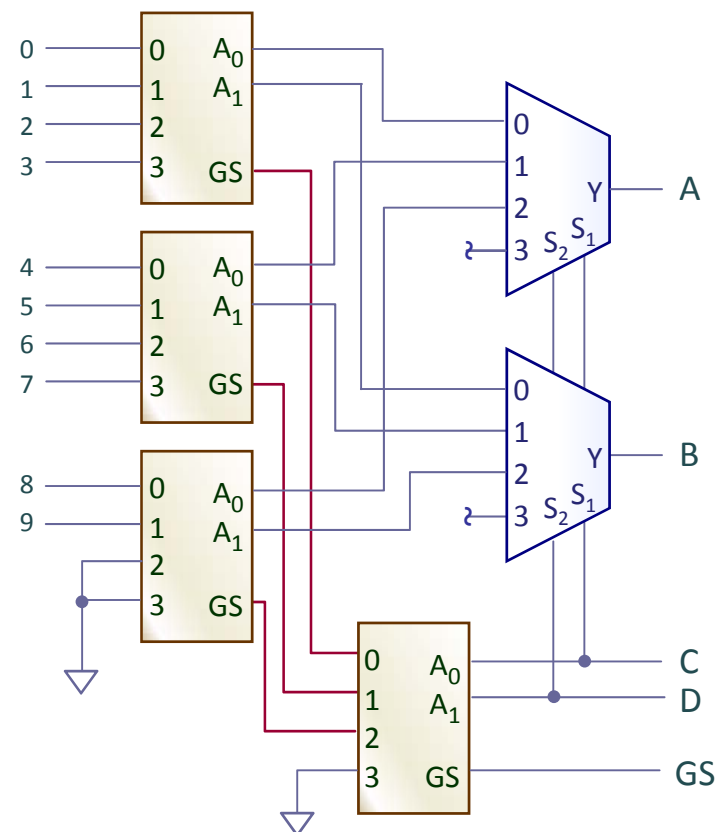
Concatenação de dois circuitos codificadores de 8 entradas do tipo 74LS148 para formação de um codificador de 16 entradas (16-to-4 PENC).



PENC de 10 entradas sintetizado a partir de PENC 4x2 usando a funcionalidade ENABLE através dos pinos EI e EO.



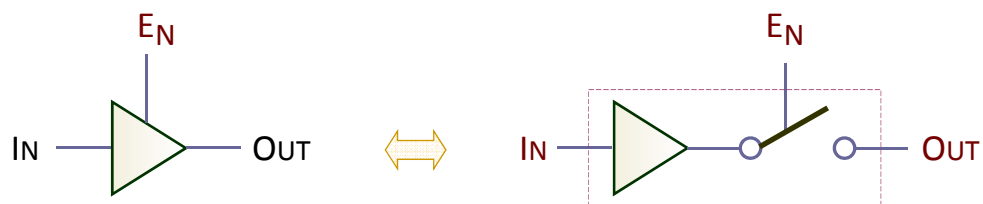
Símbolo lógico de UM PENC de 10 entradas.



PENC de 10 entradas sintetizado a partir de PENC 4-to-2 e Mux 4-to-1. Não é utilizada a funcionalidade ENABLE dos PENC.

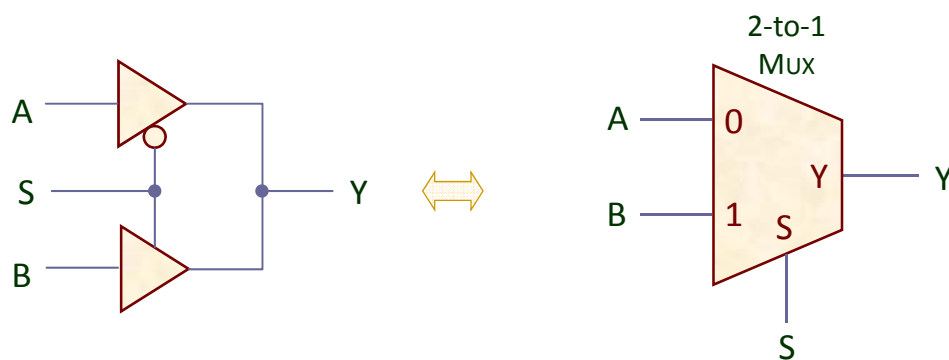
Algumas saídas de circuitos lógicos podem assumir um terceiro estado eléctrico, que não é 0 nem 1, mas sim **ALTA IMPEDÂNCIA (HIGH-Z)** ou estado **FLUTUANTE**. Este estado equivale a ter a saída desligada do resto do circuito. Os circuitos lógicos que têm uma saída passível de estar em alta impedância possuem uma entrada de controlo (ENABLE) que define se a saída está em HIGH-Z ou não. Essa entrada de ENABLE pode ser do tipo ACTIVE-HIGH ou ACTIVE-LOW.

Uma saída desta natureza designa-se de **THREE-STATE** (ou **TRI-STATE**, de 3 estados) .



Símbolo lógico e analogia mecânica do controle ENABLE de um BUFFER TRI-STATE.

Quando na entrada de controle ENABLE ( $E_N$ ) do tipo ACTIVE-HIGH se aplica um 0 a saída da porta fica em ALTA IMPEDÂNCIA, e tudo se passa como se a saída estivesse desligada (ou seja, não é possível identificar 1s e 0s na saída).



MULTIPLEXER 2-to-1 implementado à custa de 2 BUFFERS TRI-STATE com ENABLES de sinal contrário.

Podem ligar-se as saídas de duas ou mais portas TRI-STATE desde que se controlem devidamente as portas por forma a que **uma e apenas uma** das portas funcione de cada vez, e as outras mantenham as suas saídas em alta impedância. Esta operação de selecção de uma das saídas das portas de cada vez designa-se por **multiplexagem temporal**.

Um **BUFFER TRI-STATE** (ou THREE-STATE) é um circuito com uma entrada e uma saída, que coloca na saída a entrada (negada ou não), ou em alternativa coloca a saída em alta impedância.

Há 4 variantes de buffer tri-state :

- Buffer com a entrada  $E_N$  e a saída do tipo ACTIVE-HIGH – não opera nenhuma transformação lógica no sinal de saída.
- Buffer com a entrada  $E_N$  do tipo ACTIVE-HIGH e a saída do tipo ACTIVE-LOW (conversor de polaridade).
- Buffer com a entrada  $E_N$  do tipo ACTIVE-LOW e a saída do tipo ACTIVE-HIGH .
- Buffer com a entrada  $E_N$  e a saída ambas do tipo ACTIVE-LOW (conversor de polaridade).



Símbolos lógicos de buffer tri-state.

$E_N$	IN	OUT
0	0	Z
0	1	Z
1	0	0
1	1	1

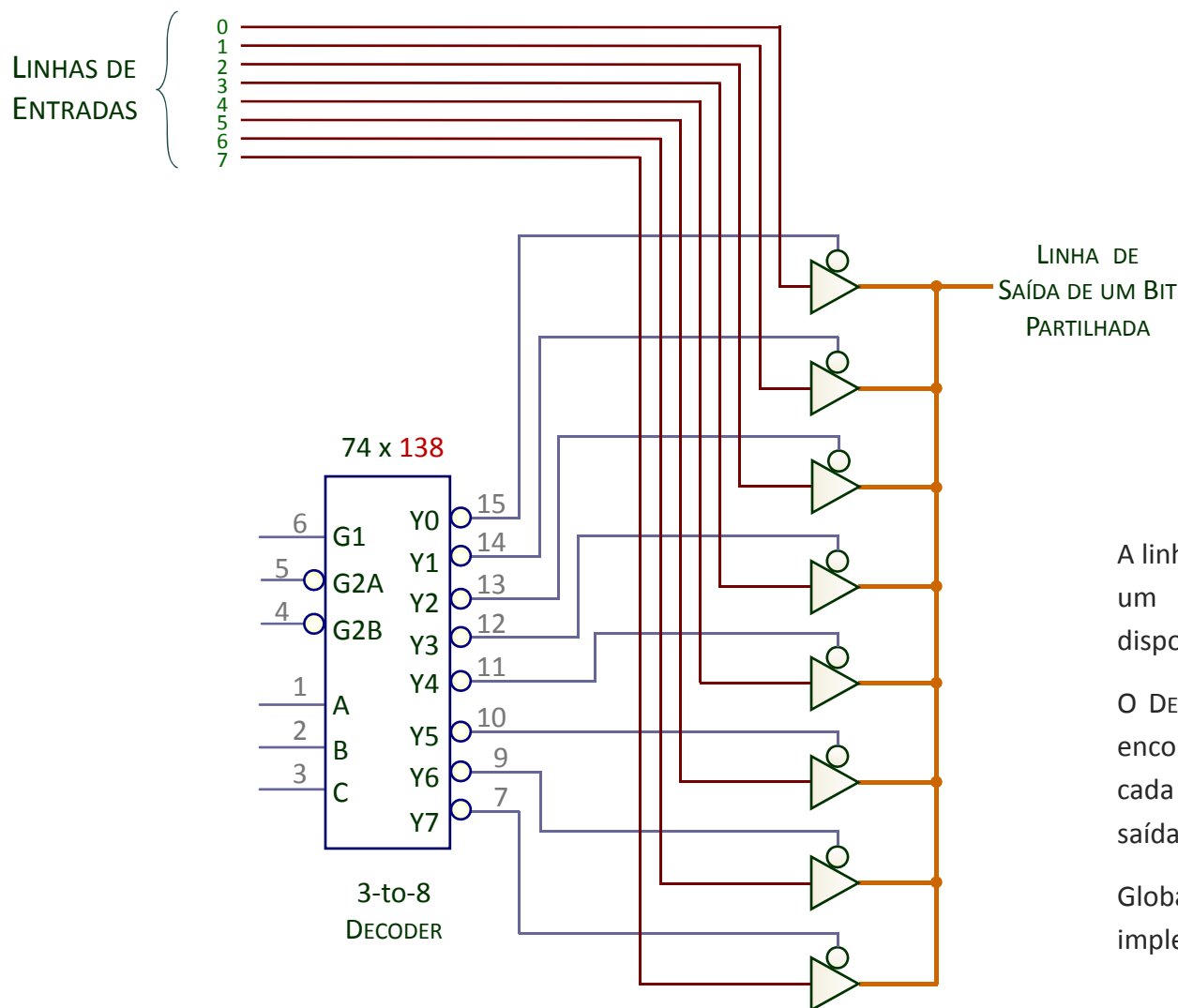
$E_N$	IN	OUT
0	0	Z
0	1	Z
1	0	1
1	1	0

$E_N$	IN	OUT
0	0	0
0	1	1
1	0	Z
1	1	Z

$E_N$	IN	OUT
0	0	1
0	1	0
1	0	Z
1	1	Z

Tabelas de verdade dos vários tipos de buffer tri-state.





Os BUFFERS TRI-STATE permitem ligar várias saídas a um mesmo ponto dum circuito, desde que apenas um  $E_N$  esteja activo em cada instante.

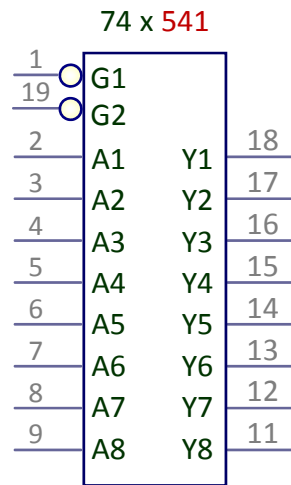
Esta situação é necessária quando vários dispositivos se ligam a um linha partilhada (comum).

A linha de saída partilhada é um barramento (bus) de um único elemento que comunica com vários dispositivos distribuídos espacialmente.


O DECODER 3-to-8 assegura que os vários buffers se encontrem activos sempre em exclusão – um de cada vez – anulando as hipóteses de conflito entre as saídas de cada um.

Globalmente o circuito corresponde à implementação de um MULTIPLEXER 8-to-1.

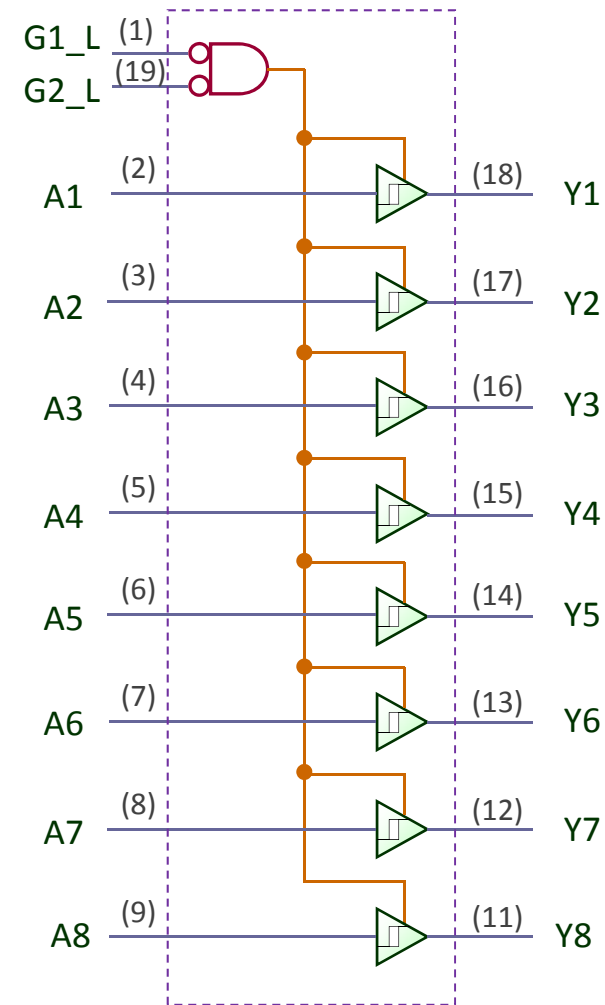
Diagrama lógico de um circuito de partilha de uma linha de saída por 8 linhas de entrada.



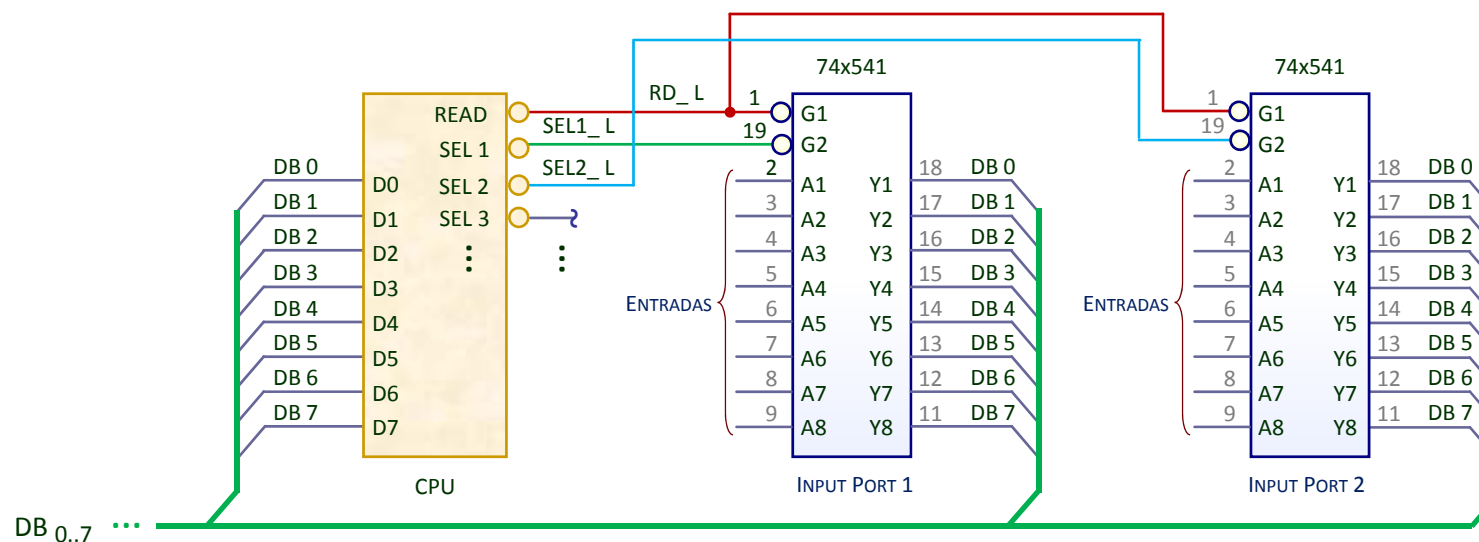
Configuração dos pinos do OCTAL  
NON-INVERTING BUFFER 74x541.

Os símbolos rectangulares (  ) dentro dos símbolos dos buffers individuais na Fig. ao lado significam que as entradas dos buffers têm um comportamento do tipo Schmitt-trigger devido à existência de histerese.

A **histerese** (no valor de 0,4 V) é uma característica eléctrica das entradas de alguns circuitos que reforça a sua imunidade ao ruído.



Estrutura interna do OCTAL  
NON-INVERTING BUFFER 74x541.



Exemplo de utilização do buffer octal 74x541 como Porto de Entrada num sistema microprocessador.

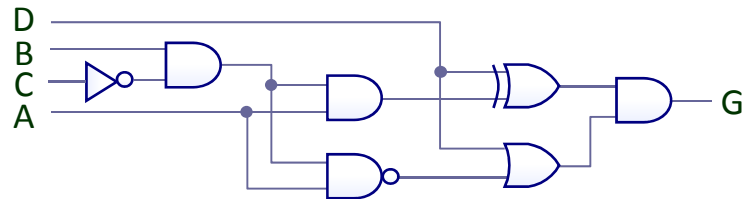
O Porto de Entrada 1 (Input Port 1) é seleccionado pelo CPU (Central Processor Unit) através da activação simultânea do sinal SEL1\_L e do sinal RD\_L (Read).

Quando essa activação simultânea acontecer, o buffer tri-state 74x541 correspondente é activado colocando no barramento de dados DB (Data Bus), em baixa impedância e em simultâneo, os 8-bits a serem lidos.

# IMPLEMENTAÇÃO DE UMA FUNÇÃO UTILIZANDO MÓDULOS COMBINATÓRIOS MSI (Ex. 4-3-1)

4-44

Exemplo 4-3



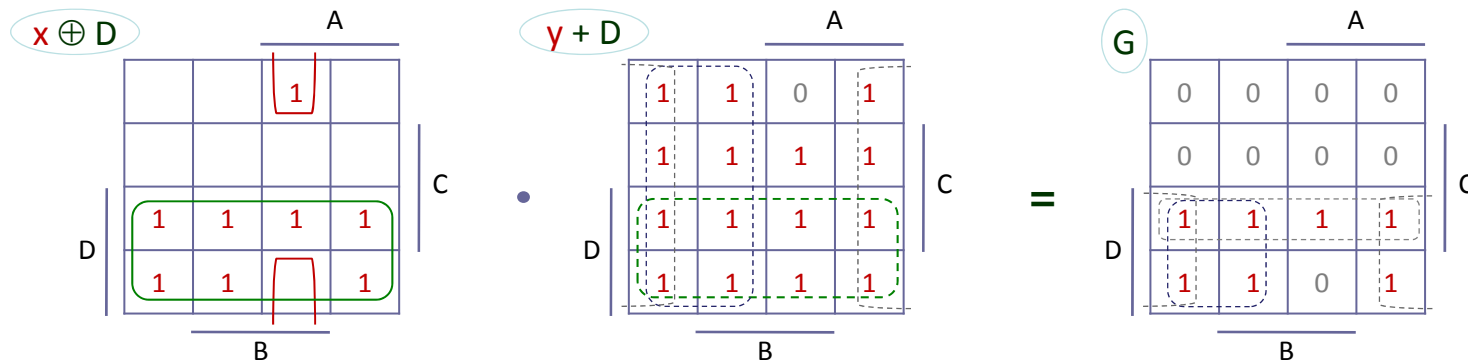
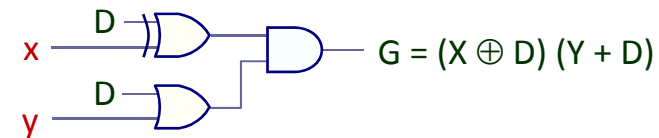
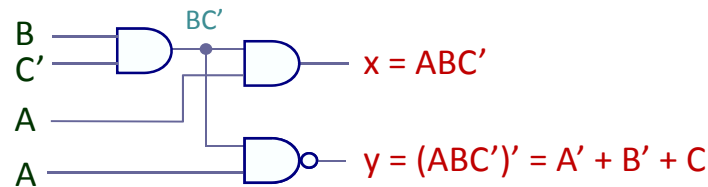
## OBJECTIVO

Para o circuito da figura ao lado:

1. Obter a forma AND-OR simplificada da função **G**.
2. Implementar G utilizando exclusivamente portas **NAND de 2 entradas**.
3. Implementar G apenas com **MULTIPLEXERS 4x1**.

Para esta implementação **não dispõe de complemento das variáveis**.

1) Implementação na forma **AND-OR** simplificada.

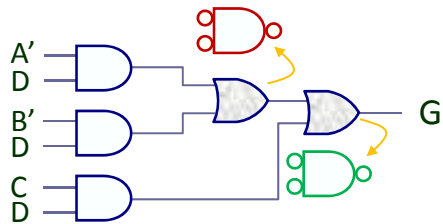


$$G = A'D + B'D + CD$$

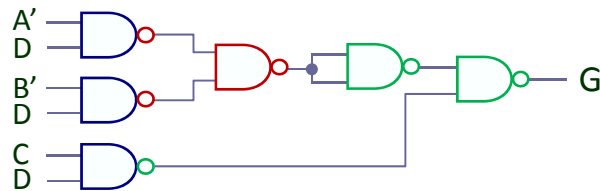


Exemplo 4-3

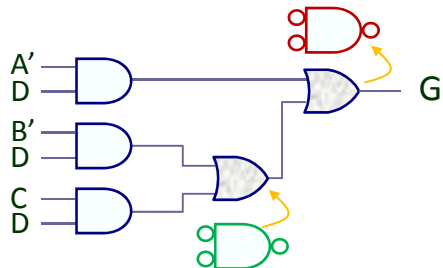
2) Implementação só portas **NAND de 2 entradas**.



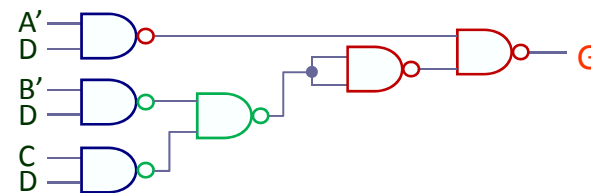
$$G = (A'D + B'D) + CD$$



6 portas NAND de 2 entradas, 2 versões tirando partido da associatividade da porta OR.



$$G = A'D + (B'D + CD)$$



Recorda-se que:

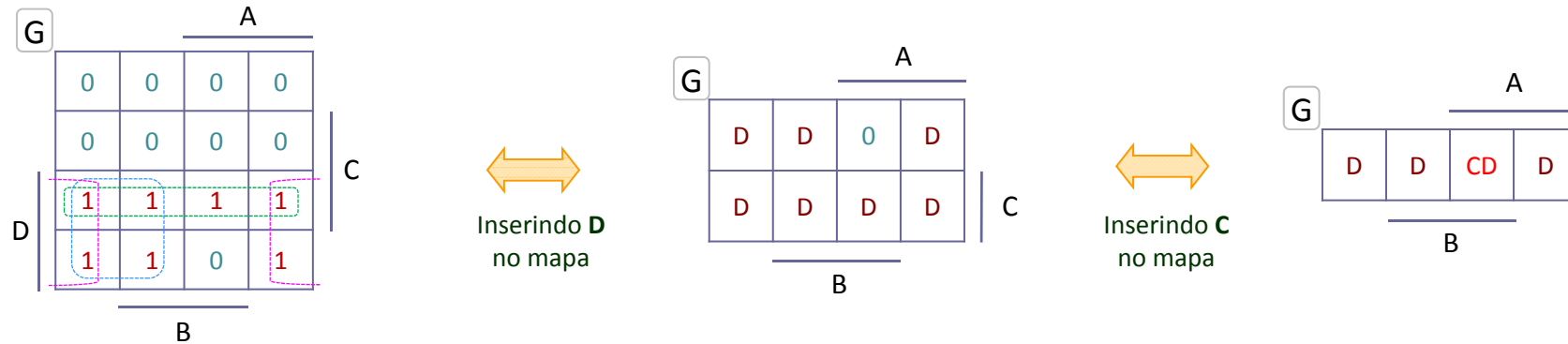


# IMPLEMENTAÇÃO DE UMA FUNÇÃO UTILIZANDO MÓDULOS COMBINATÓRIOS MSI (Ex. 4-3-3)

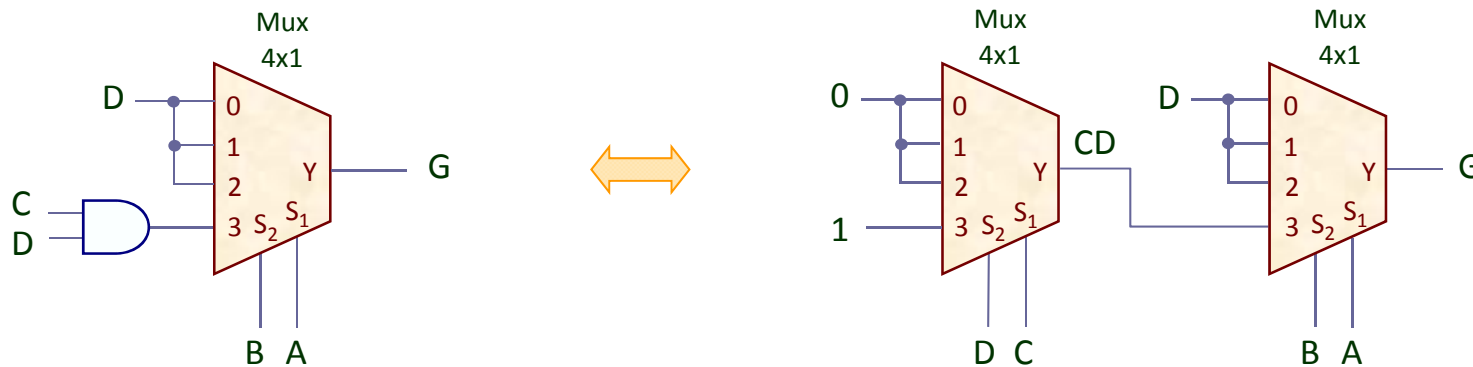
4-46

Exemplo 4-3

- 3) Implementação só com MULTIPLEXERS DE 4x1, não se dispondo de complemento das variáveis.



$$G = A'D + B'D + CD$$



1. **LSD 4 – MÓDULOS COMBINATÓRIOS MSI**
2. Descodificador (Decoder)
3. Descodificadores com Saídas Active-High e Active-Low
4. Descodificador 74x138
5. Descodificador 3-to-8 em PAL 22V10
6. Geração de Mintermos e Maxtermos com Descodificadores
7. Implementação de Funções com Descodificadores
8. Descodificador de 16 Saídas – Expansão a partir de Descodificadores de 8 Saídas
9. Descodificadores de 16 e de 64 Saídas – Formas de Expansão
10. BCD-to-7-Segment Decoder/Driver 74x147
11. Decoder/Drivers com Saídas Active-high e Active-low
12. Multiplexer
13. Multiplexer de 2 Entradas
14. Multiplexer de 4 Entradas
15. Multiplexers 74x153 e 74x157
16. Multiplexer 74x151
17. Multiplexer de 8 Entradas em PAL 22V10
18. Implementação de Funções com Multiplexers (Ex. 4-1-1)
19. Implementação de Funções com Multiplexers (Ex. 4-1-2)
20. Implementação de Funções com Multiplexers (Ex. 4-1-3)
21. Implementação de Funções com Multiplexers (Ex. 4-2-1)
22. Implementação de Funções com Multiplexers (Ex. 4-2-2)
23. Realização de Portas Lógicas Básicas com Multiplexers
24. Realização de uma Porta Lógica XOR com Multiplexers
25. Realização de uma Unidade Funcional com Multiplexer



26. Multiplexers de 8 e de 12 Entradas – Formas de Expansão
27. Multiplexer de 64 Entradas – Expansão Unidimensional e Bidimensional
28. Demultiplexer
29. Canal de Comunicação com Multiplexer e Demultiplexer
30. Demultiplexer de 32 Saídas – Expansão (Cascading)
31. Demultiplexers de 64 Saídas – Expansão Unidimensional e Bidimensional
32. Encoder (Codificador)
33. Priority Encoder (Penc)
34. Priority Encoder 74x148
35. Priority Encoder 74x148
36. Priority Encoder de 8 Entradas em PAL 22V10
37. Priority Encoder de 16 Entradas – Expansão a partir de Priority Encoders de 8 Entradas
38. Priority Encoder de 10 Entradas – Expansão
39. Saídas Tri-State
40. Tri-state Buffers – Variantes Active High e Active Low
41. Tri-state Buffers – Partilha de uma Linha
42. Tri-state Octal Non-Inverting Buffer 74LS151
43. Utilização do Tri-state Octal Non-Inverting Buffer 74LS151 num Porto de Entrada
44. Implementação de uma função utilizando Módulos Combinatórios MSI (Ex. 4-3-1)
45. Implementação de uma função utilizando Módulos Combinatórios MSI (Ex. 4-3-2)
46. Implementação de uma função utilizando Módulos Combinatórios MSI (Ex. 4-3-3)
47. LSD – 4 Índice 1
48. LSD – 4 Índice 2

