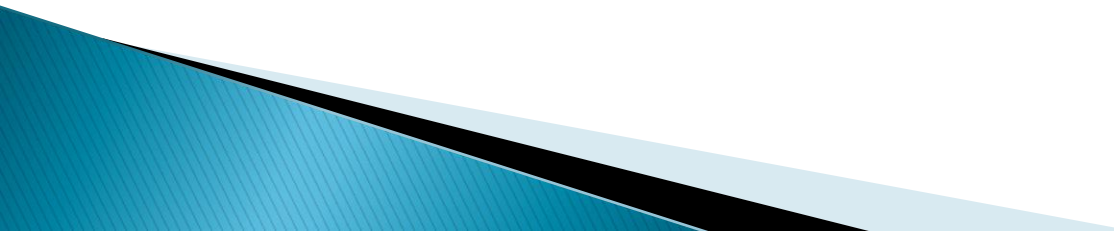


# Capítulo 2–Projeto Lógico Combinacional F

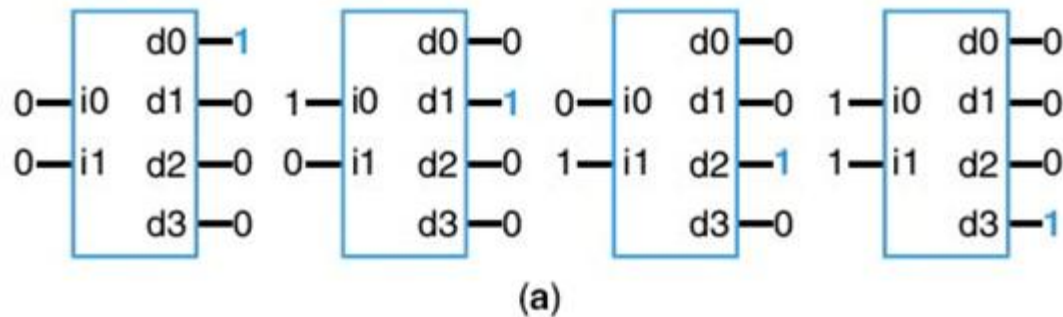
Profa. Eliete Caldeira

# Módulos combinacionais

- ▶ Sistemas digitais complexos são implementados como redes modulares
  - ▶ Módulos–padrão: correspondem a funções identificadas como úteis para uma grande variedade de aplicações e
  - ▶ Estão disponíveis como componentes de prateleira e biblioteca, ou seja, prontos para usar
  - ▶ Módulos–padrão: decodificadores, codificadores, multiplexadores, demultiplexadores, deslocadores
  - ▶ Módulos–padrão aritméticos: somadores, subtratores, Unidades Lógicas e Aritméticas (ALUs), comparadores, multiplicadores,
- 

# Decodificadores (DECOD)

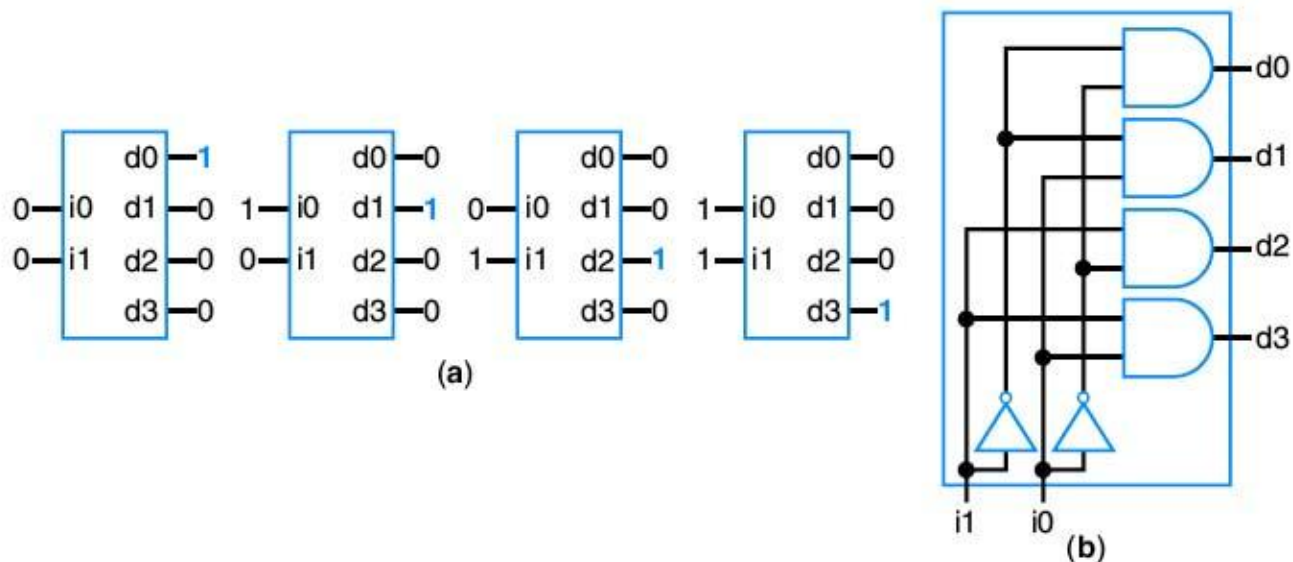
- ▶ Decodifica de binários em n bits colocando uma e apenas uma das  $2^n$  saídas em 1
- ▶ Um DECOD 2x4



- ▶ Qual a expressão das saídas d0, d1, d2 e d3?

# Decodificadores (DECOD)

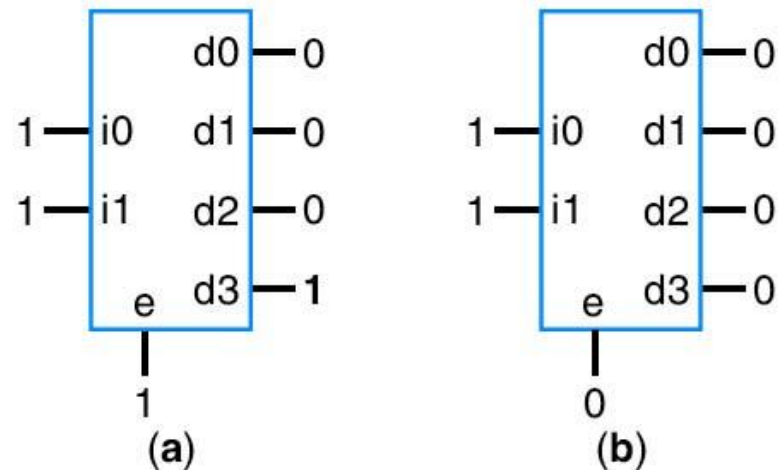
- ▶ Decodifica de binários em n bits colocando uma e apenas uma das  $2^n$  saídas em 1
- ▶ Um DECOD 2x4



**Figure 2.50** 2x4 decoder: (a) outputs for possible input combinations, (b) internal design.

# Decodificador com *enable*

- ▶ Quando a entrada *enable* (habilita) é 1, o circuito funciona normalmente e quando *enable* é 0, todas as saídas são 0.

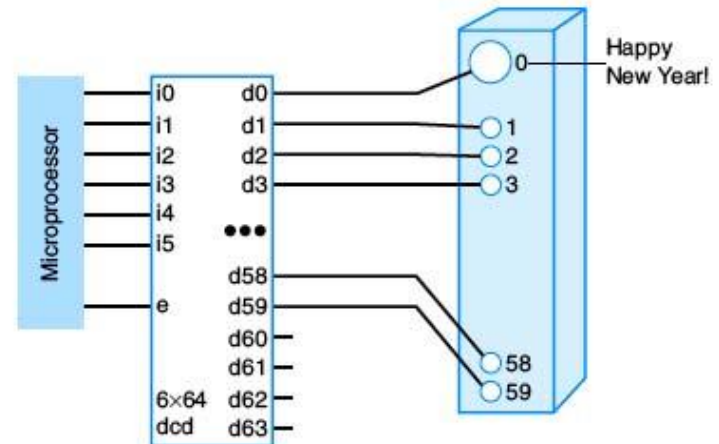


**Figure 2.51** Decoder with enable: (a)  $e=1$ : normal decoding, (b)  $e=0$ : all outputs 0.

# Exemplo 1

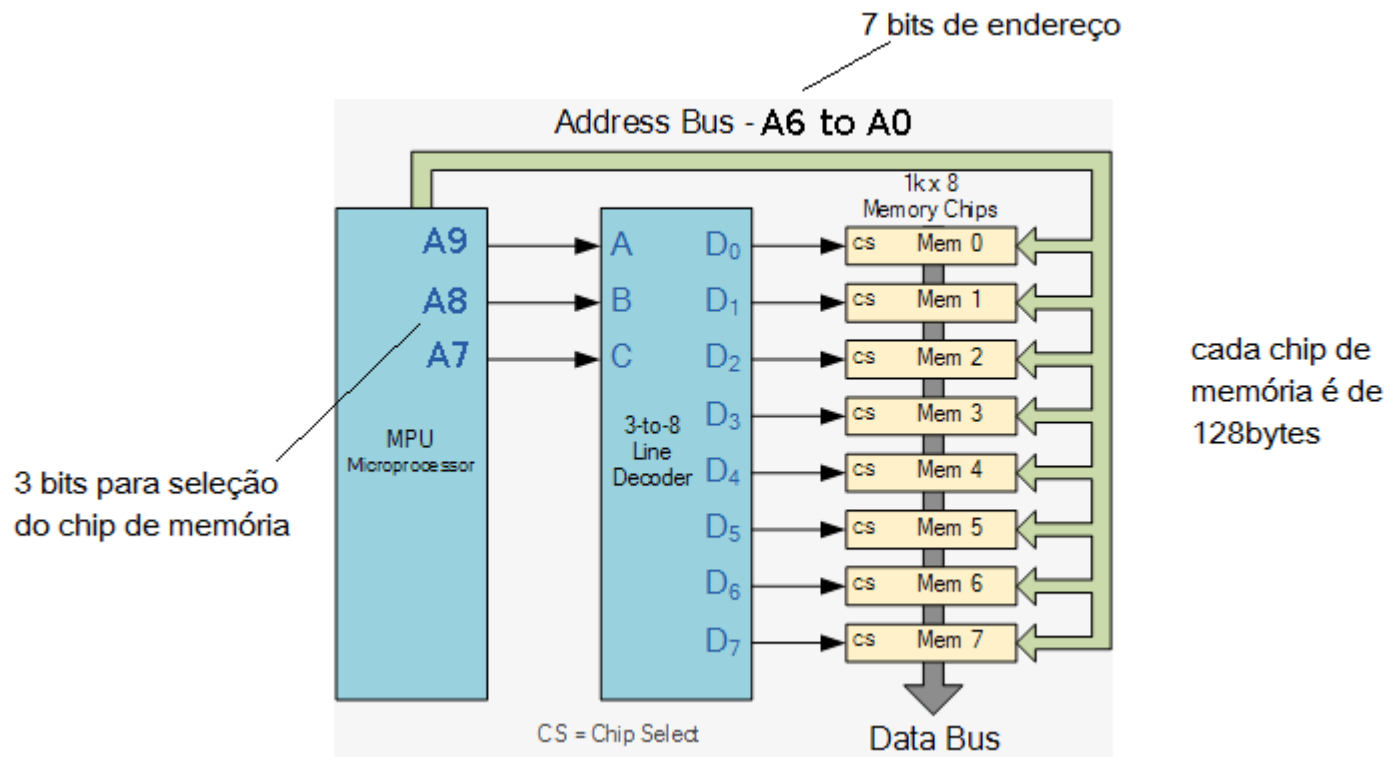
- ▶ O circuito da figura vai realizar a contagem regressiva de ano novo.
  - Queremos que apenas uma das lâmpadas acenda por vez, começando com a 59 e terminando com a 0.
  - O microprocessador conta de 59 a 0 em binário (6 bits  $\Rightarrow 2^6 = 64$ ) e o decodificador escolhe que lâmpada acende.
  - O sinal de *enable* deixa todas as lâmpadas apagadas até que seja o minuto final.

**Figure 2.52** Using a 6x64 decoder to interface a microprocessor and a column of lights for a New Year's Eve display. The microprocessor sets  $e = 1$  when the last minute countdown begins, and then counts down from 59 to 0 in binary on the pins  $i5 \dots i0$ . Note that the microprocessor should never output 60, 61, 62, or 63 on  $i5 \dots i0$ , and thus those outputs of the decoder go unused.



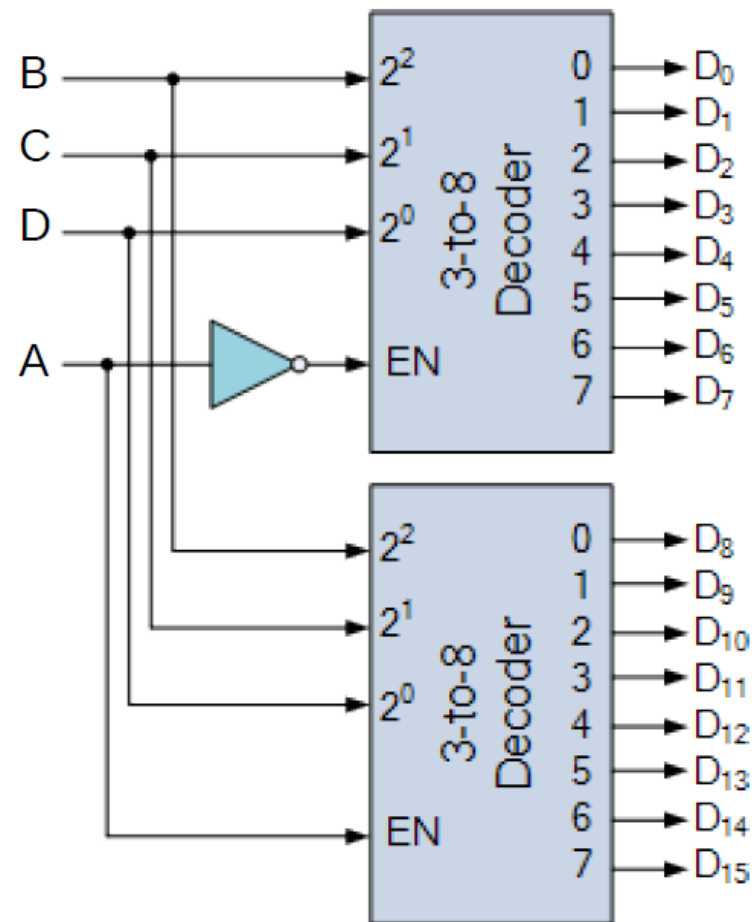
# Exemplo 2

## ► Endereçamento de memória





# Decod 4x16 usando decods 3x8



4-to-16 Line Decoder Implemented  
with two 3-to-8 Decoders



# Decodificador e porta OR: conjunto universal

- ▶ Um decod de  $n$  entradas e uma porta OR podem implementar qualquer função de  $n$  variáveis
- ▶ Como a  $i$ -ésima saída corresponde ao mintermo  $m_i$ , basta fazer o OR de todos os mintermos da função

# Decodificador e porta OR: conjunto universal

- Exemplo: Projetar o circuito que implementa as funções  $z_1$ ,  $z_2$  e  $z_3$  na tabela usando um Decod 3x8 e 3 portas OR

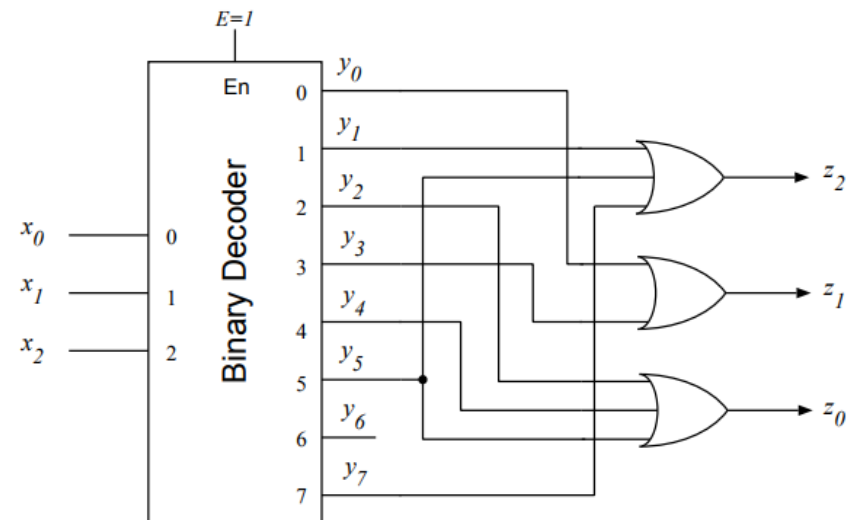
$x_2x_1x_0$	$z_2$	$z_1$	$z_0$
000	0	1	0
001	1	0	0
010	0	0	1
011	0	1	0
100	0	0	1
101	1	0	1
110	0	0	0
111	1	0	0

$$(y_7, \dots, y_0) = \text{DEC}(x_2, x_1, x_0, 1)$$

$$z_2(x_2, x_1, x_0) = y_1 + y_5 + y_7$$

$$z_1(x_2, x_1, x_0) = y_0 + y_3$$

$$z_0(x_2, x_1, x_0) = y_2 + y_4 + y_5$$



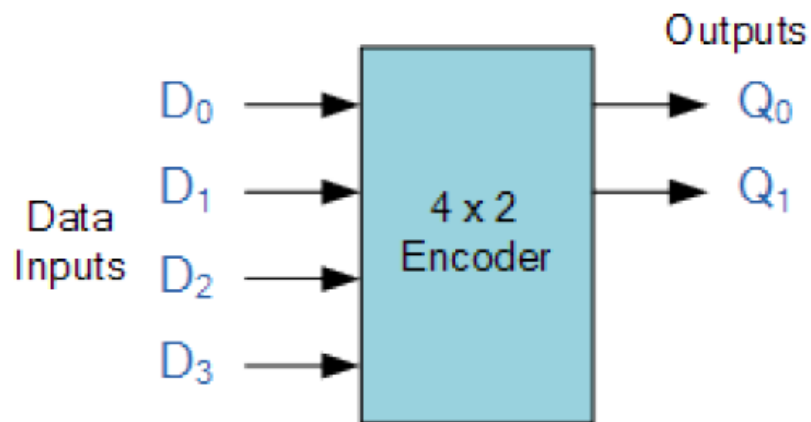
# Decodificador e porta OR: conjunto universal

- ▶ Exemplo: Projetar o circuito que implementa as funções  $z_1$ ,  $z_2$  e  $z_3$  na tabela usando um Decod 3x8 e 3 portas OR

$x_2x_1x_0$	$z_2$	$z_1$	$z_0$
000	0	1	0
001	1	0	0
010	0	0	1
011	0	1	0
100	0	0	1
101	1	0	1
110	0	0	0
111	1	0	0

# Codificador binário (encoder)

- ▶ Codificador  $2^n \times n \rightarrow$  Fornece o código binário em  $n$  bits correspondente à entrada que está ativada

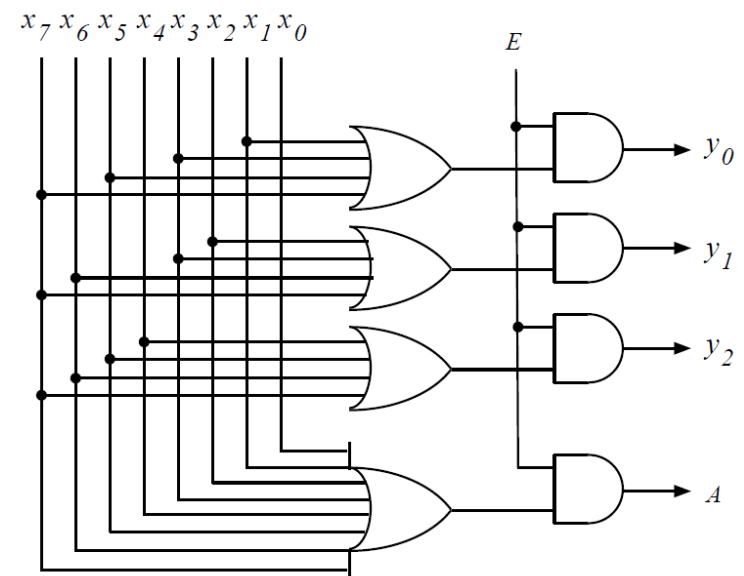


Inputs				Outputs	
$D_3$	$D_2$	$D_1$	$D_0$	$Q_1$	$Q_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

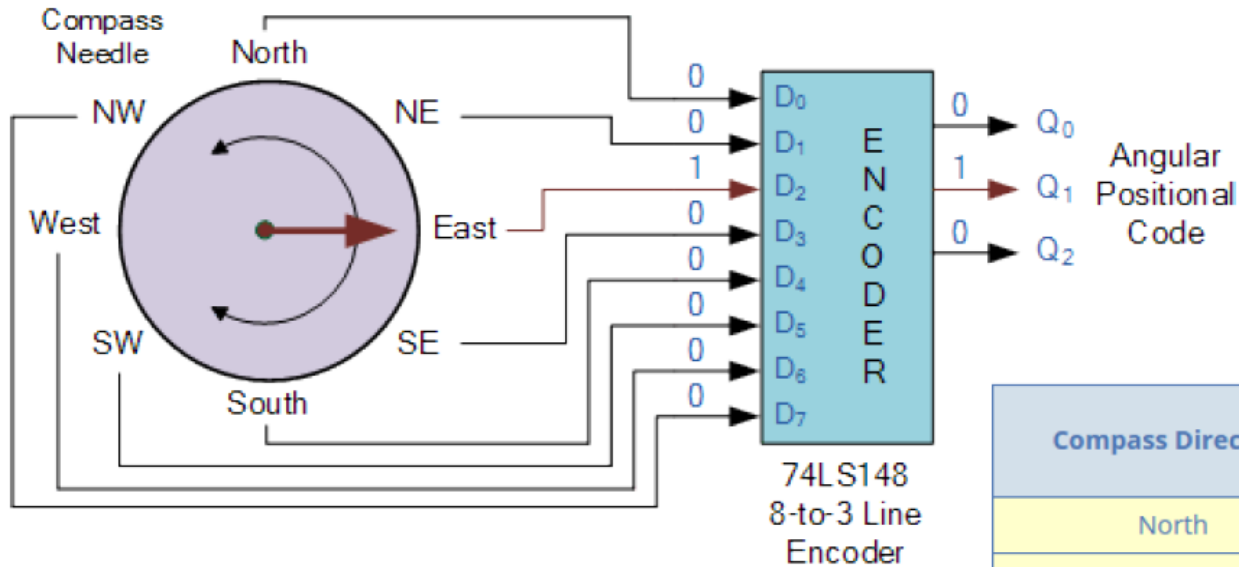
# Codificador binário (encoder)

- ▶ Codificador 8x3 com *enable*: circuito interno considerando que apenas uma entrada fica ativa por vez

$E$	$x_7$	$x_6$	$x_5$	$x_4$	$x_3$	$x_2$	$x_1$	$x_0$	$y$	$y_2$	$y_1$	$y_0$	$A$
1	0	0	0	0	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	1	0	1	0	0	1	1
1	0	0	0	0	0	1	0	0	2	0	1	0	1
1	0	0	0	0	1	0	0	0	3	0	1	1	1
1	0	0	0	1	0	0	0	0	4	1	0	0	1
1	0	0	1	0	0	0	0	0	5	1	0	1	1
1	0	1	0	0	0	0	0	0	6	1	1	0	1
1	1	0	0	0	0	0	0	0	7	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	-	-	-	-	0	0	0	0	0

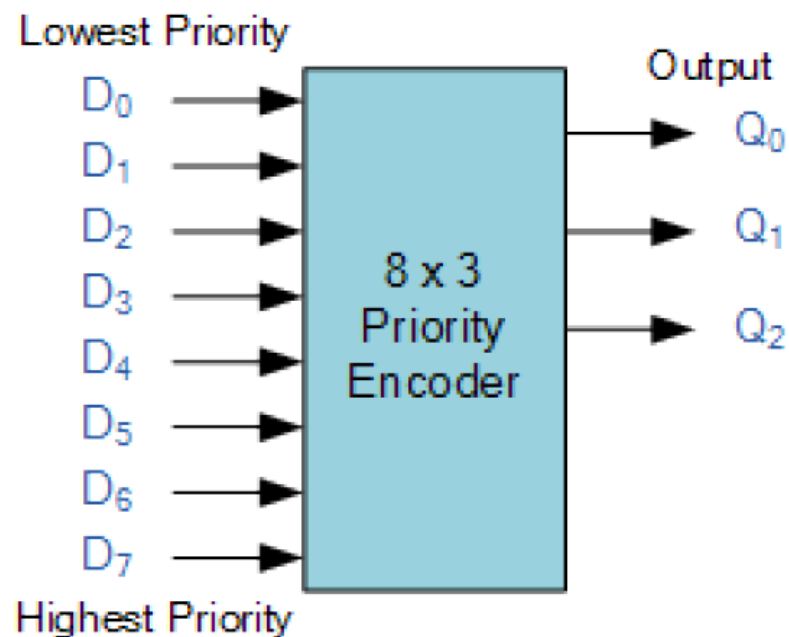


# Codificador – Exemplo de uso



Compass Direction	Binary Output		
	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
North	0	0	0
North-East	0	0	1
East	0	1	0
South-East	0	1	1
South	1	0	0
South-West	1	0	1
West	1	1	0
North-West	1	1	1

# Codificador de Prioridade



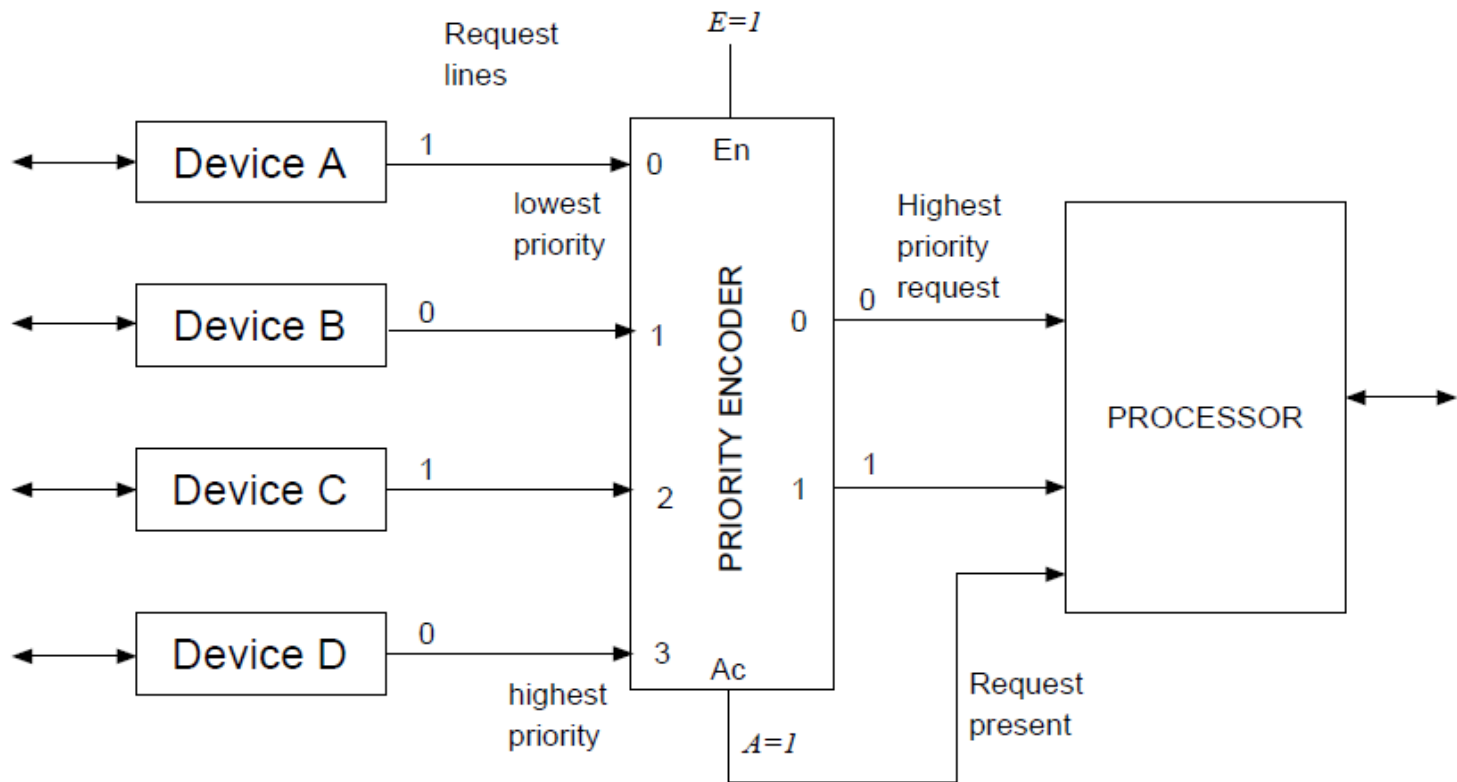
Inputs								Outputs		
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = don't care



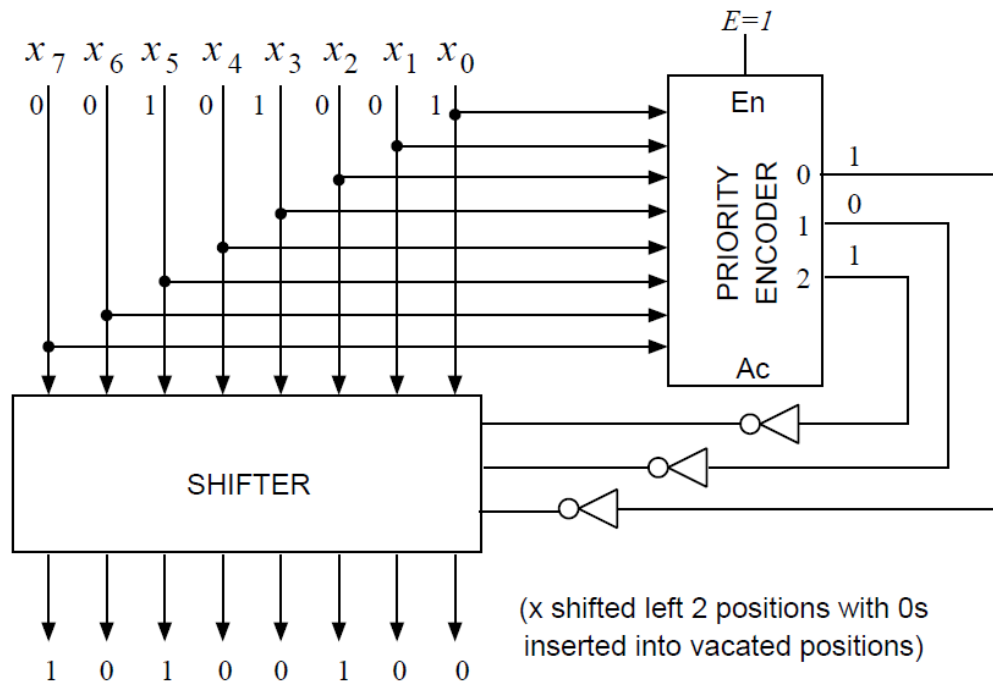
# Codificador de prioridade – uso

- ▶ Selecionar interrupção de maior prioridade



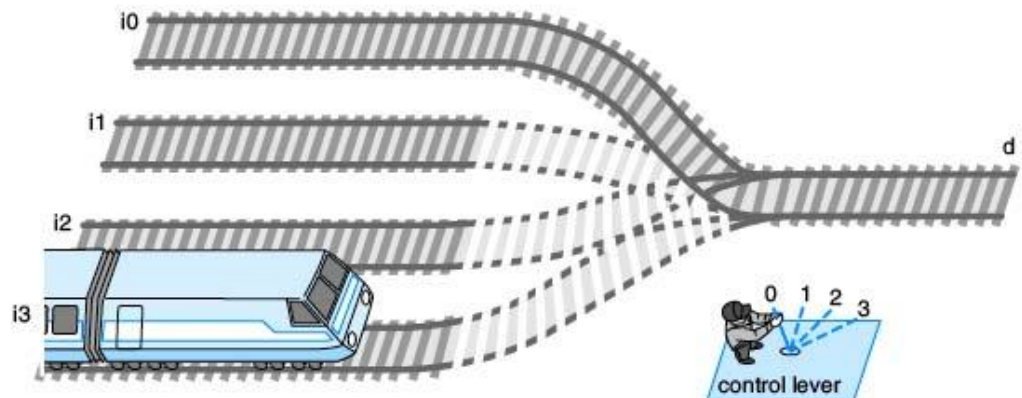
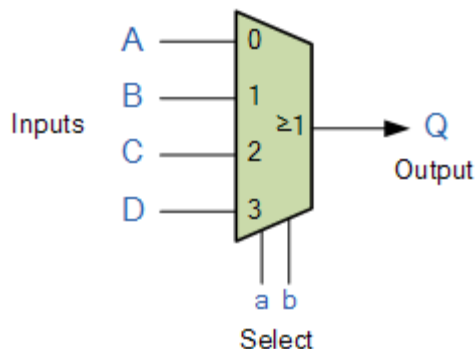
# Codificador de prioridade – uso

- ▶ Detectar a posição do bit 1 de mais à esquerda (ou mais à direita) em um número e deslocar até que não tenha mais 0's à esquerda



# Multiplexador (MUX)

- ▶ Um multiplexador Mx1 tem M entradas de dados e 1 saída
- ▶ Permite que apenas 1 das entradas seja passada para a saída → MUX = seletor
  - MUX 2x1 → 1 entrada de seleção e 2 entradas de dados
  - MUX 4x1 → 2 entradas de seleção e 4 de dados
  - MUX 8x1 → 3 entradas de seleção e 8 de dados
  - MUX Nx1 →  $\log_2(N)$  entradas de seleção e N de dados



**Figure 2.53** A multiplexer is like a railyard switch, determining which input track connects to the single output track, according to the switch's control lever.

# MUX – circuito interno

►  $s_0$  é o bit de seleção

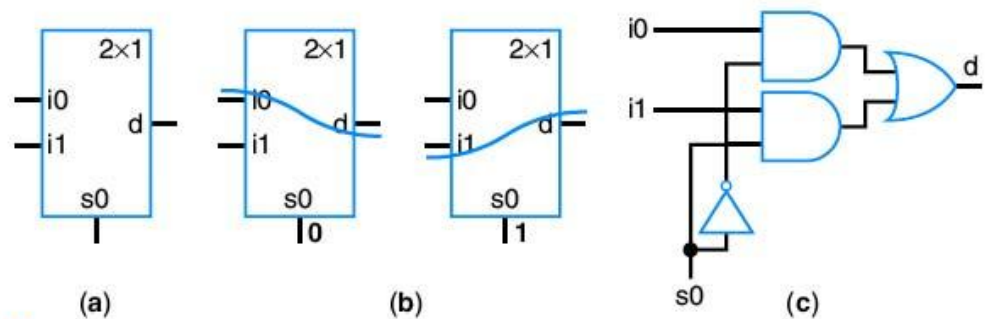


Figure 2.54 2 x 1 multiplexer: (a) block symbol, (b) connections for  $s_0=0$ , and  $s_0=1$ , and (c) internal design.

►  $s_1$  e  $s_0$  são bits de seleção

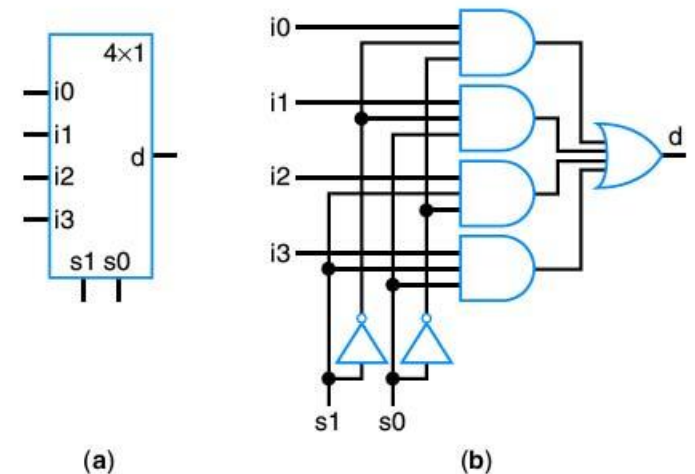
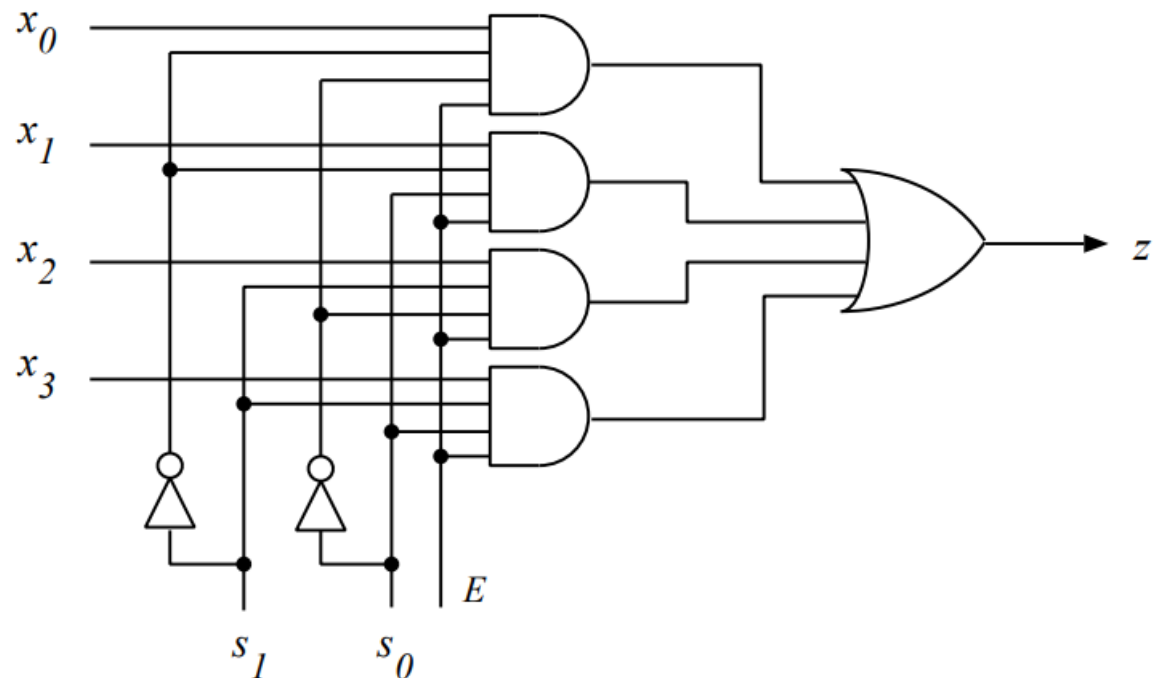


Figure 2.55 4 x 1 multiplexer: (a) block symbol and (b) internal design.

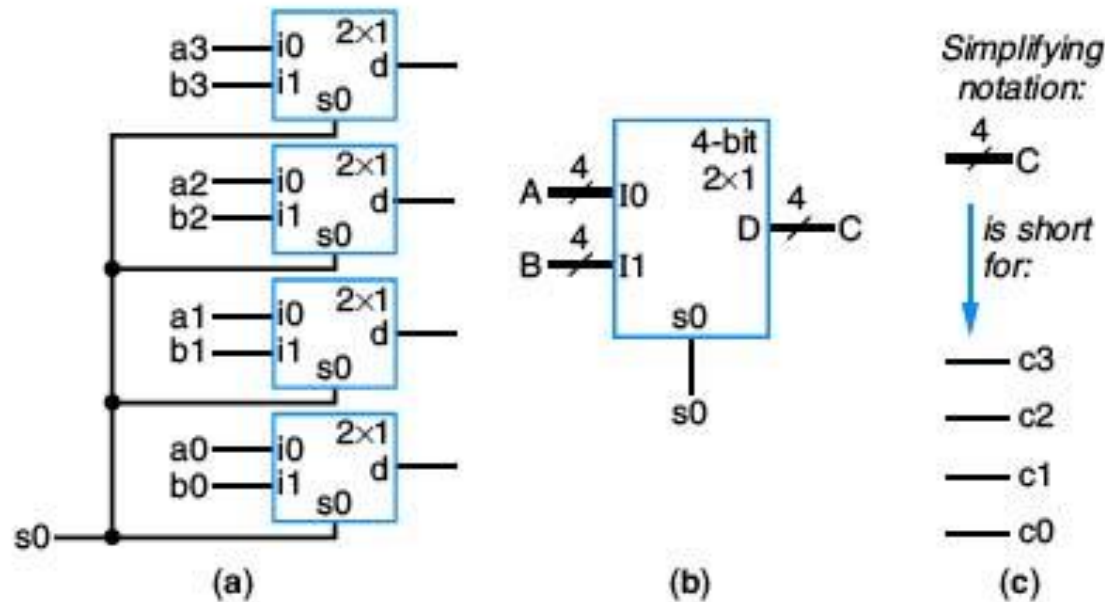
# MUX com *enable*

- Tabela verdade e circuito interno

$E$	$s_1$	$s_0$	$z$
1	0	0	$x_0$
1	0	1	$x_1$
1	1	0	$x_2$
1	1	1	$x_3$
0	-	-	0



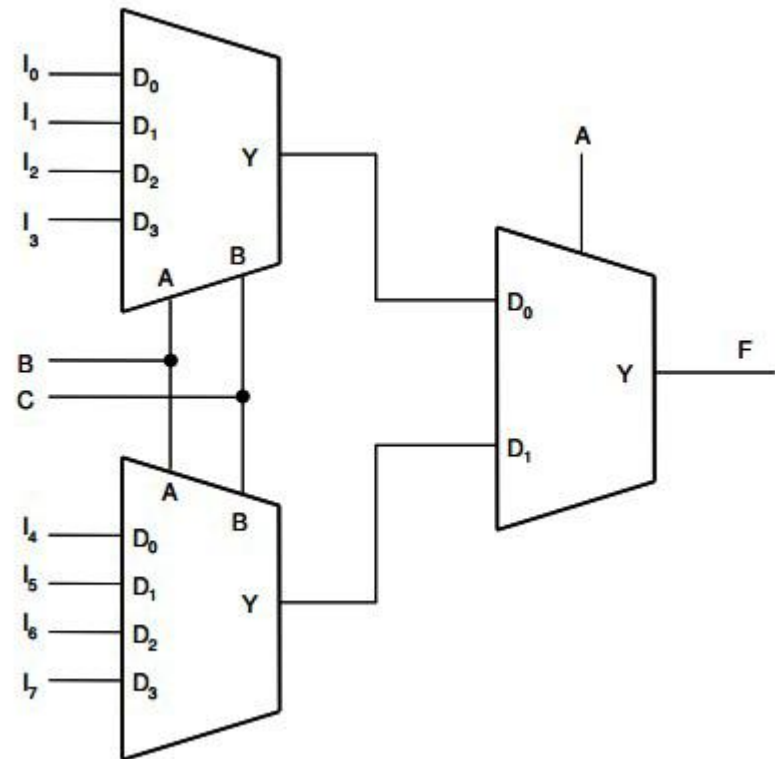
# Mux de Mx1 de N bits



**Figure 2.57** 4-bit 2x1 mux: (a) internal design using four 2x1 muxes for selecting among 4-bit data items A or B, and (b) block diagram of a 4-bit 2x1 mux component. (c) The block diagram uses a common simplifying notation, using one thick wire with a slanted line and the number 4 to represent 4 single wires.

# Implementação em árvore de MUX

- Implementação em árvore de MUX 8x1 usando dois MUXes 4:1 e um MUX 2:1





# Implementação em árvore de MUX

- Implementação em árvore de MUX 16x1 usando MUXes 4:1

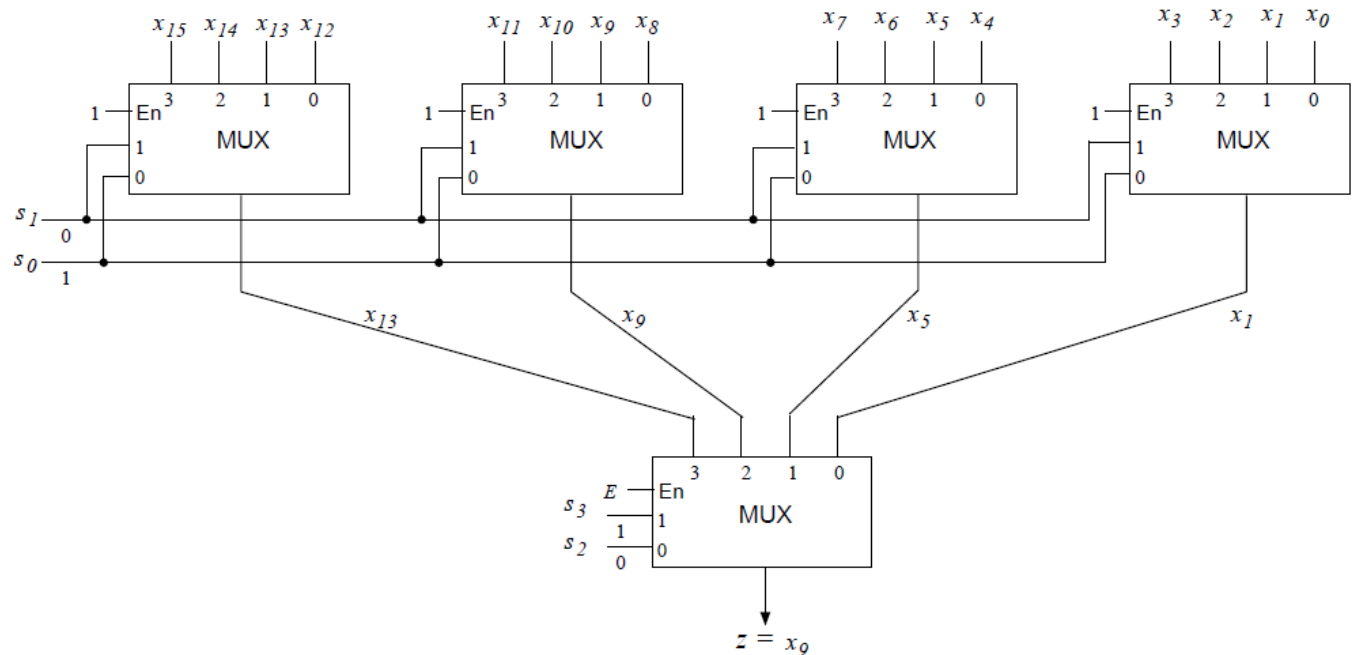
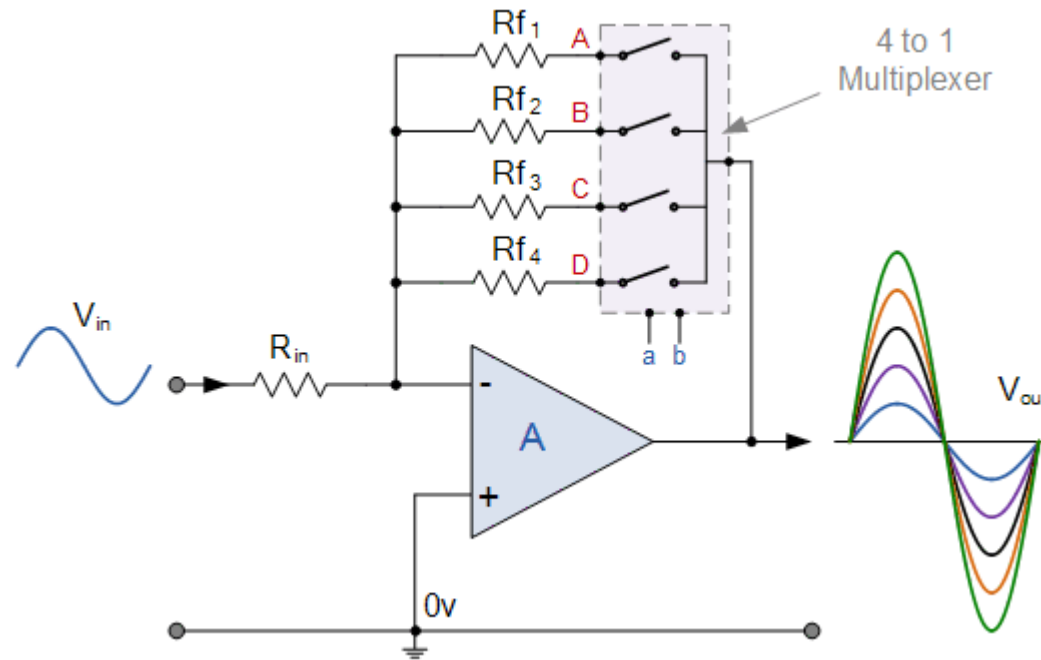


Figure 9.24: TREE IMPLEMENTATION OF A 16-INPUT MULTIPLEXER.

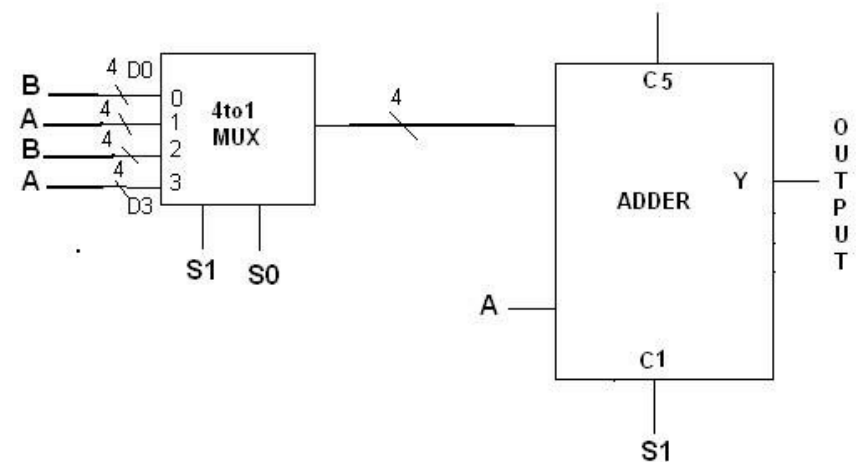
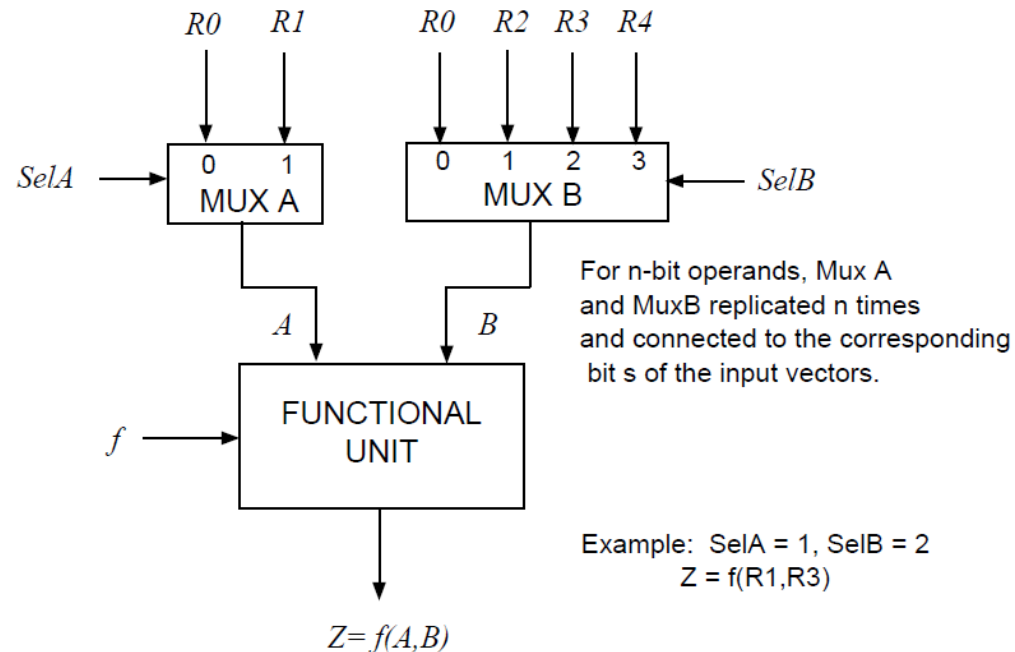
# Exemplo de uso

- ▶ Ajuste digital do ganho de um amplificador na configuração inversora



# Exemplo de uso

## ► Seleção de dados



# MUX como bloco universal

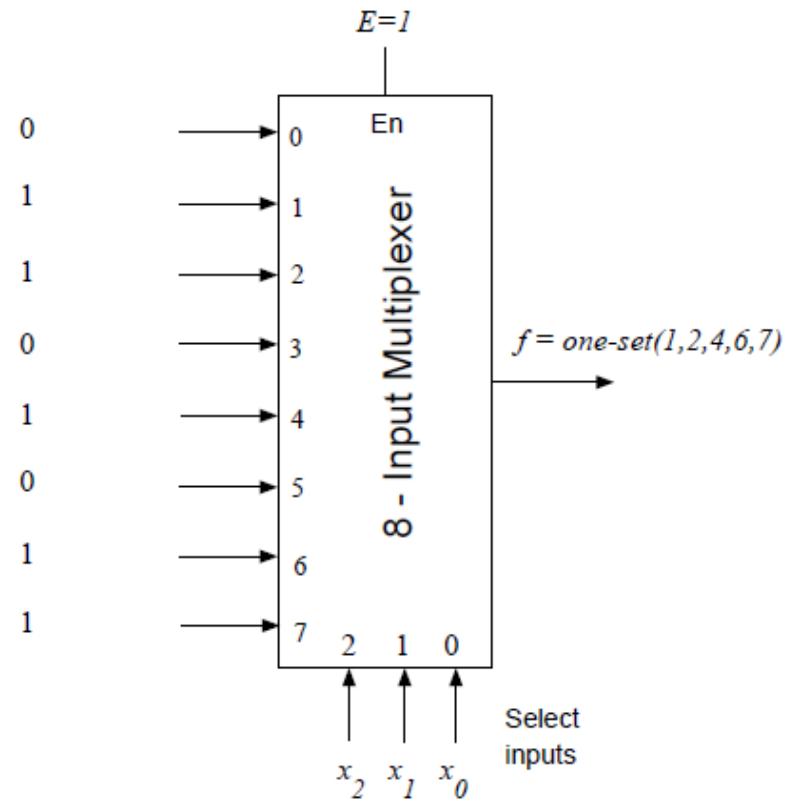
- ▶ Conecte as  $n$  variáveis da função às  $n$  entradas de seleção do MUX
- ▶ Conecte 0 ou 1 nas entradas de dados de acordo com a tabela-verdade da função

# Exemplo

- ▶ Implemente a função E usando um MUX 8x1
  - $E(x_2, x_1, x_0) = \sum m(1, 2, 4, 6, 7)$

# Exemplo

- ▶ Implemente a função E usando um MUX 8x1
  - $E(x_2, x_1, x_0) = \sum m(1, 2, 4, 6, 7)$



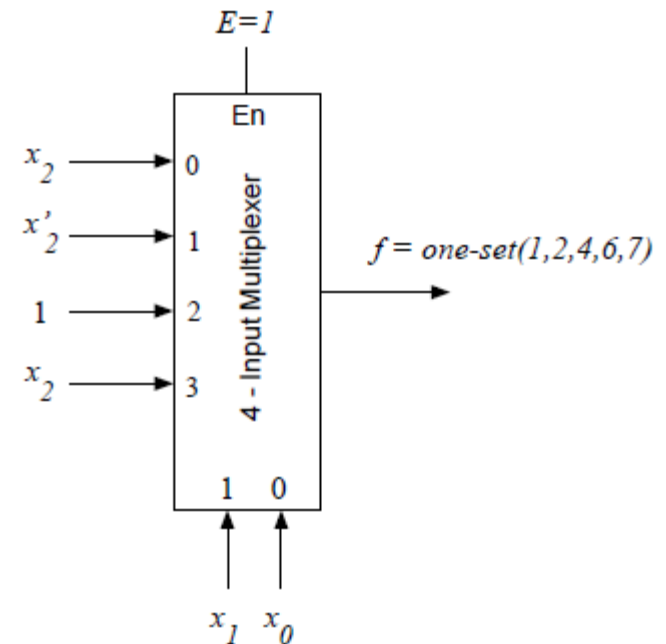
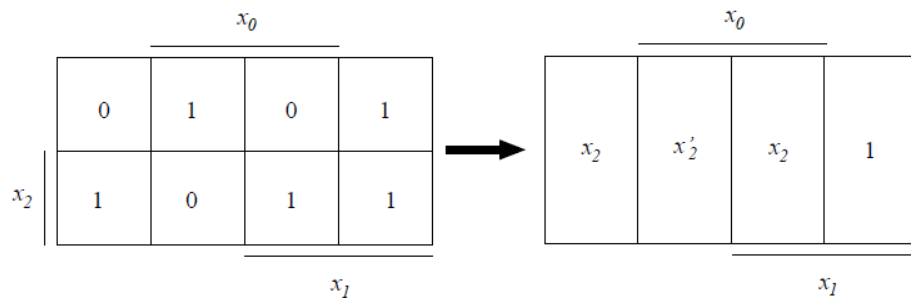
# Exemplo

- ▶ Implemente a função E usando um MUX 4x1 com a variável  $x_2$  nas entradas de dados
  - $E(x_2, x_1, x_0) = \Sigma m(1, 2, 4, 6, 7)$



# Exemplo

- ▶ Implemente a função E usando um MUX 4x1 com a variável  $x_2$  nas entradas de dados
  - $E(x_2, x_1, x_0) = \Sigma m(1, 2, 4, 6, 7)$



# Exercício

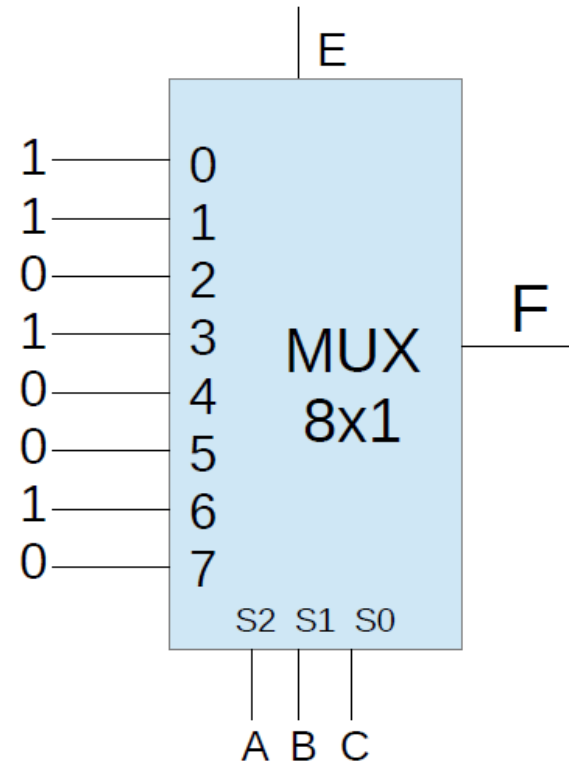
- ▶ Implemente a função na tabela-verdade usando um MUX 8x1

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

# Exercício

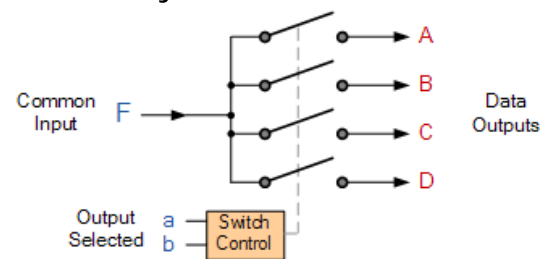
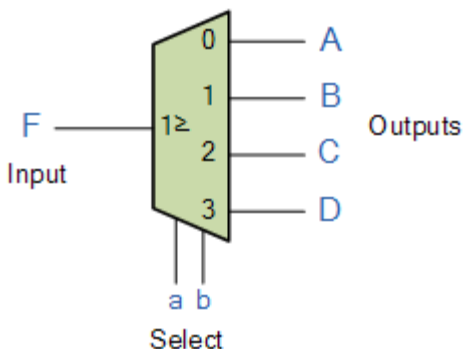
- Implemente a função na tabela-verdade usando um MUX 8x1

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



# Demultiplexador (DEMUX)

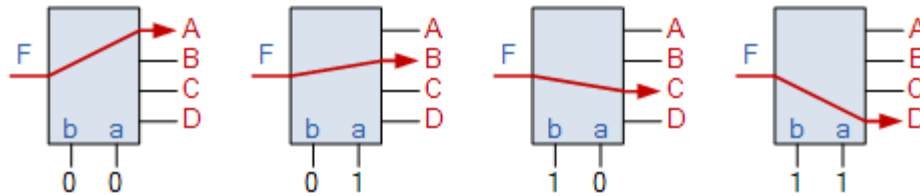
- ▶ Um demultiplexador 1xM tem 1 entrada de dados e M saídas
- ▶ Faz com que a entrada seja direcionada para apenas 1 das saídas de acordo com o código seletor
  - DEMUX 1x4 → 2 entradas de seleção
  - DEMUX 1x8 → 3 entradas de seleção
  - DEMUX 1xN →  $\log_2(N)$  entradas de seleção



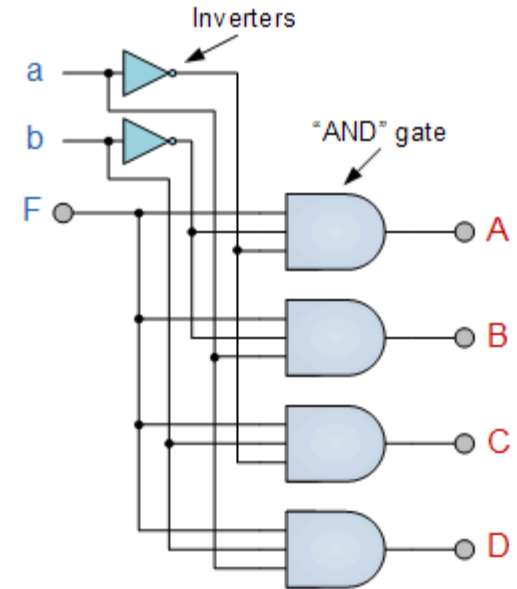
Output Select		Data Output Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

# DEMUX – circuito interno

## ► DEMUX 1x4



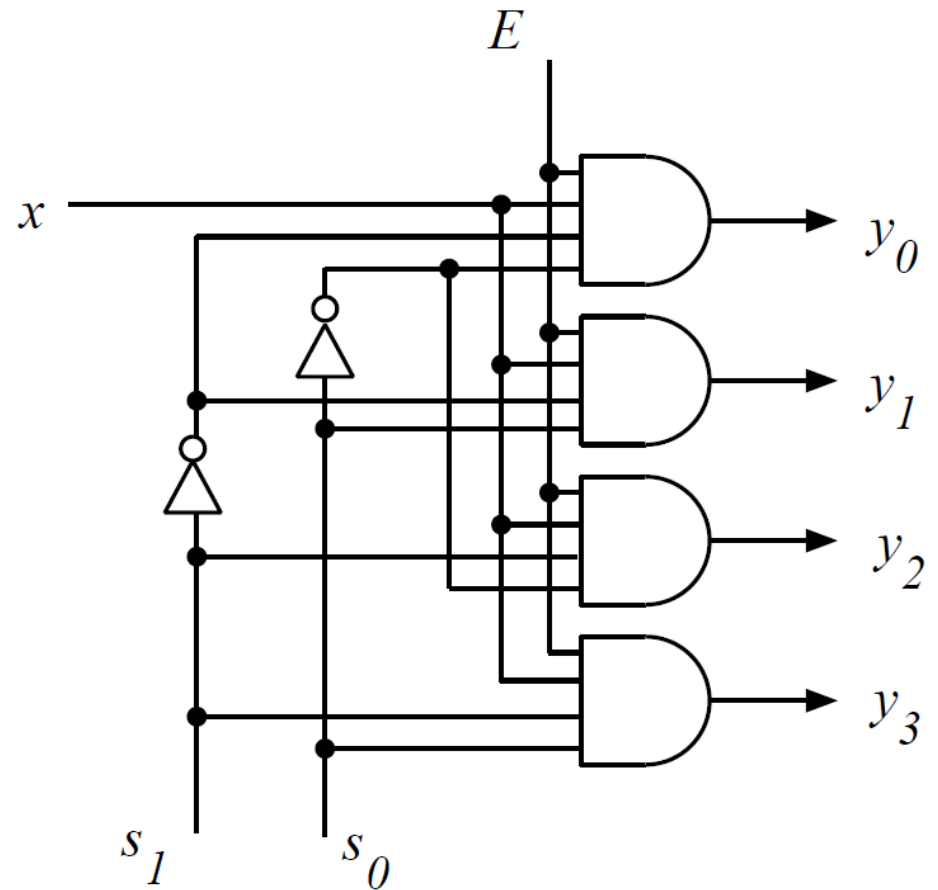
## ► DEMUX 1x4 com enable



# DEMUX – circuito interno

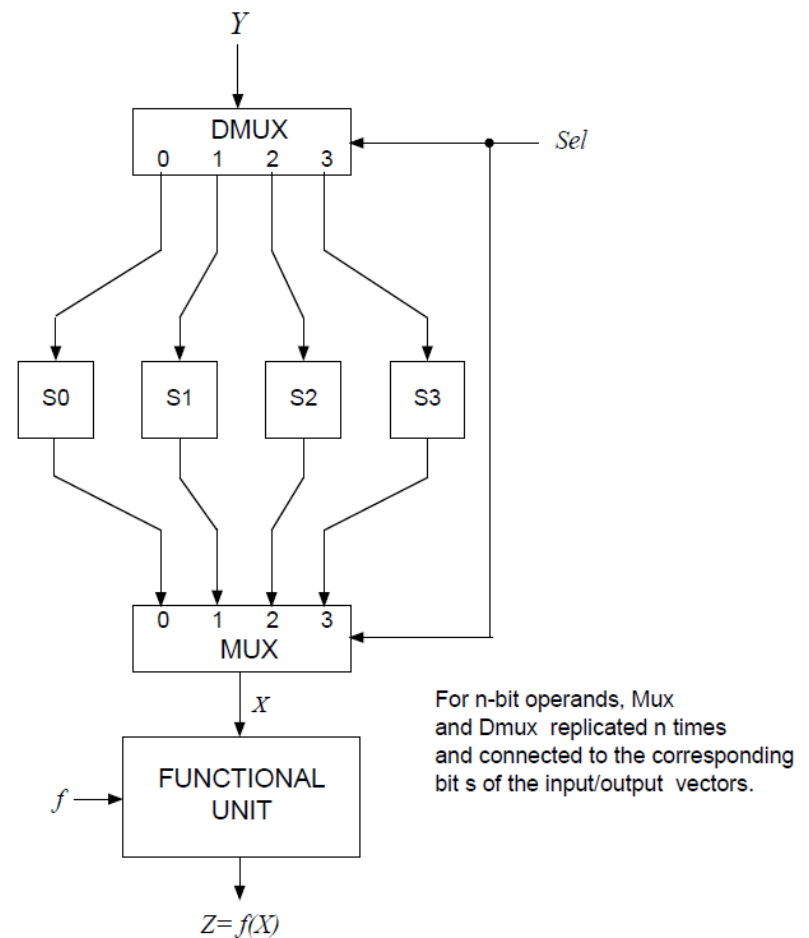
## ► DEMUX 1x4 com *enable*

$E$	$s_1$	$s_0$	$s$	$y_3$	$y_2$	$y_1$	$y_0$
1	0	0	0	0	0	0	$x$
1	0	1	1	0	0	$x$	0
1	1	0	2	0	$x$	0	0
1	1	1	3	$x$	0	0	0
0	-	-	-	0	0	0	0



# DEMUX – Exemplo de uso

- ▶ Quando a saída de uma unidade funcional deve ser distribuída para uma de diversas unidades





# DEMUX – Exemplo de uso

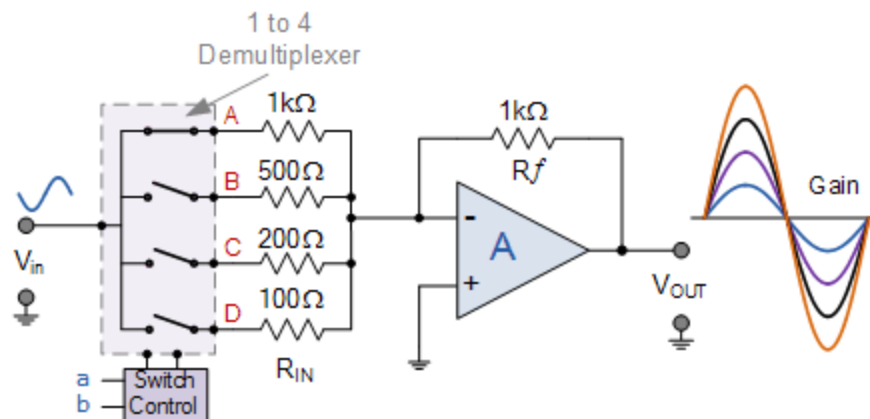
- ▶ Quando dados são transmitidos por um canal comum e têm que ser distribuídos no destino



Conceito de multiplexação.

# DEMUX – Exemplo de uso

- ▶ Ajuste digital do ganho de um amplificador na configuração inversora

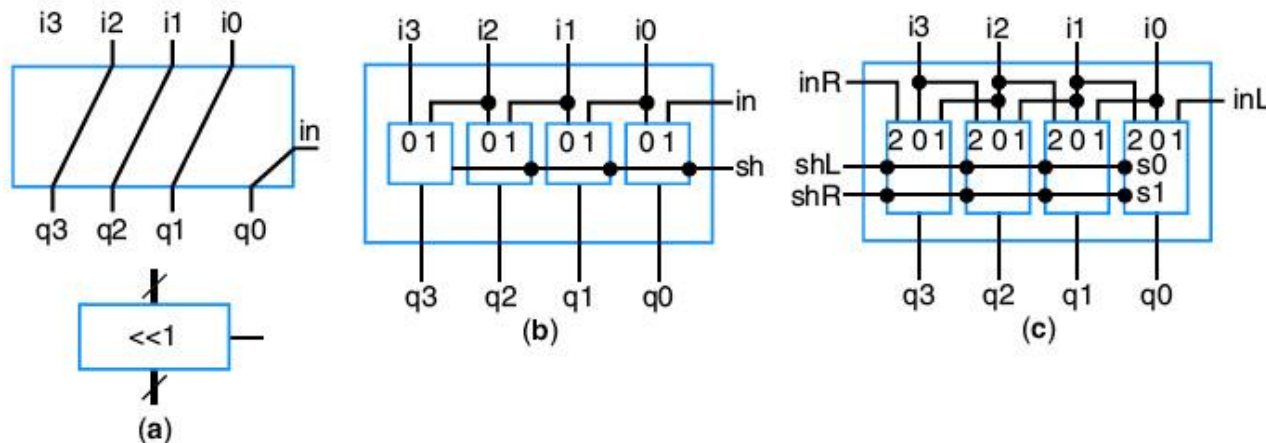


# Deslocadores (shifters)

- ▶ Um deslocador de  $n$  bits pode deslocar uma entrada de  $n$  bits um certo número de vezes para gerar uma saída de  $n$  bits
- ▶ O deslocamento pode ser à direita ou à esquerda
- ▶ Um deslocador pode ser usado para dividir ou multiplicar por potências de 2
- ▶ Também pode ser usado para alinhar um vetor de bits (ex: notação científica  $0,0001011 = 1,011 \times 2^{-4}$  ou em somas, para alinhar as mantissas  $1 \times 2^{-1} + 1 \times 2^{-4} = 0,1 + 0,0001 = 0,1001$ )

# Deslocadores (shifters)

- ▶ a) Deslocador de 1 bit à esquerda e símbolo.
  - Note que há uma entrada adicional com o novo bit a ser inserido à direita
- ▶ b) Deslocador com sh: sh=1 desloca e sh = 0, não desloca)
- ▶ c) Deslocador bidirecional



**Figure 4.39** Combinational shifters: (a) left shifter with block symbol shown at bottom, (b) left shift or pass component, (c) left/right shift or pass component.

# Deslocador simples

$d \in \{RIGHT, LEFT\}$

$s \in \{YES, NO\}$

$E \in \{0, 1\}$

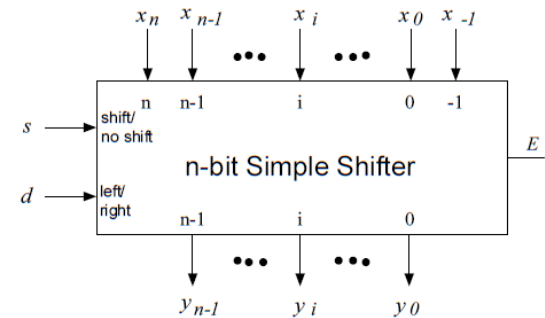
Function:

$$y_i = \begin{cases} x_{i-1} & \text{if } (d = LEFT) \text{ and } (s = YES) \text{ and } (E = 1) \\ x_{i+1} & \text{if } (d = RIGHT) \text{ and } (s = YES) \text{ and } (E = 1) \\ x_i & \text{if } (s = NO) \text{ and } (E = 1) \\ 0 & \text{if } (E = 0) \end{cases}$$

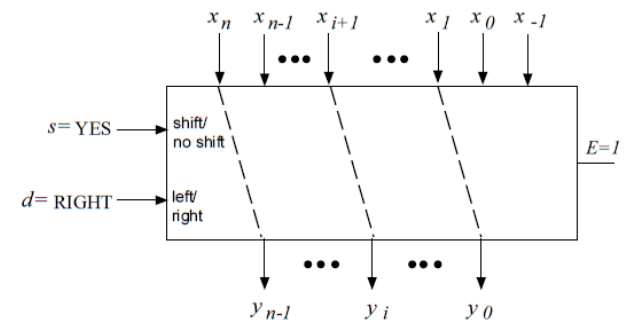
for  $0 \leq i \leq n - 1$ .

$$x_{-1} = \begin{cases} 0 & \text{left shift with 0 insert} \\ 1 & \text{left shift with 1 insert} \\ x_{n-1} & \text{left rotate} \end{cases}$$

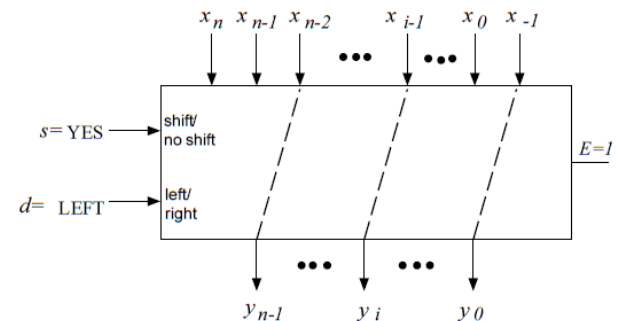
$$x_n = \begin{cases} 0 & \text{right shift with 0 insert} \\ 1 & \text{right shift with 1 insert} \\ x_0 & \text{right rotate} \end{cases}$$



(a)



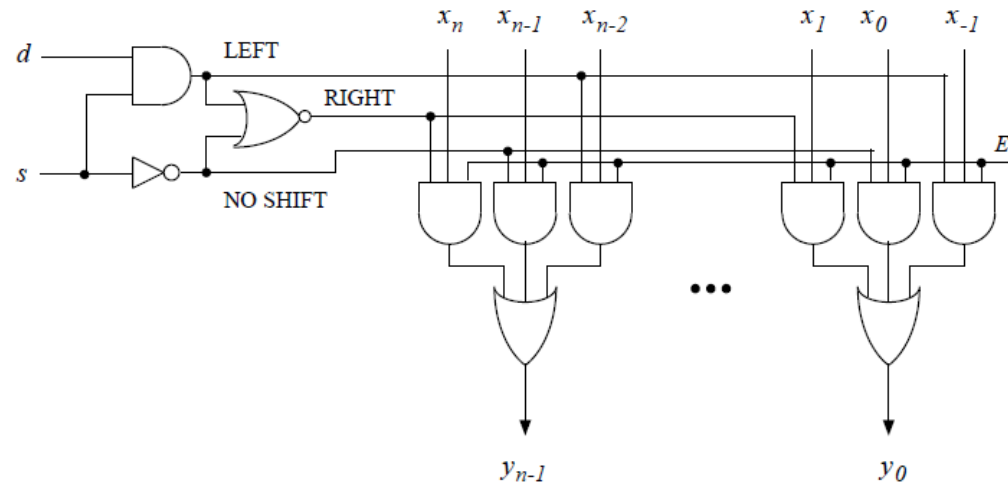
(b)



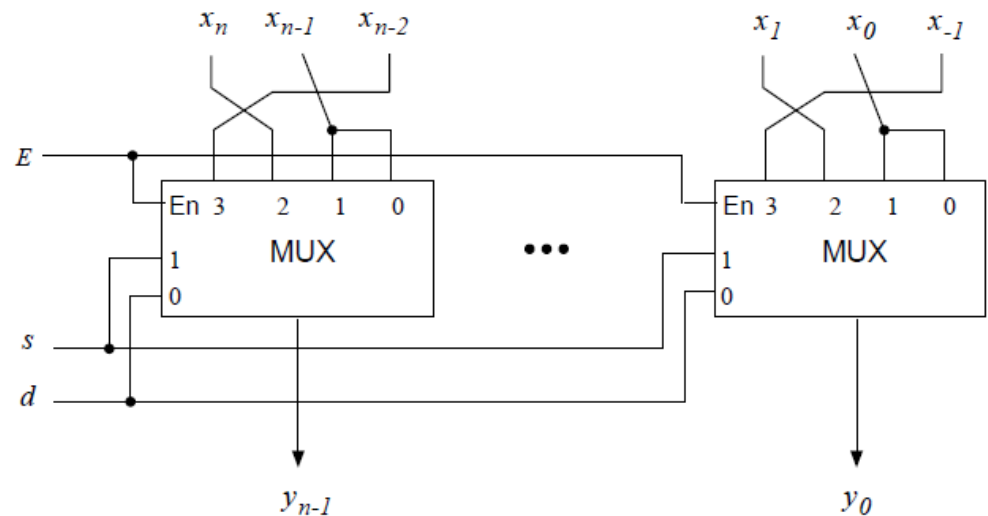
(c)

# Deslocador simples – circuito interno

- Com portas lógicas

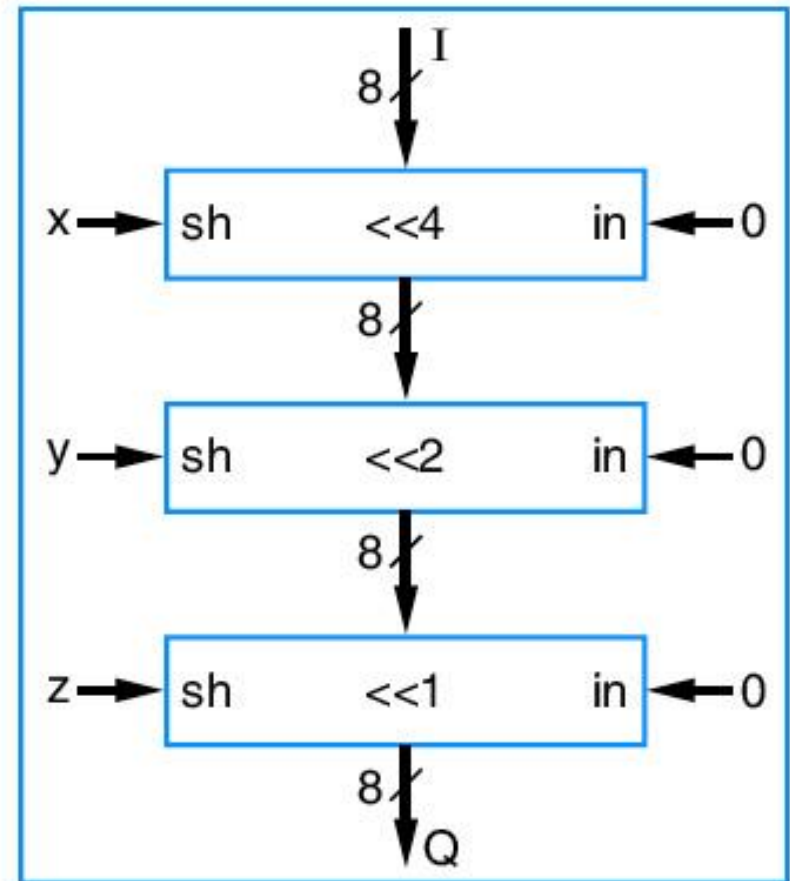


- Com MUXes



# Deslocador barrel

- ▶ Um Deslocador Barrel de N bits é:
  - Um deslocador de N bits para uso geral
  - Pode realizar deslocamentos ou rotações de qualquer número de posições
  - Deslocamentos à esquerda
  - Se  $x=1$  a entrada é deslocada 4 bits
  - Se  $y = 1$  a entrada é deslocada 2 bits
  - Se  $z=1$  a entrada é deslocada 1 bit



**Figure 4.42** 8-bit barrel shifter (left shift only).

Para ser continuado...