Introdução aos Sistemas Digitais

Blocos combinatórios de baixa e média complexidade: AOI, OAI, descodificadores, codificadores, multiplexers, desmultiplexers

ieeta

Blocos combinatórios

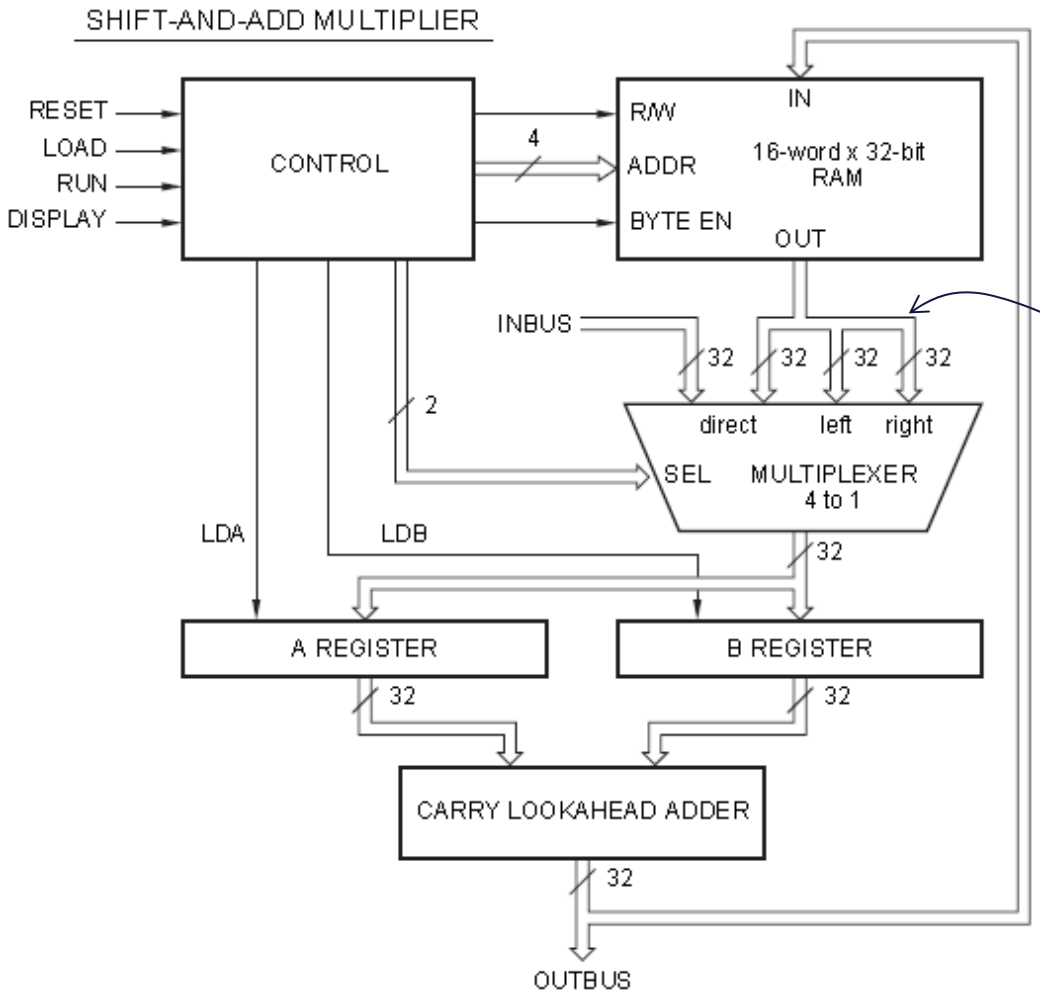
Com o aumento do número de entradas e saídas em circuitos combinatórios, torna-se impossível descrevê-los com tabelas de verdade e sintetizar a partir deste tipo de especificação.

Um circuito complicado é concebido como uma coleção de sub-circuitos mais simples, cada um dos quais pode ser projetado individualmente.

Na qualidade de sub-circuitos podem ser usados blocos de uso frequente tais como descodificadores, codificadores e multiplexers.

Diagramas de blocos

Um diagrama de blocos contém informação sobre entradas, saídas, sub-circuitos (blocos) e ligações internas entre estes num sistema.



Um barramento é a coleção de 2 ou mais sinais relacionados.

Blocos AOI e OAI

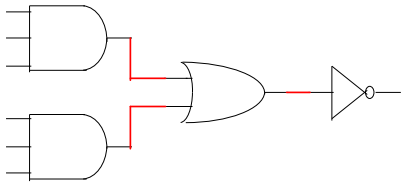
Conceptualmente, um bloco AOI (*And-Or-Invert*) é um circuito de 3 níveis em que portas AND estão no 1º nível, uma porta OR – no 2º nível e um inversor no 3º nível.

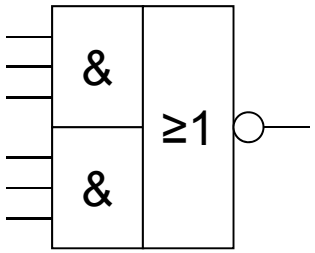
Conceptualmente, um bloco OAI (*Or-And-Invert*) é um circuito de 3 níveis em que portas OR estão no 1º nível, uma porta AND – no 2º nível e um inversor no 3º nível.

n-input m-stack
n – número de entradas nas portas do 1º nível
m – número de portas do 1º nível

Exemplo:

3-input 2-stack AOI:



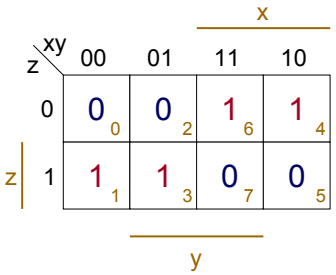


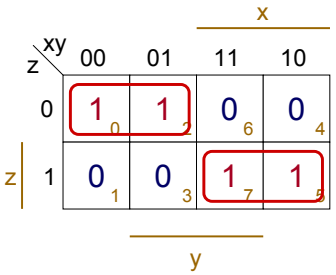
Implementação de funções com blocos AOI e OAI

Para implementar uma função lógica com blocos AOI deve-se determinar o complemento da função na forma de soma de produtos mínima.

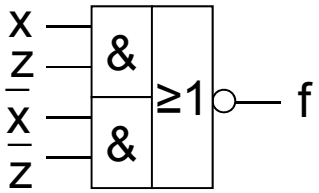
Exemplo:

Implementar função f(x,y,z) com blocos AOI: $f(x,y,z) = x \cdot \overline{z} + \overline{x} \cdot y \cdot z + \overline{x} \cdot \overline{y} \cdot z$





$\overline{f(x,y,z)} = x \cdot z + \overline{x} \cdot \overline{z}$



2-input 2-stack AOI

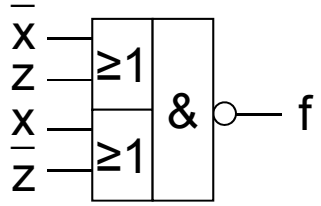
Para implementar uma função lógica com blocos OAI deve-se determinar o complemento da função na forma de produto de somas mínimo.

Exemplo:

Implementar função f(x,y,z) com blocos OAI:

$\overline{f(x,y,z)} = (\overline{x} + z) \cdot (x + \overline{z})$

2-input 2-stack OAI:



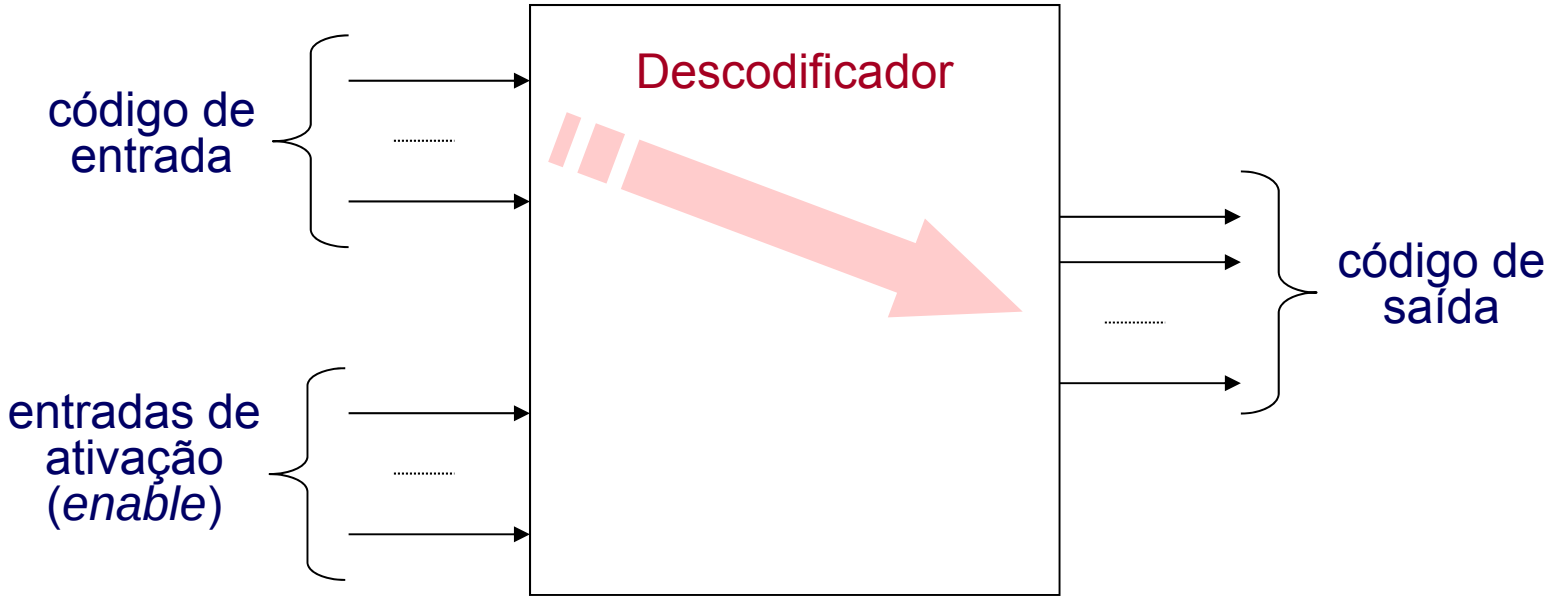
ieeta

Descodificadores

Um descodificador é um circuito lógico que tem múltiplas entradas e múltiplas saídas e converte entradas codificadas em saídas codificadas.

Os códigos de entrada têm normalmente menos bits que os códigos de saída.

O mapeamento entre códigos é 1-1, i.e. cada código de entrada produz um diferente código de saída.



ieeta

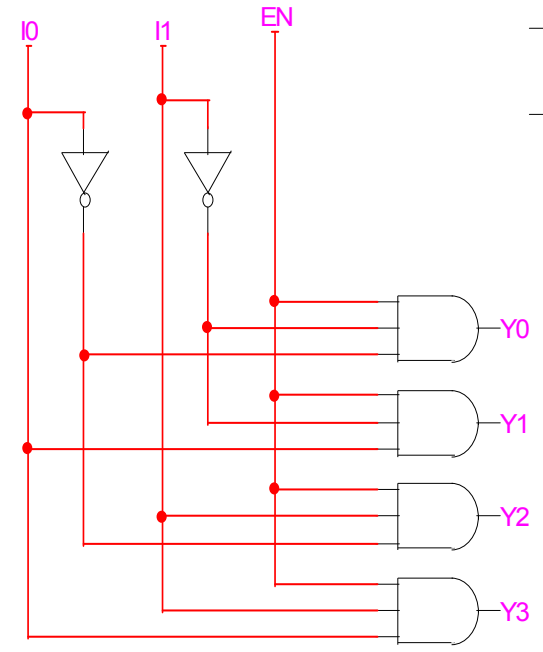
Descodificadores binários

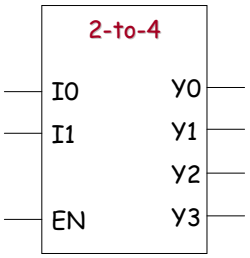
Um descodificador binário n-to-2^n tem *n* entradas (com as quais pode-se representar uma das 2^n combinações) e 2^n saídas, das quais apenas uma pode estar ativa.

Exemplo: Descodificador binário 2-to-4:

EN	I1	I0	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0





$$Y0 = EN \cdot \overline{I1} \cdot \overline{I0}$$

$$Y1 = EN \cdot \overline{I1} \cdot I0$$

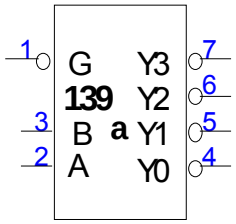
$$Y2 = EN \cdot I1 \cdot \overline{I0}$$

$$Y3 = EN \cdot I1 \cdot I0$$

ieeta

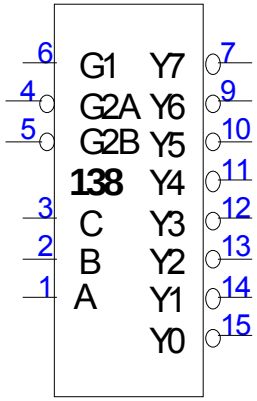
Descodificadores comerciais

74x139 - descodificador 2-to-4 dual



G_L	B	A	Y3_L	Y2_L	Y1_L	Y0_L
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

74x138 - descodificador 3-to-8



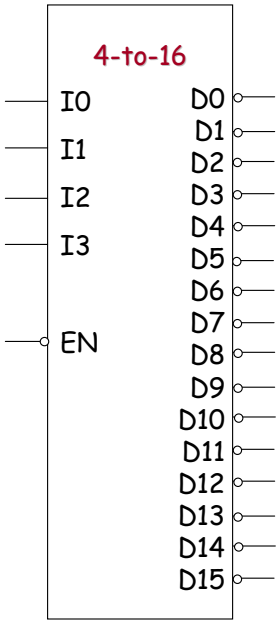
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1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

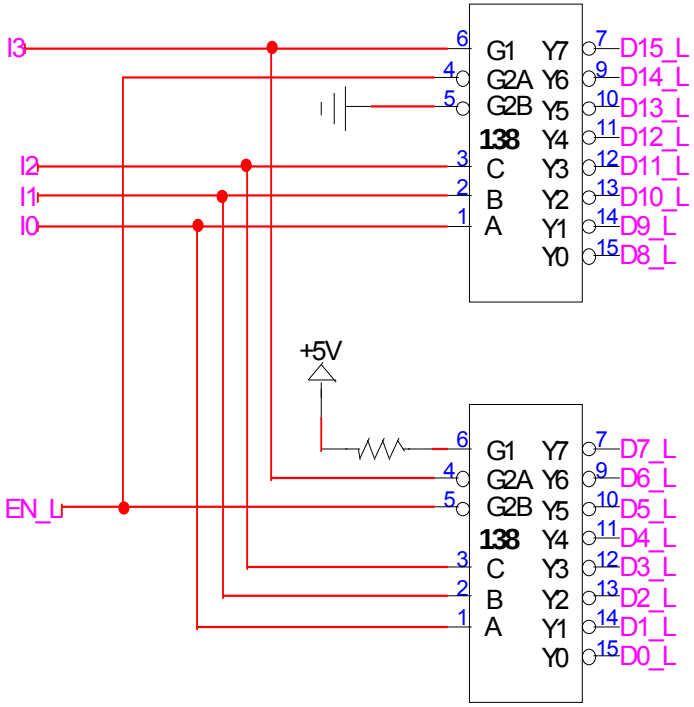
Descodificadores em cascata

Para descodificar palavras de código maiores pode-se usar vários descodificadores interligados.

Exemplo:

Construir um descodificador binário 4-to-16 com descodificadores 3-to-8:



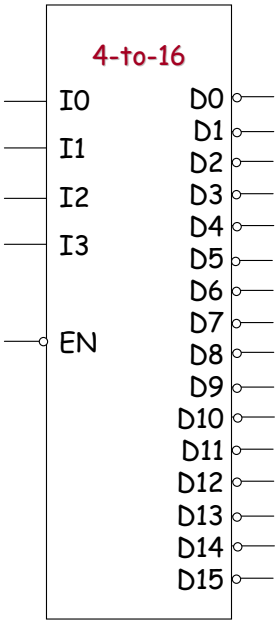


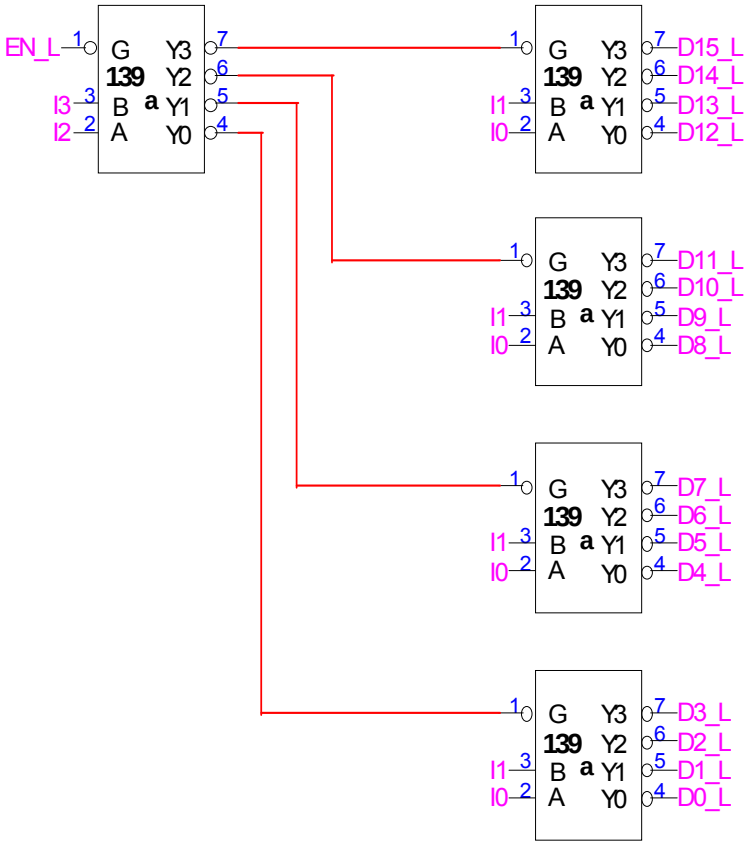
ieeta

Descodificadores em cascata (cont.)

Exemplo:

Construir um descodificador binário 4-to-16 com descodificadores 2-to-4:





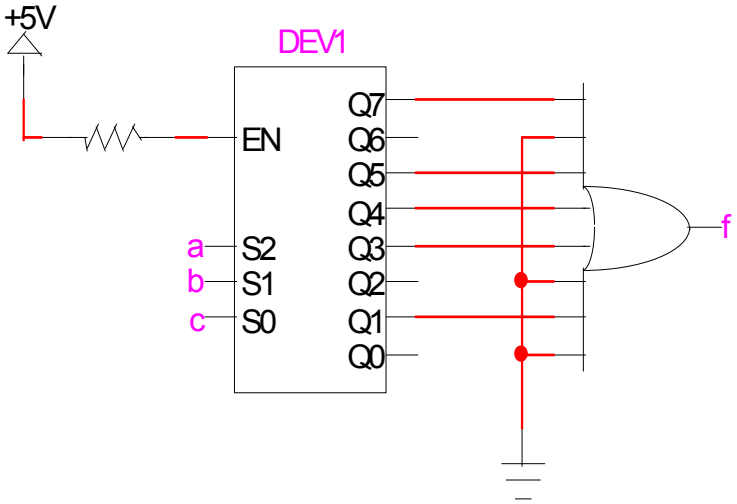
ieeta

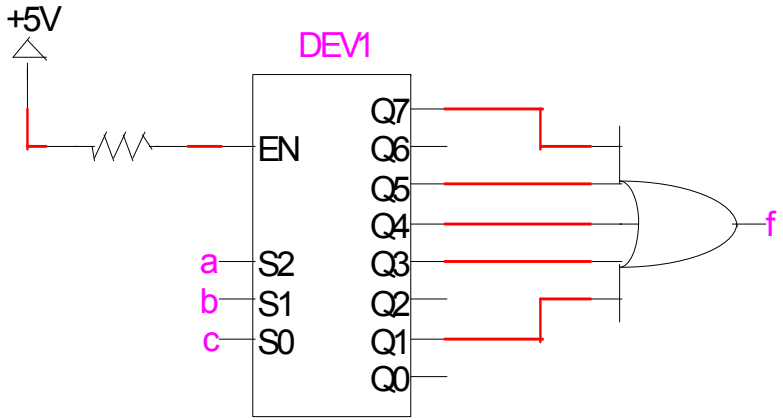
Descodificadores e funções lógicas

Com um descodificador binário n-to-2^n e uma porta OR-2^n pode-se implementar qualquer função lógica de *n* variáveis.

Para tal, deve-se expressar a função na 1ª forma canónica.

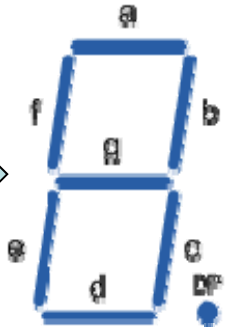
Exemplo: $f(a,b,c) = a \cdot \overline{b} + c$ $\qquad$ $f(a,b,c) = \sum m(1,3,4,5,7)$

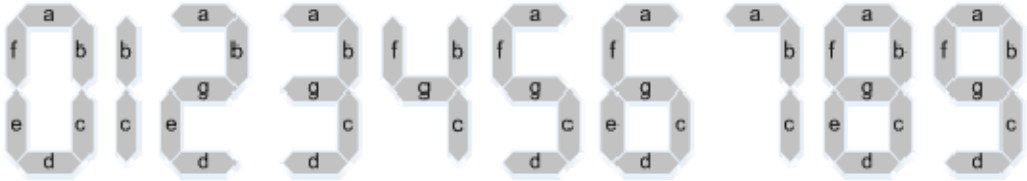


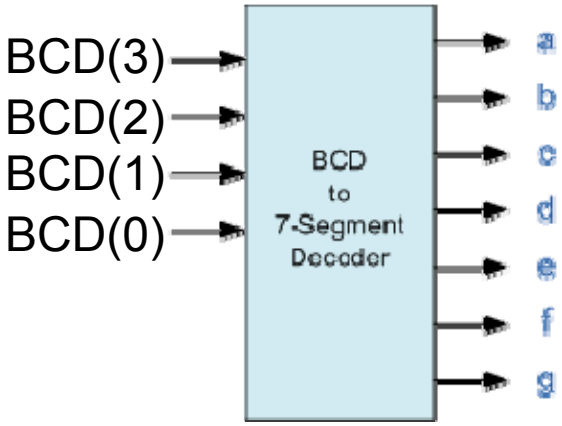


Descodificador BCD – *display* de 7 segmentos

0,1,…,9 → ?







BCD	número	segmentos individuais						
		a	b	c	d	e	f	g
0000	0	1	1	1	1	1	1	
0001	1		1	1				
0010	2	1	1		1	1		1
0011	3	1	1	1	1			1
0100	4		1	1			1	1
0101	5	1		1	1		1	1
0110	6	1		1	1	1	1	1
0111	7	1	1	1				
1000	8	1	1	1	1	1	1	1
1001	9	1	1	1	1		1	1
101x	x	x	x	x	x	x	x	x
11xx	x	x	x	x	x	x	x	x

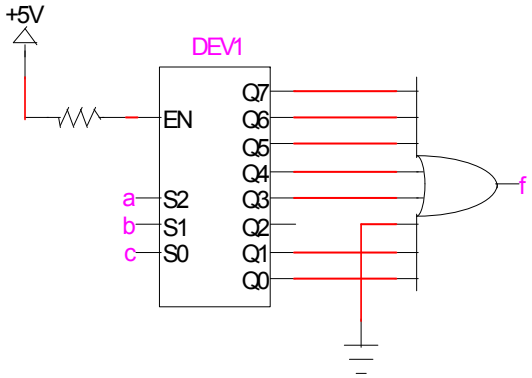
ieeta

Exercícios

Implemente a função seguinte usando um bloco AOI e um bloco OAI.
Explicite as dimensões mínimas destes blocos.

$$f(a,b,c) = a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot \overline{c} + a \cdot \overline{b} \cdot c$$

Implemente a função $f(a,b,c) = a + \overline{b} + c$ com um descodificador 3-to-8 e portas OR

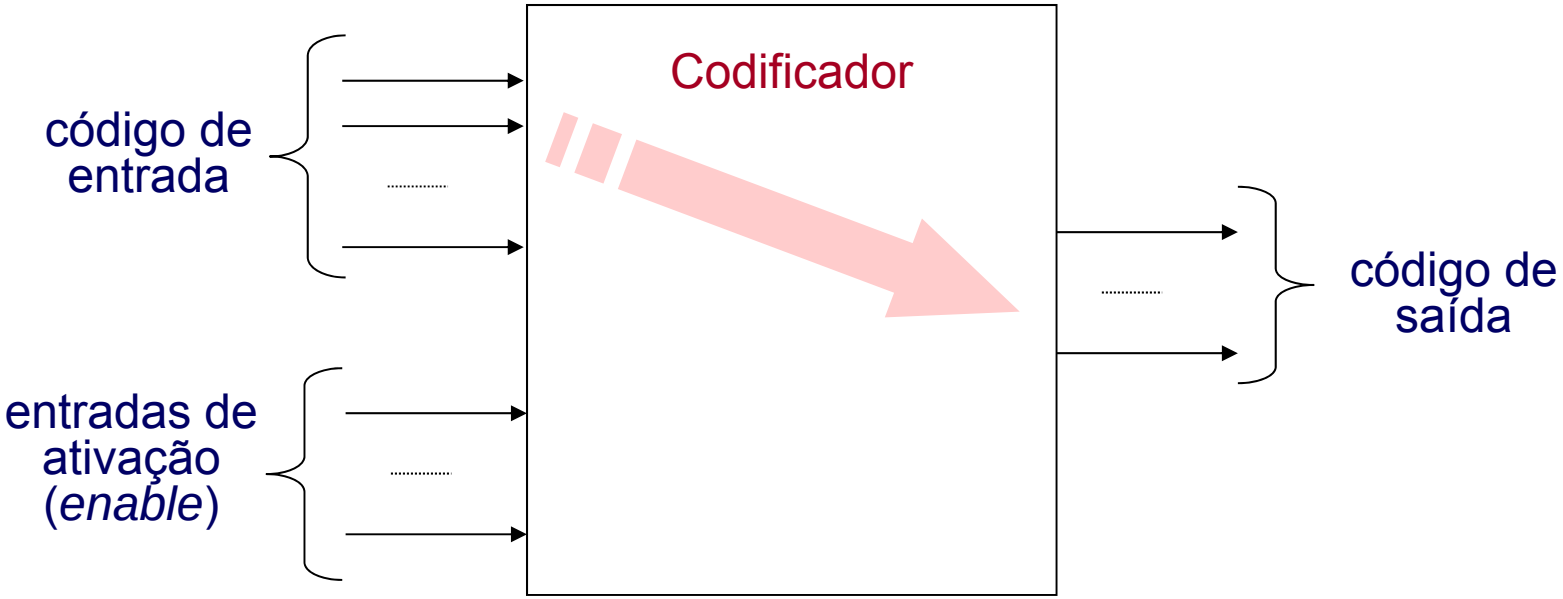


Codificadores

Um codificador é um circuito lógico que tem múltiplas entradas e múltiplas saídas e converte entradas codificadas em saídas codificadas.

Os códigos de entrada têm normalmente mais bits que os códigos de saída.

O mapeamento entre códigos é 1-1, i.e. cada código de entrada produz um diferente código de saída.



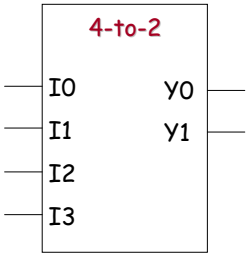
ieeta

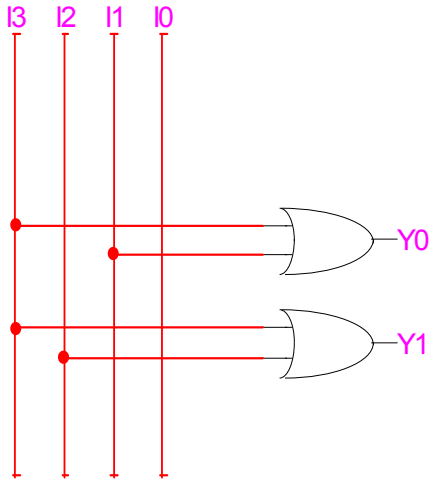
Codificadores binários

Um codificador binário 2^n-to-n tem 2^n entradas (das quais apenas uma pode estar ativa) e n saídas, que indicam o código binário natural da entrada ativa.

Um codificador binário 2^n-to-n pode ser construído com n portas OR com 2^{n-1} entradas cada.

Exemplo: Codificador binário 4-to-2:





$$Y0 = I3 + I1$$

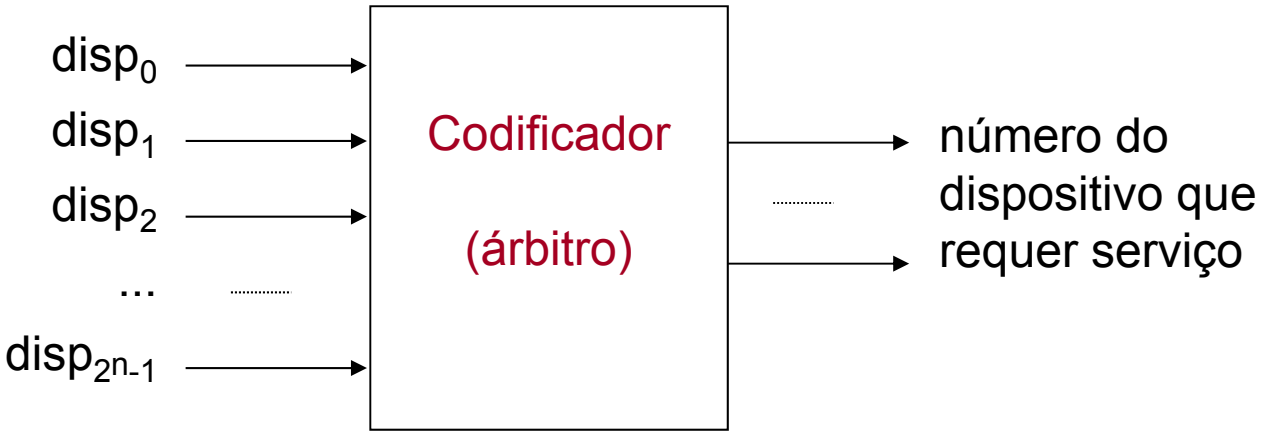
$$Y1 = I3 + I2$$

I3	I2	I1	I0	Y1	Y0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

ieeta

Codificadores de prioridade

Um codificador binário 2^n-to-n só funciona corretamente se no máximo 1 entrada está ativa.



No caso do codificador 4-to-2, se ativarmos I2 e I1 na saída será gerado código "11" identificando incorretamente a entrada I3.

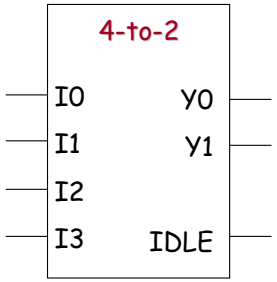
$$Y0 = I3 + I1$$

$$Y1 = I3 + I2$$

A solução é atribuir prioridade às entradas tal que se aparecerem múltiplos pedidos de serviço será processado apenas aquele que tem a maior prioridade.

Codificadores de prioridade (cont.)

Exemplo: Codificador binário 4-to-2 com prioridade:

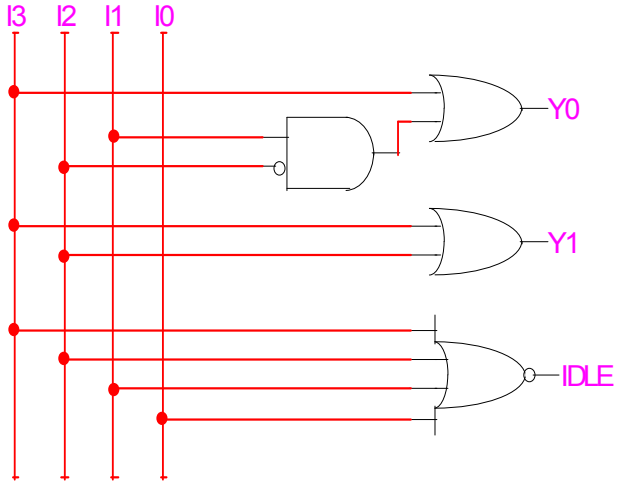


$$Y0 = I3 + \overline{I2} \cdot I1$$

$$Y1 = I3 + I2$$

$$IDLE = \overline{I3} \cdot \overline{I2} \cdot \overline{I1} \cdot \overline{I0}$$

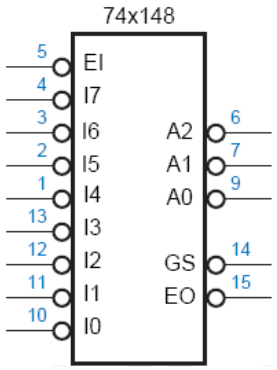
I3	I2	I1	I0	Y1	Y0	IDLE
1	x	x	x	1	1	0
0	1	x	x	1	0	0
0	0	1	x	0	1	0
0	0	0	1	0	0	0
0	0	0	0	0	0	1



ieeta

Codificadores comerciais

74x148 - codificador 8-to-3 com prioridade



EI_L	I7_L	I6_L	I5_L	I4_L	I3_L	I2_L	I1_L	I0_L	A2_L	A1_L	A0_L	EO_L	GS_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	0	x	x	x	x	x	x	x	0	0	0	1	0
0	1	0	x	x	x	x	x	x	0	0	1	1	0
0	1	1	0	x	x	x	x	x	0	1	0	1	0
0	1	1	1	0	x	x	x	x	0	1	1	1	0
0	1	1	1	1	0	x	x	x	1	0	0	1	0
0	1	1	1	1	1	0	x	x	1	0	1	1	0
0	1	1	1	1	1	1	0	x	1	1	0	1	0
0	1	1	1	1	1	1	1	0	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	0	1

GS_L – indica que o dispositivo está ativo (EI_L='0') e uma das entradas está ativa (a '0')

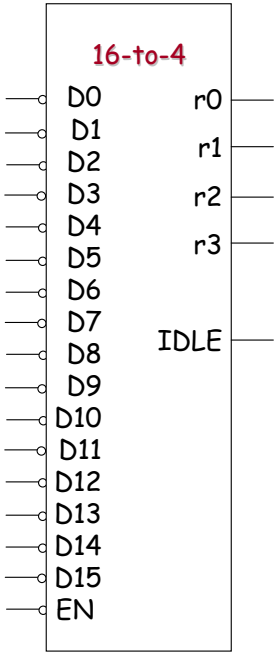
EO_L (*enable output*) – indica que o dispositivo está ativo (EI_L='0') mas nenhuma das entradas está ativa (a '0') – serve para construir codificadores em cascata

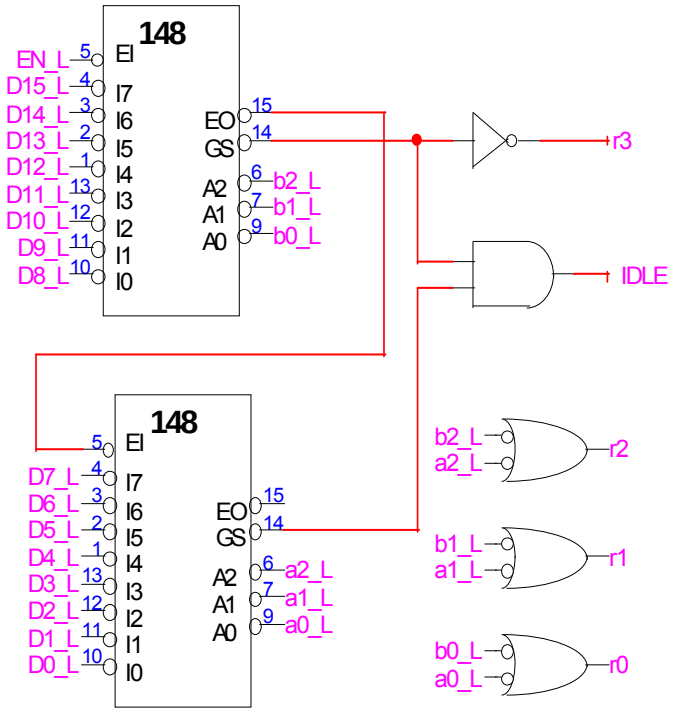
Codificadores em cascata

Para codificar palavras de código maiores pode-se usar vários codificadores interligados.

Exemplo:

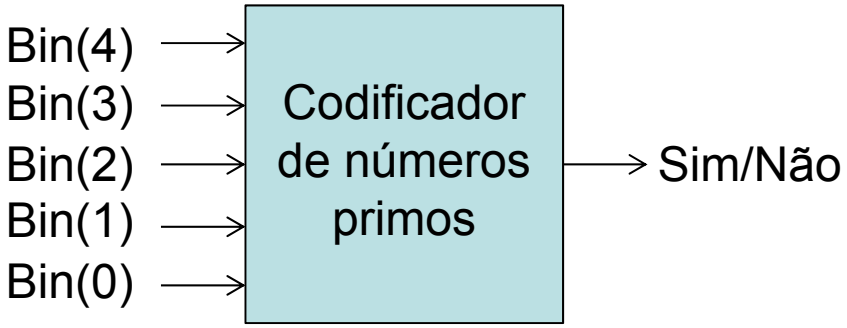
Construir um codificador binário 16-to-4 com codificadores 8-to-3 e portas lógicas adicionais:





Codificador de números primos

1,2,…,31 ⟹ é primo?

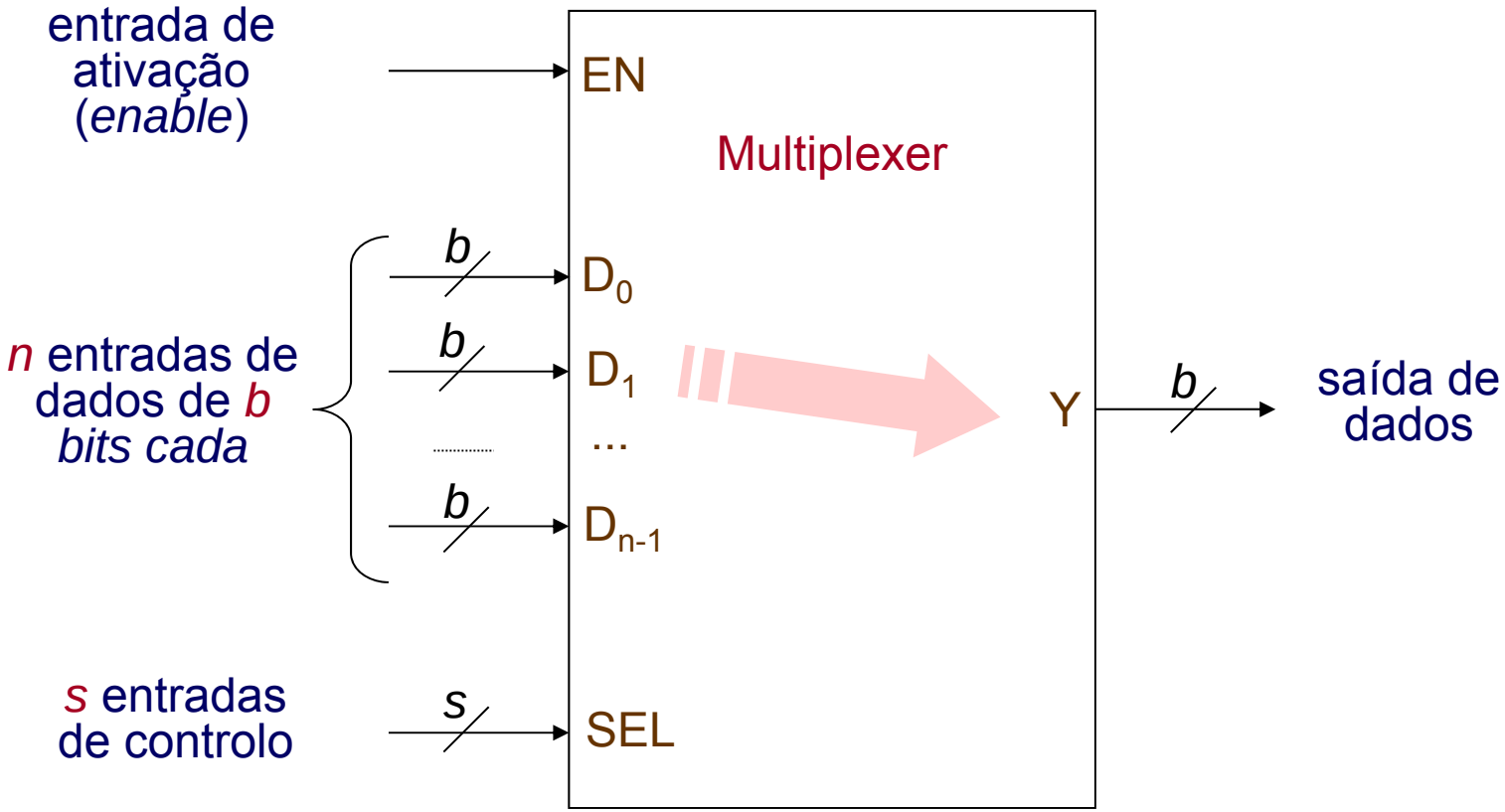


Bin	número	Sim
00000	0	0
00001	1	0
00010	2	1
00011	3	1
00100	4	0
00101	5	1
00110	6	0
00111	7	1
01000	8	0
01001	9	0
…	…	…
11111	31	1

ieeta

Multiplexers

Um multiplexer encaminha dados de uma das *n* fontes para a única saída. A fonte é selecionada com base em $s=\lceil \log_2(n) \rceil$ entradas de controlo.



Multiplexers (cont.)

Equação de saída de um multiplexer n:1:

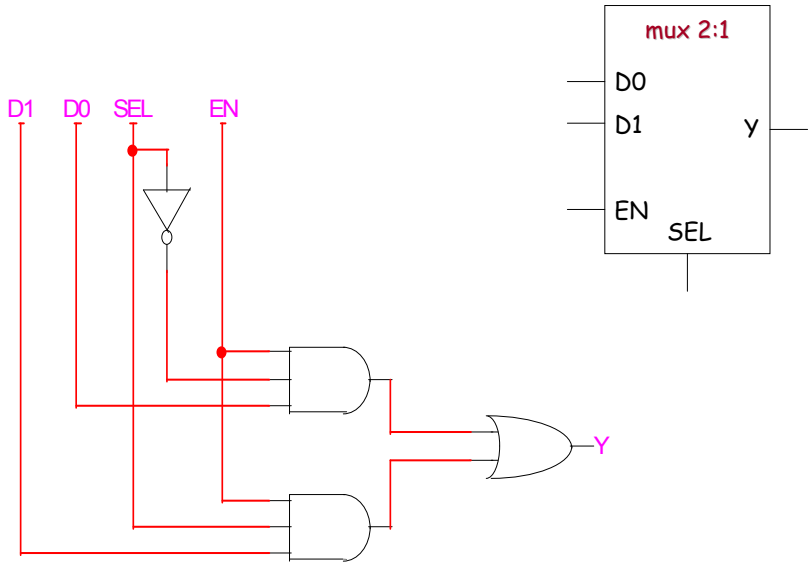
(m_j – termo mínimo j das entradas de controlo)

$$Y = \sum_{j=0}^{n-1} EN \cdot m_j \cdot D_j$$

Exemplo: Multiplexer 2:1:

EN	SEL	D1	D0	Y
0	x	x	x	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

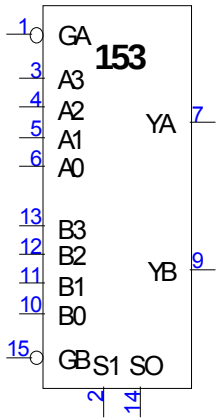
EN	SEL	Y
0	x	0
1	0	D0
1	1	D1



$$Y = EN \cdot \overline{SEL} \cdot D0 + EN \cdot SEL \cdot D1$$

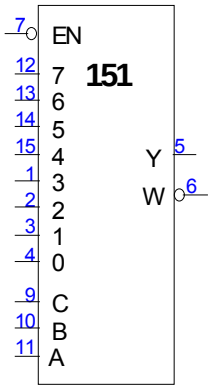
Multiplexers comerciais

74x153 - multiplexer 4:1 dual



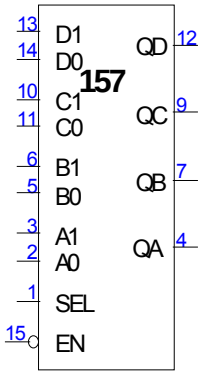
GA_L	GB_L	S1	S0	YA	YB
0	0	0	0	A0	B0
0	0	0	1	A1	B1
0	0	1	0	A2	B2
0	0	1	1	A3	B3
0	1	0	0	A0	0
0	1	0	1	A1	0
0	1	1	0	A2	0
0	1	1	1	A3	0
1	0	0	0	0	B0
1	0	0	1	0	B1
1	0	1	0	0	B2
1	0	1	1	0	B3
1	1	x	x	0	0

74x151 - multiplexer 8:1 de 1 bit



EN_L	C	B	A	Y	W_L
1	x	x	x	0	1
0	0	0	0	D0	$\overline{D0}$
0	0	0	1	D1	$\overline{D1}$
0	0	1	0	D2	$\overline{D2}$
0	0	1	1	D3	$\overline{D3}$
0	1	0	0	D4	$\overline{D4}$
0	1	0	1	D5	$\overline{D5}$
0	1	1	0	D6	$\overline{D6}$
0	1	1	1	D7	$\overline{D7}$

74x157 - multiplexer 2:1 de 4 bits



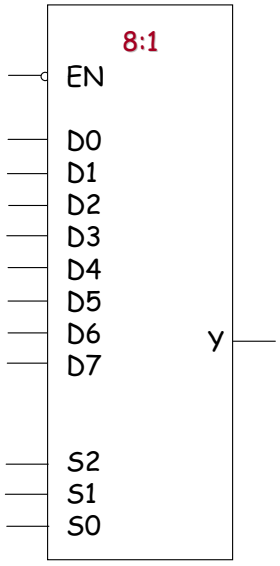
EN_L	SEL	QD	QC	QB	QA
1	x	0	0	0	0
0	0	D0	C0	B0	A0
0	1	D1	C1	B1	A1

Multiplexers em cascata

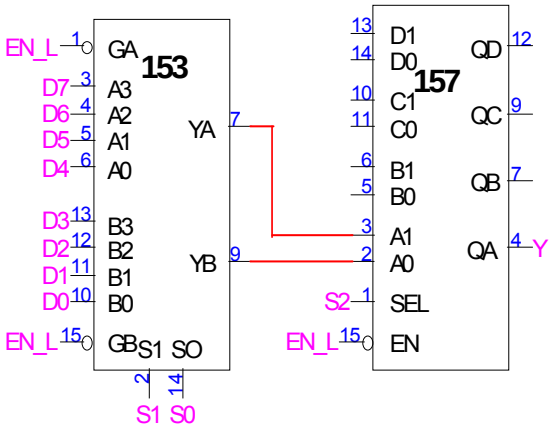
Multiplexers maiores podem ser implementados colocando multiplexers mais pequenos em cascata.

Exemplo:

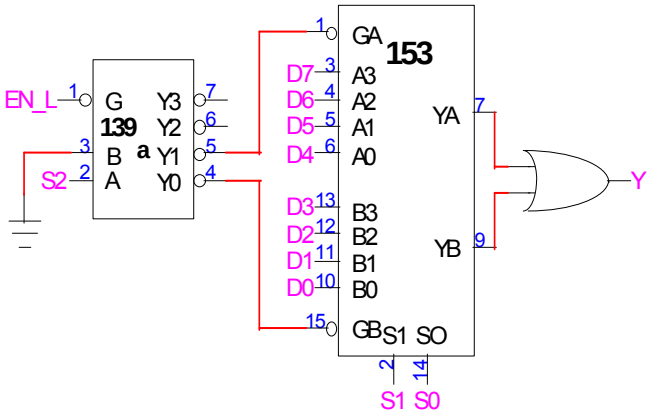
Construir um multiplexer 8:1:



com mux 4:1 e mux 2:1



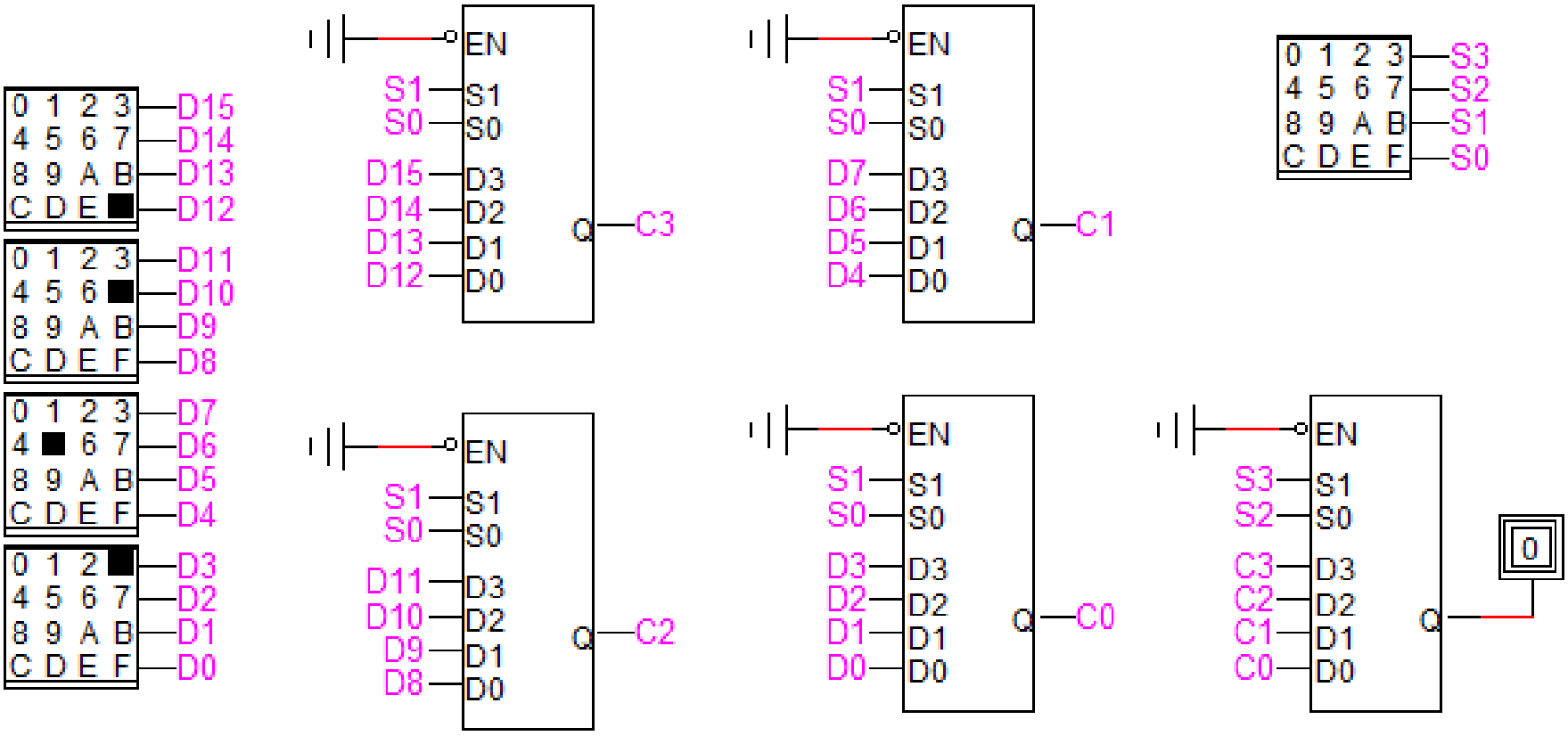
com mux 4:1 e lógica adicional



Multiplexers em cascata

Exemplo:

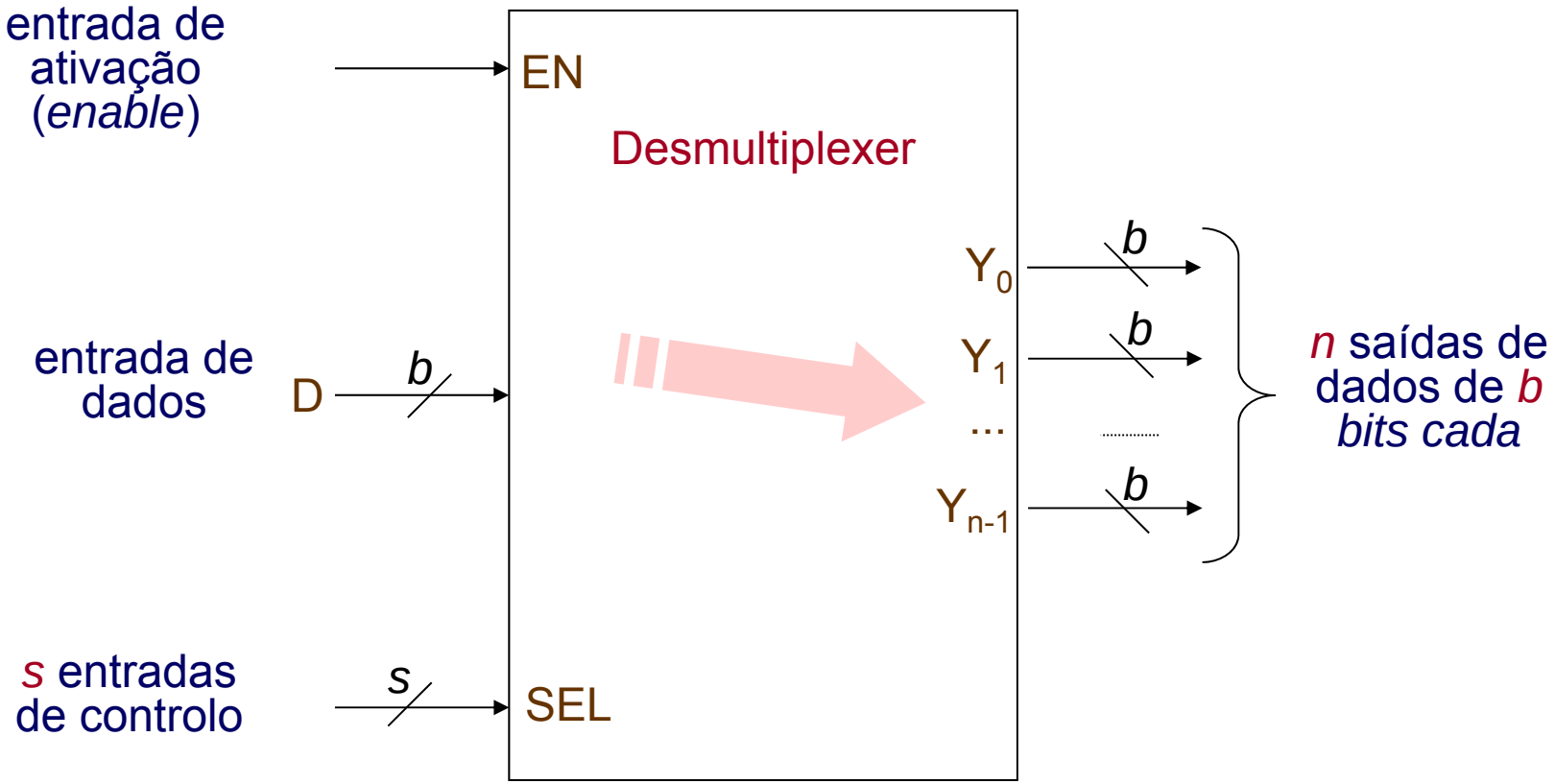
Construir um multiplexer 16:1 com multiplexers 4:1.



ieeta

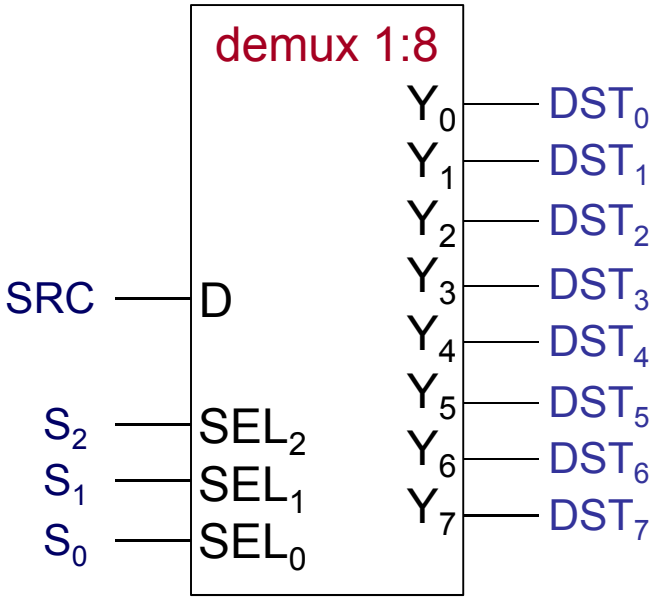
Desmultiplexers

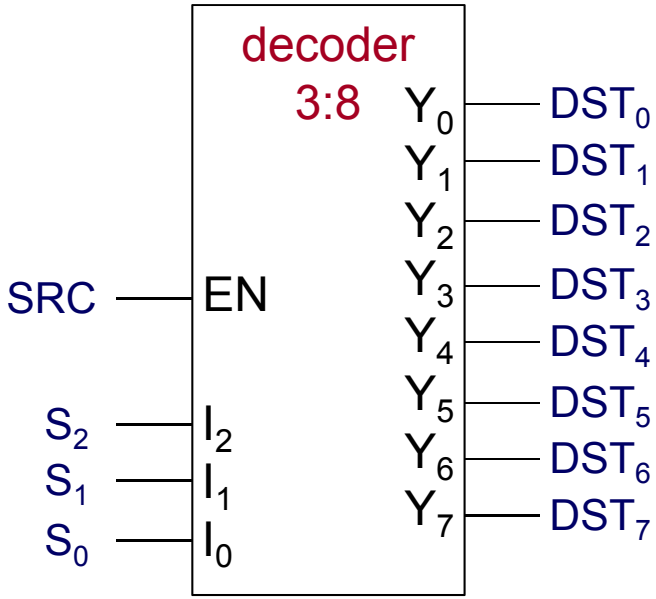
Um desmultiplexer encaminha uma única entrada de dados para uma das *n* saídas. A saída é selecionada com base em $s = \lceil \log_2(n) \rceil$ entradas de controlo.

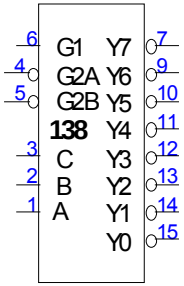


Desmultiplexers (cont.)

Um descodificador binário com entradas de habilitação *(enable)* pode ser usado como circuito de desmultiplexagem.



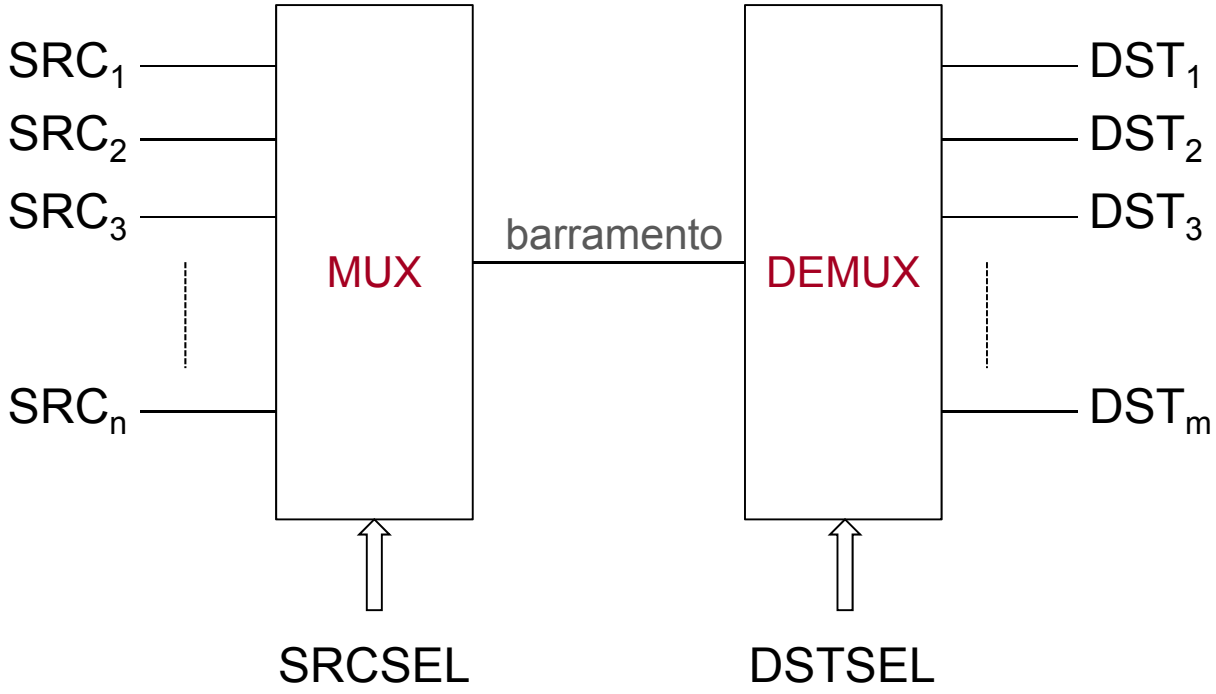




74x138 – descodificador/desmultiplexer

Uso comum de multiplexers e desmultiplexers

Multiplexers e desmultiplexers desempenham funções importantes em circuitos de comutação.



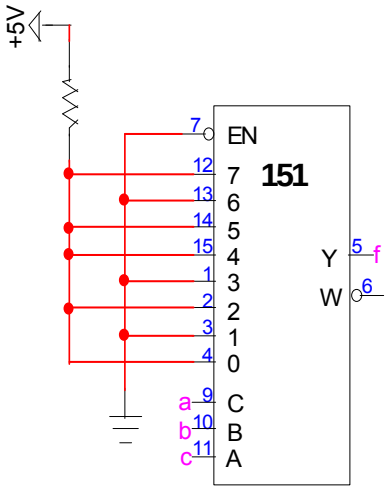
Multiplexers e funções lógicas

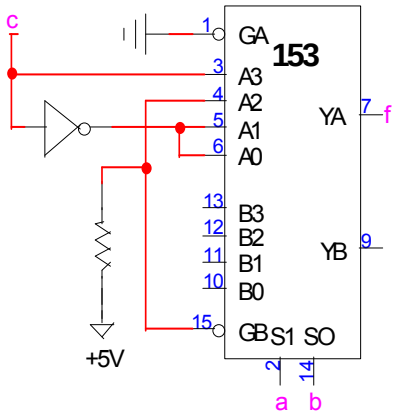
Com um multiplexer 2^n:1 e constantes 0 e 1 pode-se implementar qualquer função lógica de *n* variáveis.

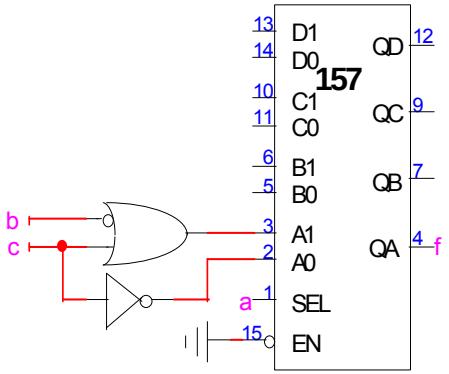
Com um multiplexer 2^{n-1}:1, constantes 0 e 1 e portas NOT pode-se implementar qualquer função lógica de *n* variáveis.

Exemplos: $f(a,b,c) = a \cdot \overline{b} + a \cdot c + \overline{a} \cdot \overline{c}$

a	b	c	f	f	f
0	0	0	1	$\overline{c}$	$\overline{c}$
0	0	1	0		
0	1	0	1	$\overline{c}$	
0	1	1	0		
1	0	0	1	1	$\overline{b}+c$
1	0	1	1		
1	1	0	0	c	
1	1	1	1		

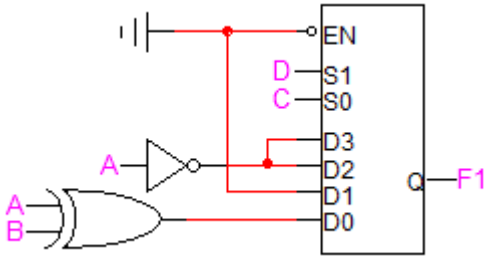


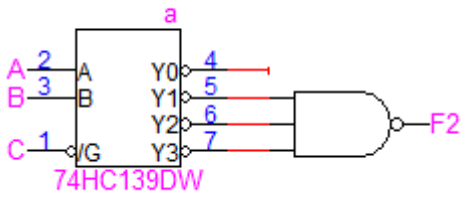


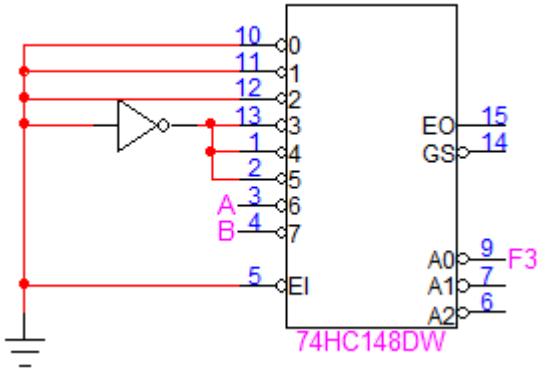


Exercícios

Analise os circuitos seguintes e determine a expressão mais simples para as funções em termos do operador NAND.







Exercícios (cont.)

Projete um codificador 10:4 em cujas entradas podem aparecer palavras de código *1-out-of-10* e cujas saídas representam o código BCD da entrada ativada.

Projete um circuito que converte palavras de código de Gray de 3 bits em código binário natural.

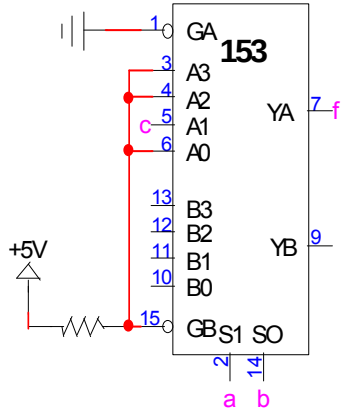
Exercícios (cont.)

Um *barrel shifter* de 4 bits é um circuito lógico combinatório que tem 4 entradas de dados, 4 saídas de dados e 2 entradas de controlo. A palavra de saída é igual à palavra de entrada "rodada" tantas posições quantas especificadas pelas entradas de controlo. Por exemplo se a palavra de entrada for ABCD e as entradas de controlo forem 10_2, a palavra de saída será CDAB. Projete um *barrel shifter* de 4 bits usando blocos lógicos que conhece.

Implemente a função $f(a,b,c) = a + \overline{b} + c$ com:

a) um multiplexer 4:1 e constantes 0 e 1;

b) um multiplexer 2:1 e lógica adicional.



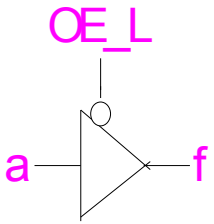
Buffers 3-state

Para além de estados elétricos LOW e HIGH existe um terceiro estado - alta impedância (Z) que representa uma resistência infinita.

Uma saída com 3 estados possíveis chama-se saída *three-state* (ou *tri-state*).

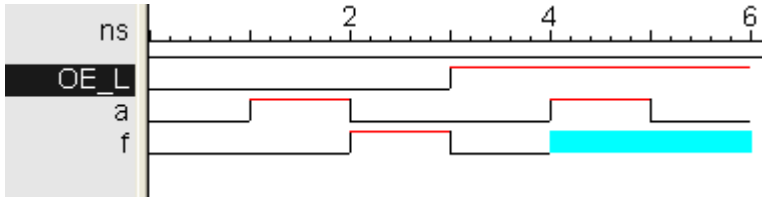
Dispositivos *3-state* têm uma entrada adicional de ativação *(enable)*.

Buffer 3-state



OE_L	a	f
0	0	0
0	1	1
1	x	Z

OE_L	f
0	a
1	Z



O estado Z permite ligar mais que um dispositivo *3-state* ao mesmo ponto, desde que apenas um tenha a sua saída ativa em cada instante.

Para tal dispositivos *3-state* são projetados a maneira de entrarem no estado Z mais rapidamente do que saírem do mesmo:

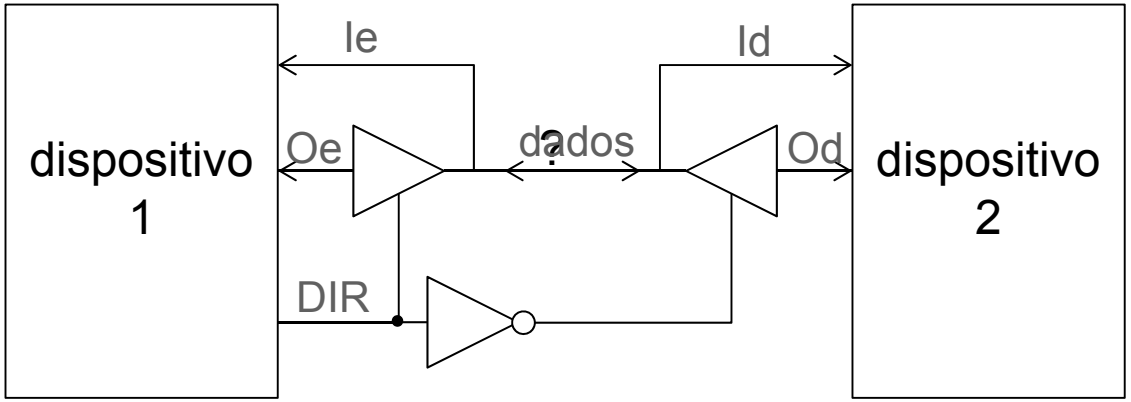
$t_{pLZ} < t_{pZL}$ $t_{pLZ} < t_{pZH}$

$t_{pHZ} < t_{pZH}$ $t_{pHZ} < t_{pZL}$

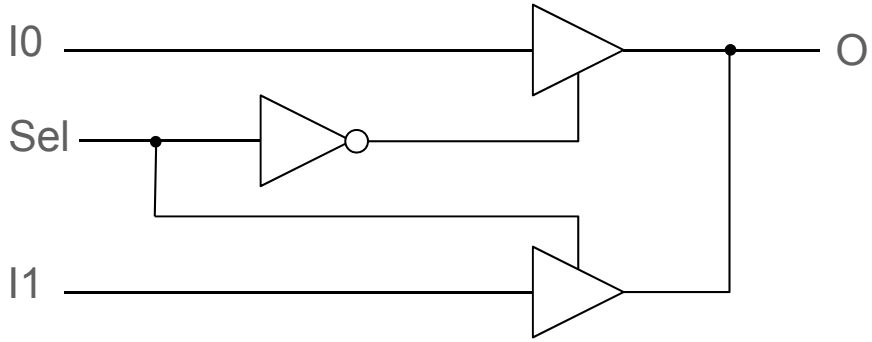
ieeta

Aplicação dos buffers 3-state

- Linhas bidirecionais:



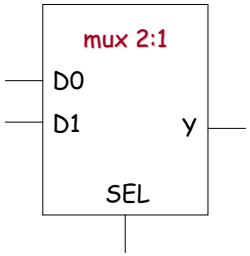
- Ligação de vários sinais a uma linha:



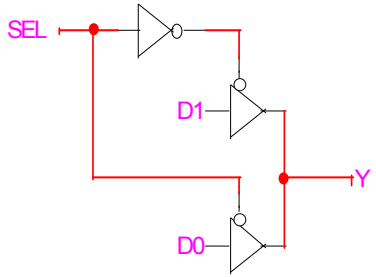
Multiplexagem com buffers 3-state

Com portas 3-state pode-se construir multiplexers “baratos”.

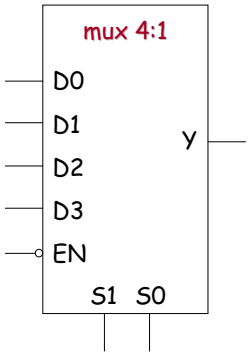
Exemplos: Multiplexer 2:1:



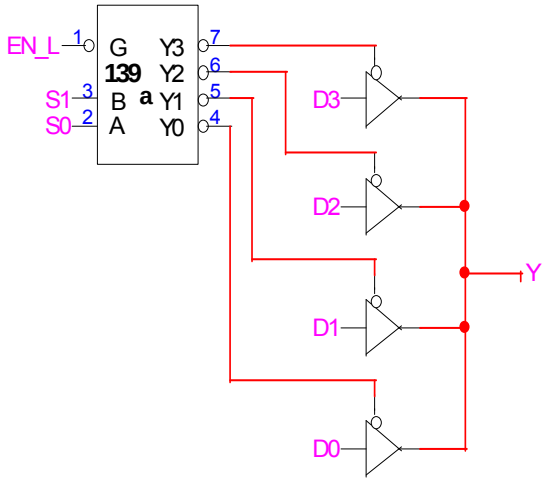
SEL	Y
0	D0
1	D1



Multiplexer 4:1:



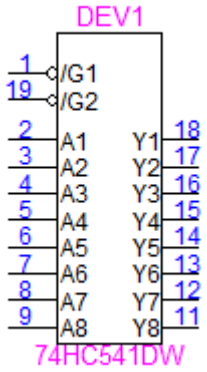
EN_L	S1	S0	Y
1	x	x	0
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3



ieeta

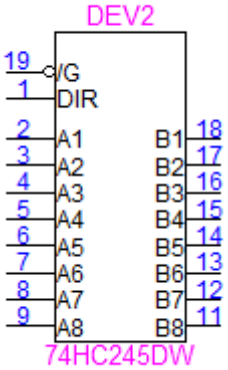
Dispositivos 3-state comerciais

74x541 – buffer 3-state octal



G1_L	G2_L	Y
1	x	Z
x	1	Z
0	0	A

74x245 – transmissor/recetor de 8 bits

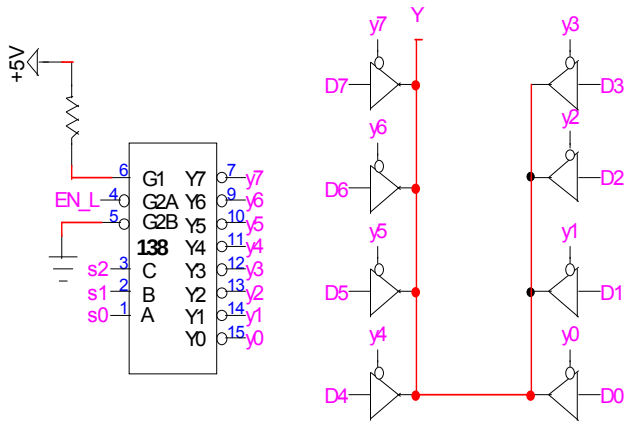


G_L	DIR	A	B
1	0	Z	x
1	1	x	Z
0	0	B	B
0	1	A	A

ieeta

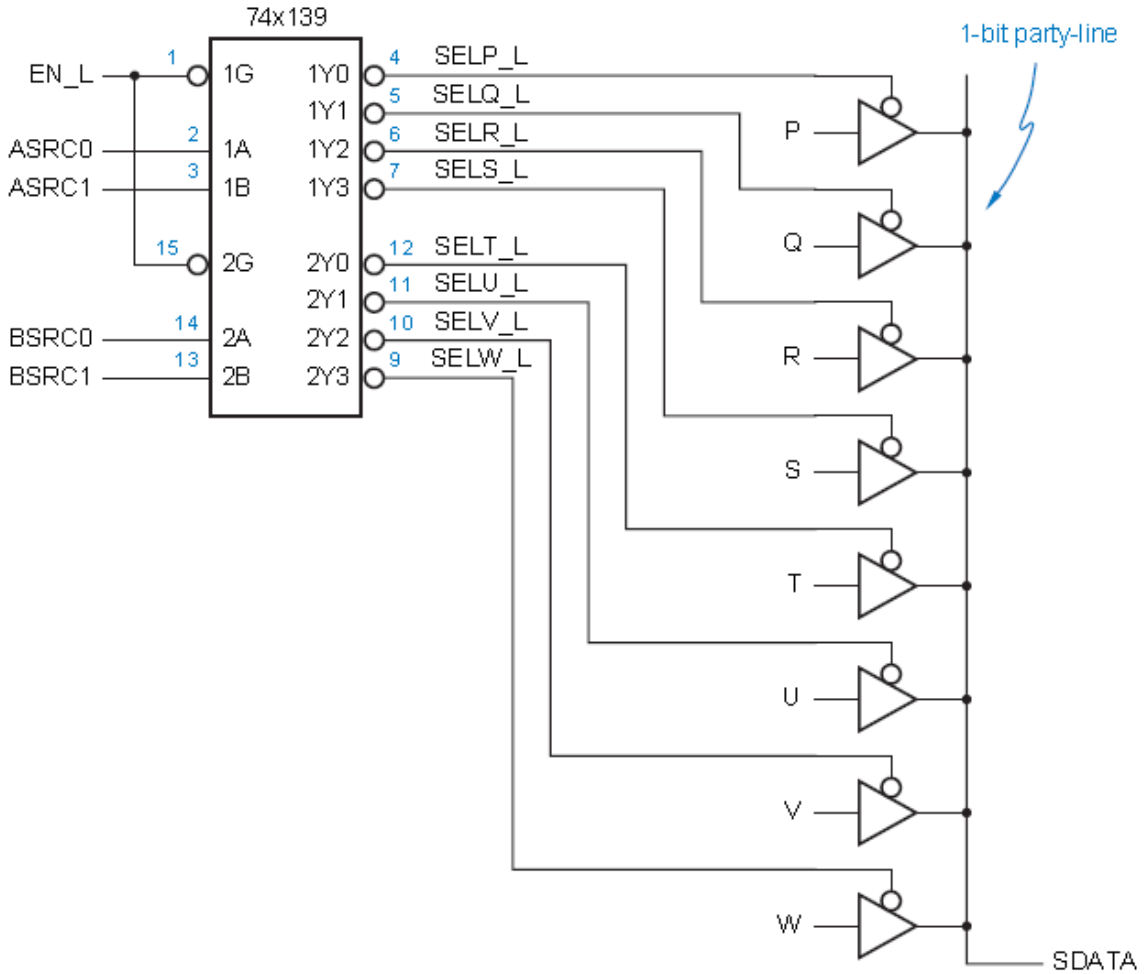
Exercícios

Construa um multiplexer 8:1 com um descodificador e *buffers* 3-state.



Exercícios (cont.)

Que problema tem o circuito da figura?



Exercícios (cont.)

Qual é a função do circuito seguinte?