Fundamentos de Arquitetura de Computadores

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília

UnB Gama
O novo endereço da tecnologia

Sumário

Módulo 03

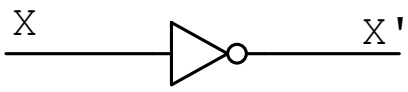
- Portas Lógicas
- Decodificadores
- Multiplexadores

UnB Gama
O novo endereço da tecnologia

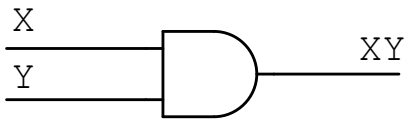
Portas Lógicas

Normalmente **representamos** funções booleanas utilizando um *diagrama lógico*, isto é, um desenho esquemático de portas simbólicas (*gate symbols*) conectados por linhas.

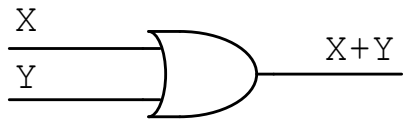
As três portas lógicas básicas são, não surpreendetemente, as portas **e**, **ou** e **não** (também chamada de *porta inversora*). Os símbolos dessas três portas são:



X	$\overline{X}$
0	1
1	0



X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1



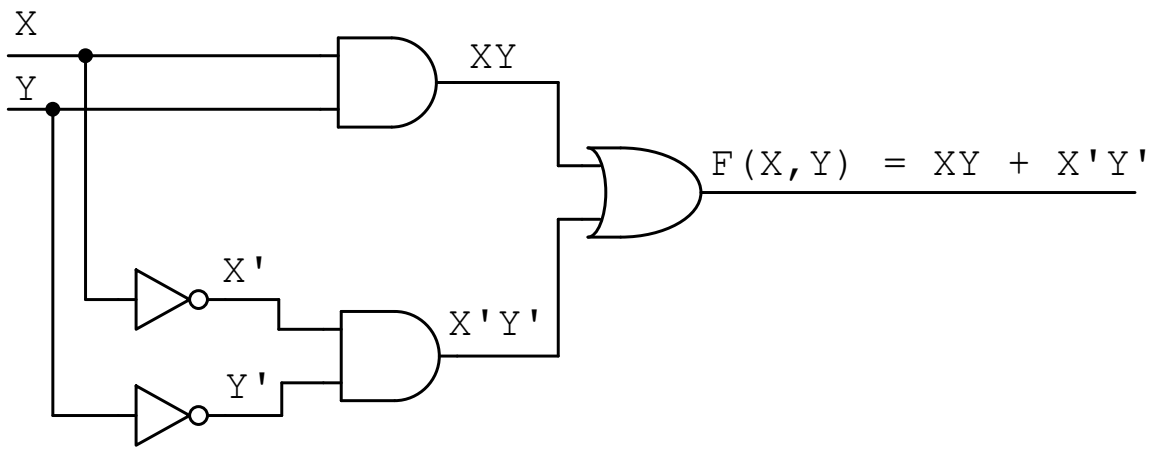
X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

Usando apenas esses três símbolos, podemos desenhar diagramas lógicos para *qualquer* função booleana!

Prova: Há 2^{2^2} funções de duas variáveis. Para provar o que foi afirmado acima, basta implementar as outras 13 operações (funções) lógicas de duas variáveis partindo dessas três funções lógicas.

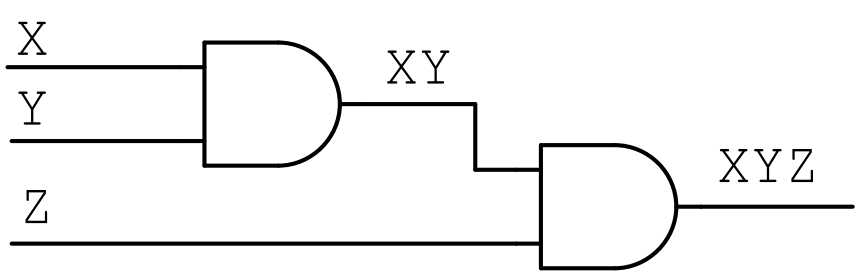
Desenhando diagramas lógicos de funções booleanas:

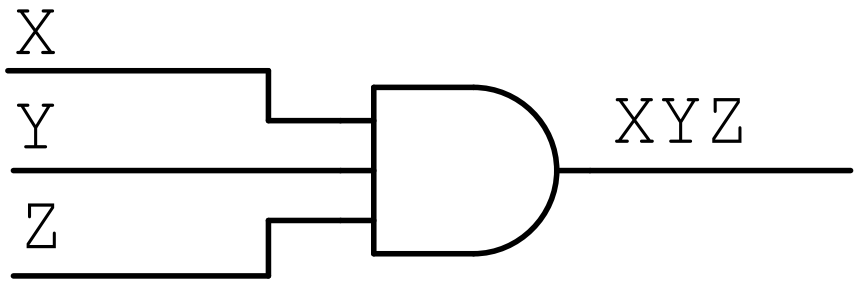
$$\mathbf{F}(\mathbf{X},\mathbf{Y}) = \mathbf{X}\cdot\mathbf{Y} + \overline{\mathbf{X}}\cdot\overline{\mathbf{Y}}$$



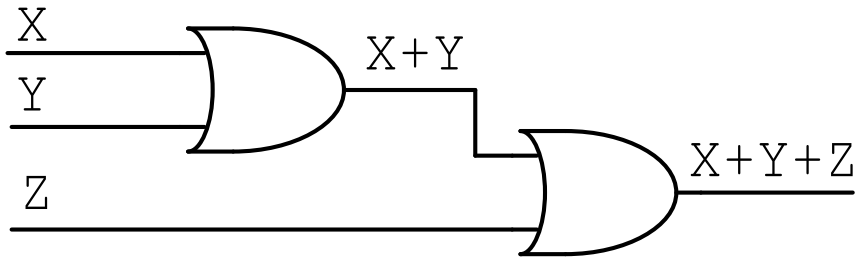
Note que podemos utilizar portas lógicas **e** e **ou** de múltiplas entradas.

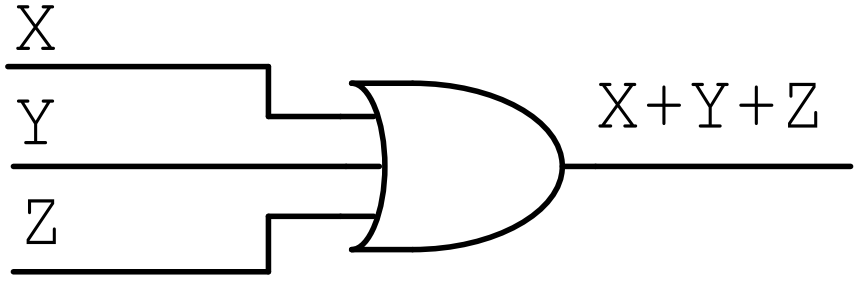
$$\mathbf{F}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \mathbf{X} \cdot \mathbf{Y} \cdot \mathbf{Z}$$





$$\mathbf{F}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \mathbf{X} + \mathbf{Y} + \mathbf{Z}$$



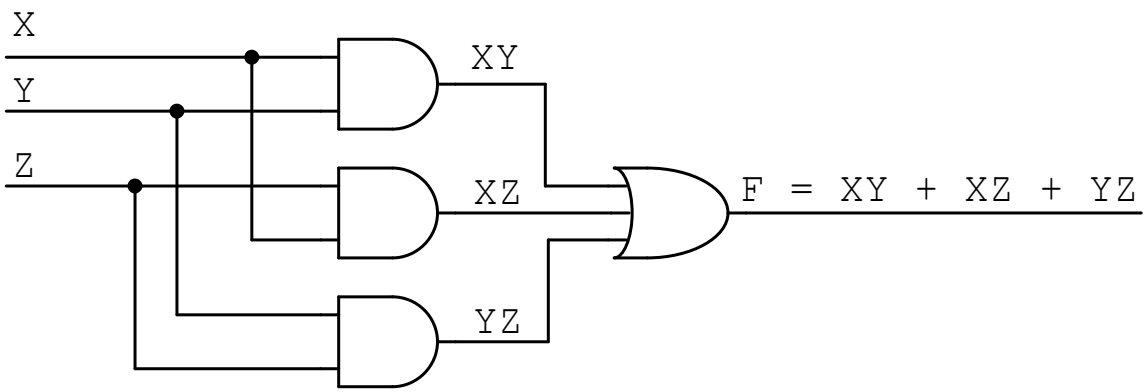


Desenhando diagramas lógicos de funções booleanas:

$$\mathbf{F}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \mathbf{X} \cdot \mathbf{Y} + \mathbf{X} \cdot \mathbf{Z} + \mathbf{Y} \cdot \mathbf{Z}$$

Desenhando diagramas lógicos de funções booleanas:

$$\mathbf{F}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \mathbf{X} \cdot \mathbf{Y} + \mathbf{X} \cdot \mathbf{Z} + \mathbf{Y} \cdot \mathbf{Z}$$



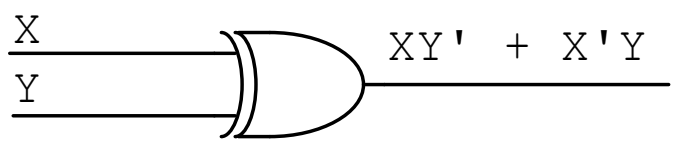
Portas Ou-Exclusivo (XOR)

As portas lógicas básicas (**e**, **ou** e **inversora**) são suficientes para realizar qualquer função booleana. Apesar disso, outras portas importantes tem seu próprio símbolo, como a porta **ou-exclusivo**.

- **XOR**: a porta XOR (**ou-exclusivo**, ou *exclusive-or*), é definida como uma porta em que a saída é $\mathbf{1}$ se e somente se as entradas são diferentes, e $\mathbf{0}$ se forem iguais.
- O símbolo da porta **XOR** em álgebra booleana é $\oplus$.

$\mathbf{X}$	$\mathbf{Y}$	$\mathbf{X \oplus Y}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}$

$$\mathbf{F}(\mathbf{X},\mathbf{Y}) = \mathbf{X}\cdot\overline{\mathbf{Y}} + \overline{\mathbf{X}}\cdot\mathbf{Y}$$
$$= \mathbf{X}\oplus\mathbf{Y}$$



Identidades com a Porta XOR

comutatividade	$x \oplus y = y \oplus x$
associatividade	$(x \oplus y) \oplus z = x \oplus (y \oplus z)$

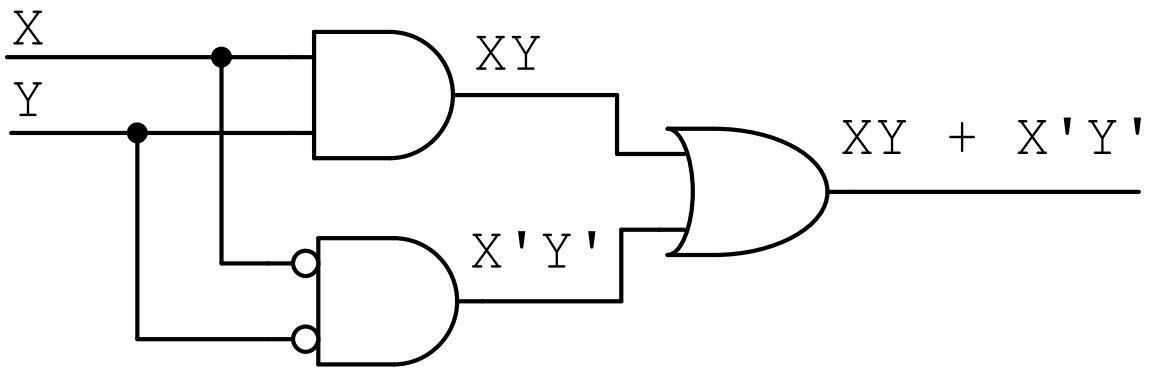
- Em aritmética módulo-2, a porta **XOR** funciona exatamente como a **adição** módulo-2!
- A porta **XOR** é bastante utilizada em geradores de bits de paridade.
- **ATENÇÃO**: Diferente das portas **e** e **ou**, a porta **XOR** *apenas é definida para duas entradas!*
- Como funcionaria uma porta **XOR** de múltiplas entradas?

UnB Gama
O novo endereço da tecnologia

A Notação de Inversão

É comum utilizarmos o complemento de várias variáveis na implementação de funções booleanas. Por isso, existe uma notação mais simples para a inversão de uma variável qualquer: a **bolha**.

$$\mathbf{F}(\mathbf{X}, \mathbf{Y}) = \mathbf{X} \cdot \mathbf{Y} + \overline{\mathbf{X}} \cdot \overline{\mathbf{Y}}$$

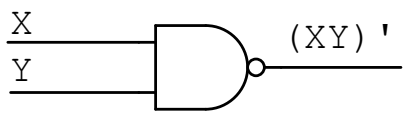


Portas NAND e Portas NOR

Com a notação de inversão, são definidas as três últimas **portas lógicas básicas**:

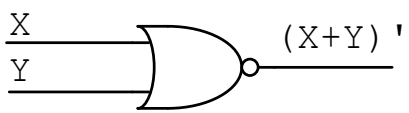
- **NAND**: a porta NAND (**não-e**), é definida como uma porta **e** com saída invertida.
- **NOR**: a porta NOR (**não-ou**), é definida como uma porta **ou** com saída invertida.
- **NXOR**: a porta NXOR (**não-ou-exclusivo**), é definida como uma porta **XOR** com saída invertida.

NAND



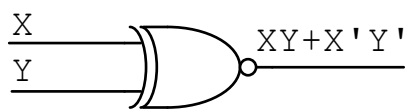
X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR



X	Y	$\overline{X + Y}$
0	0	1
0	1	0
1	0	0
1	1	0

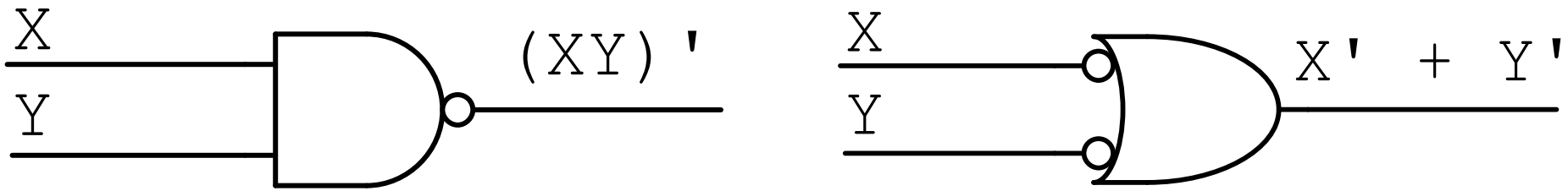
NXOR



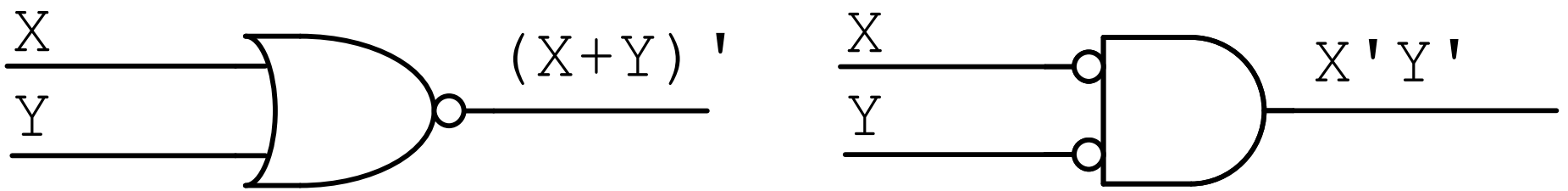
X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

Lembrando o **Teorema de De Morgan**:

$$\overline{\mathbf{X} \cdot \mathbf{Y}} = \overline{\mathbf{X}} + \overline{\mathbf{Y}}$$



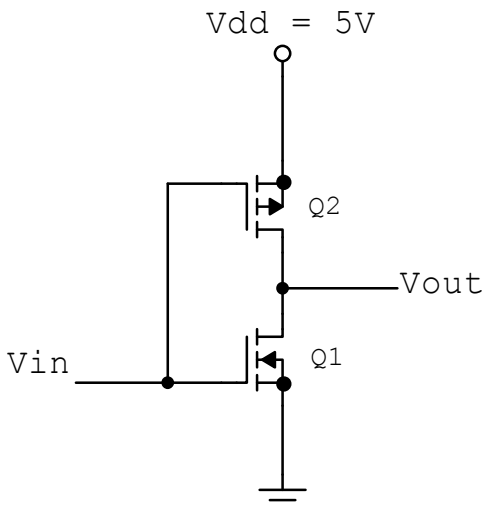
$$\overline{\mathbf{X} + \mathbf{Y}} = \overline{\mathbf{X}} \cdot \overline{\mathbf{Y}}$$



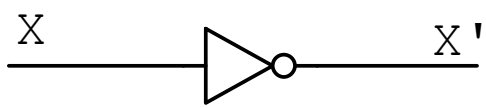
Implementação Física: Circuitos Eletrônicos

As portas lógicas que vimos podem ser implementadas (ou realizadas) utilizando transistores. Transistores são dispositivos de 3 terminais que agem como uma resistência controlada por tensão. A tensão aplicada no terminal de entrada (V_{in}) controla a resistência entre os demais terminais.

De maneira simplificada, se a tensão de entrada é 0, a resistência entre os demais terminais é muito alta e o transistor é dito *desligado* (*off*). Caso contrário, se a tensão de entrada é alta ($5V$), a resistência entre os demais terminais é muito baixa e o transistor é dito *ligado* (*on*).



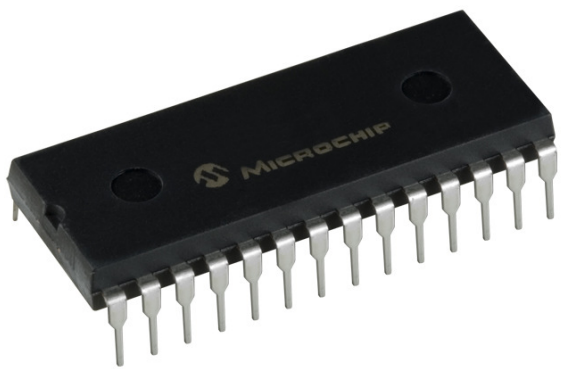
V_{in}	Q_1	Q_2	V_{out}
low	*off*	*on*	high
high	*on*	*off*	low

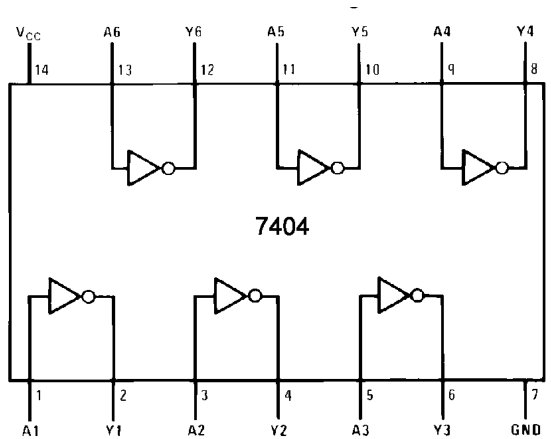


Circuitos Integrados

Portas lógicas podem ser encontradas na forma de *circuitos integrados* (CIs).

- No mesmo CI, em geral temos várias portas lógicas.
- O CI deve ser "alimentado" (isto é, deve-se fornecer a tensão $V+$ e uma referência gnd): sim, ele deve ser ligado e, ao funcionar, consome energia!
- As características específicas das portas de um CI, como o *fan-in*, *fan-out*, atraso, etc.. são fornecidas pelo fabricante no *datasheet* do CI.





Famílias Lógicas

Existem diversas famílias lógicas de CIs. As mais comuns:

- CMOS - *Complementary Metal Oxide Semiconductor*.
- ECL - *Emitter Coupled Logic*.
- TTL - *Transistor-Transistor Logic*.

UnB Gama
O novo endereço da tecnologia

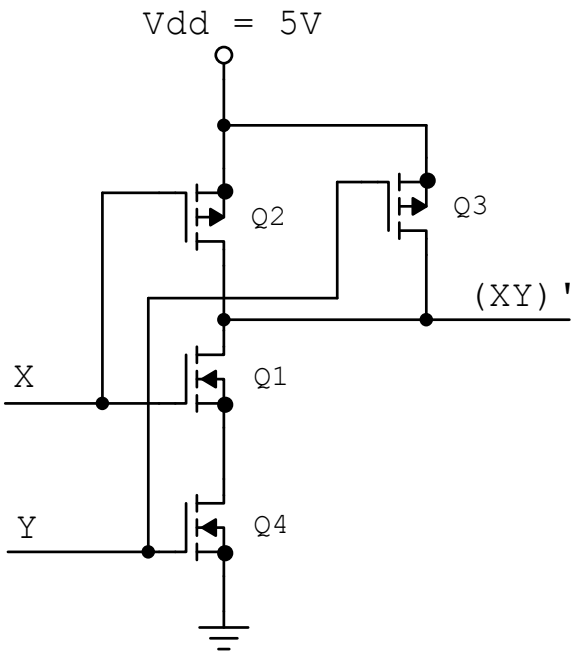
Implementação de Circuitos com Portas NAND

É mais fácil implementar portas NANDs e NORs com transistores do que implementar portas ANDs e ORs!

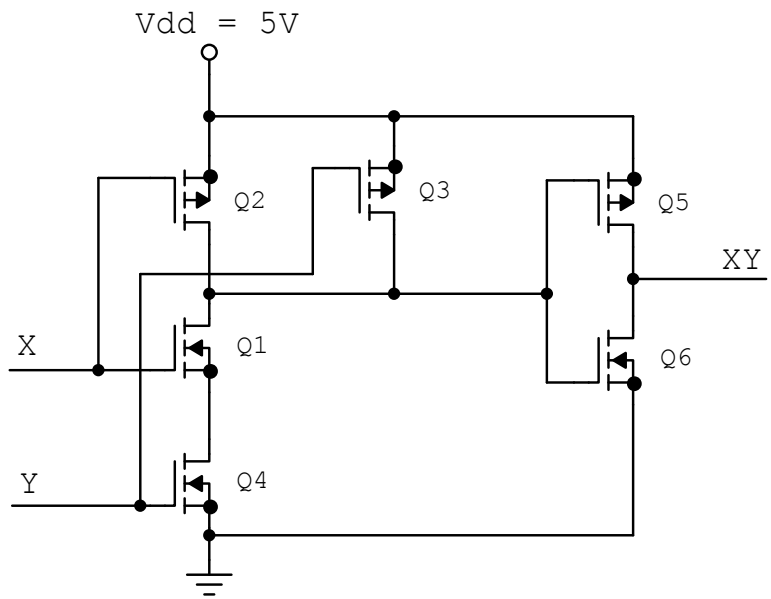
Por essa razão, é importante aprender a implementar circuitos utilizando apenas essas portas.

Olha o **Teorema de De Morgan** mostrando seu poder novamente!

NAND



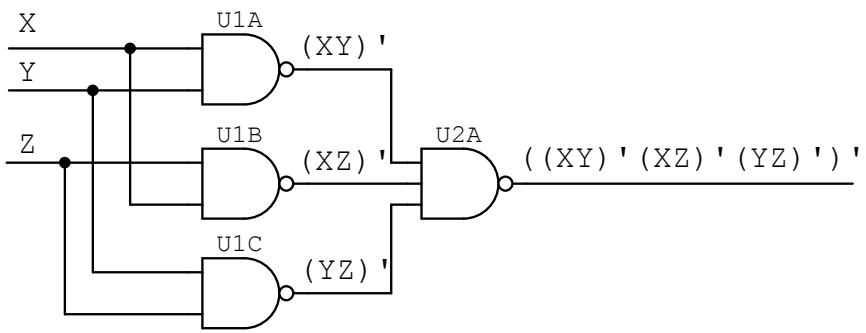
AND

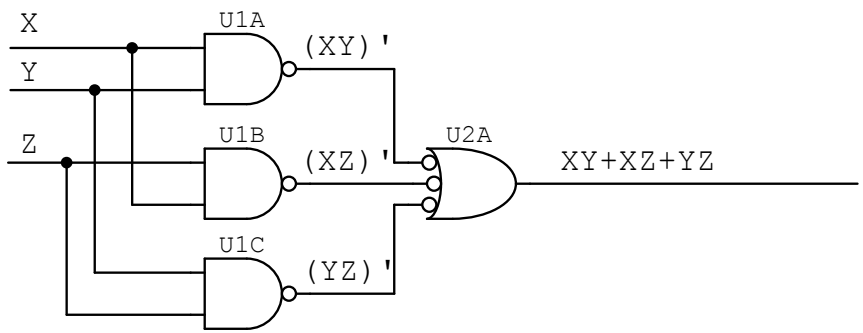


Implementação de Circuitos com Portas NAND

Para isso, é importante saber implementar funções booleanas com portas NAND. Para isso, utilizamos o Teorema de De Morgan.

$$\begin{aligned}\mathbf{F}(\mathbf{X},\mathbf{Y},\mathbf{Z}) &= \mathbf{X}\cdot\mathbf{Y}+\mathbf{X}\cdot\mathbf{Z}+\mathbf{Y}\cdot\mathbf{Z}\\ &= \overline{\overline{\mathbf{X}\cdot\mathbf{Y}+\mathbf{X}\cdot\mathbf{Z}+\mathbf{Y}\cdot\mathbf{Z}}}\\ &= \overline{\overline{\mathbf{X}\cdot\mathbf{Y}}\cdot\overline{\mathbf{X}\cdot\mathbf{Z}}\cdot\overline{\mathbf{Y}\cdot\mathbf{Z}}}\end{aligned}$$





Decodificadores

Um *decodificador* é um circuito de **múltiplas entradas** e **múltiplas saídas** que converte códigos de entrada em códigos de saída, onde os dois códigos são diferentes.

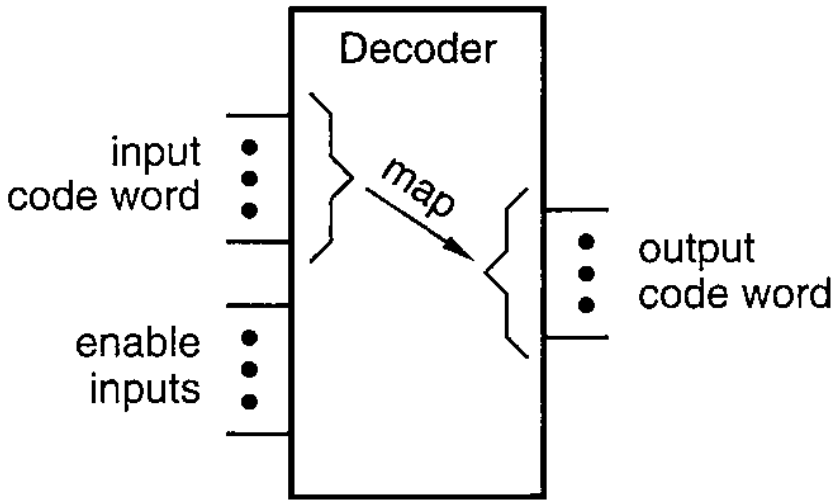
Em geral, o código de entrada tem **menos bits** que o código de saída, e existe um mapeamento *um-pra-um* do código de entrada para o código de saída.

O código de saída mais utilizado é o 1-de-m, que usa m bits para m códigos e, para cada código, apenas um dos m bits está ativo. Por exemplo:

Ativo em alto	Ativo em baixo
1000	**0111**
0100	**1011**
0010	**1101**
0001	**1110**

UnB Gama
O novo endereço da tecnologia

Decodificadores



Para que o decodificador funcione, *todas* as suas entradas de *enable* (habilitadoras) devem estar ativadas.

Caso qualquer uma delas não esteja ativa, o decodificador mapeia qualquer entrada para um código "desativado".

Decodificadores

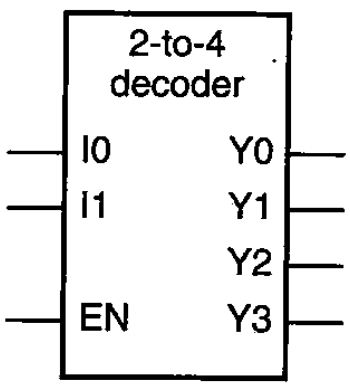
O decodificador mais comum é o n-*para*-2^n, ou *decodificador binário*, que mapeia um código binário de n bits para um de 2^n códigos diferentes (em geral, números decimais de 0 a $2^n - 1$).

Às vezes, utilizamos menos que os 2^n valores possíveis. Por exemplo, é comum utilizar um decodificador BCD, que tem 4 entradas e 10 saídas (os números BCD, de 0 a 9).

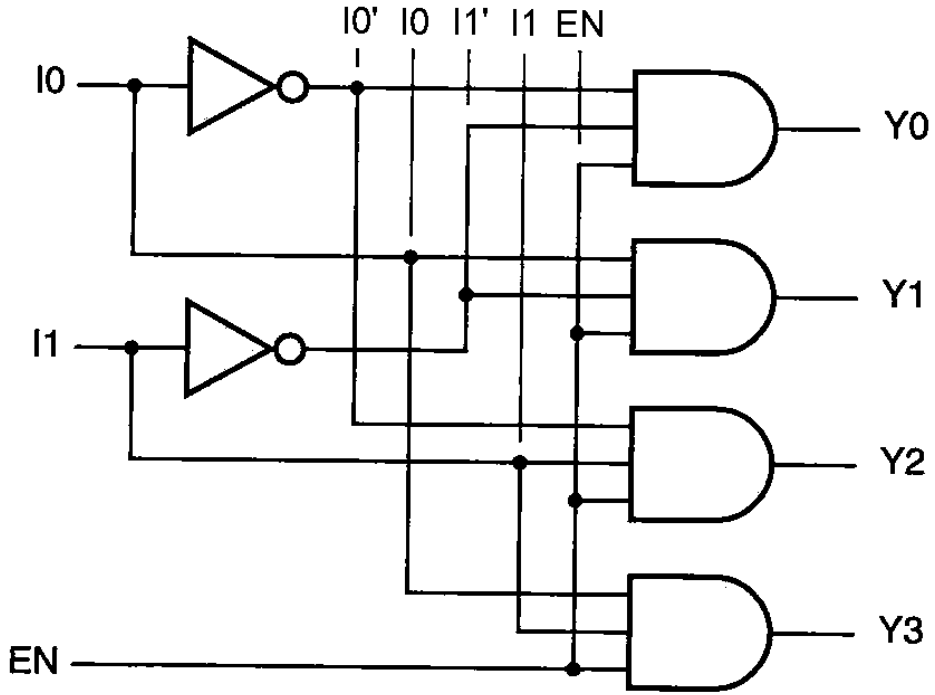
UnB Gama
O novo endereço da tecnologia

Decodificadores

Exemplo: Decodificador 2-para-4.



(a)



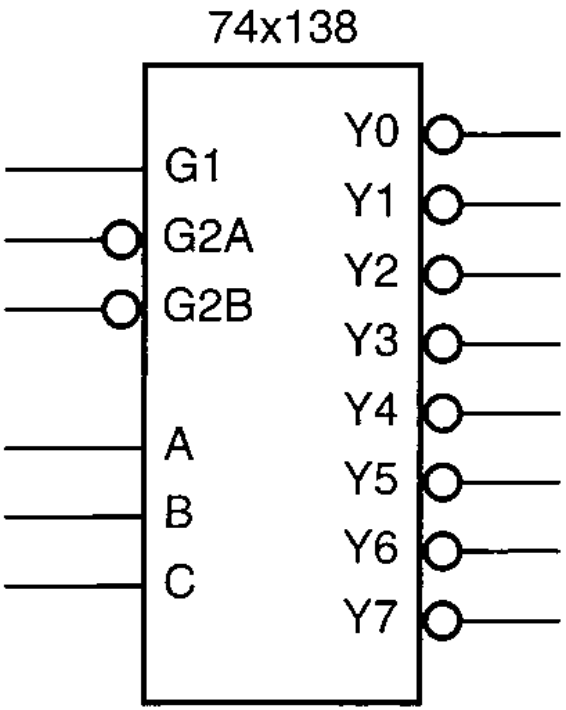
(b)

Decodificadores

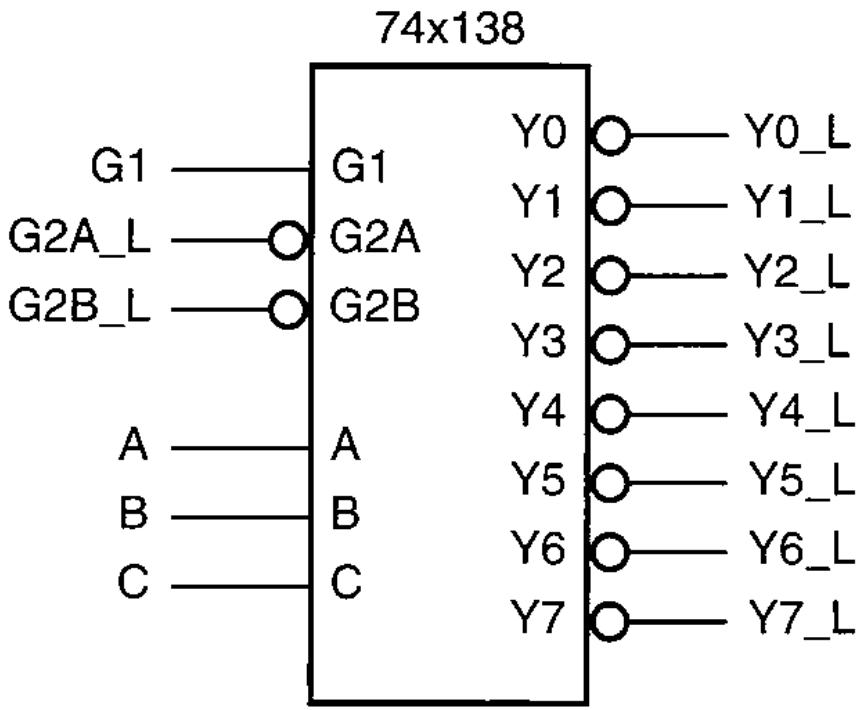
Exemplo: Decodificador 2-para-4.

Entradas			Saídas			
EN	I_1	I_0	Y_3	Y_2	Y_1	Y_0
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**

O CI 74x138



(a)



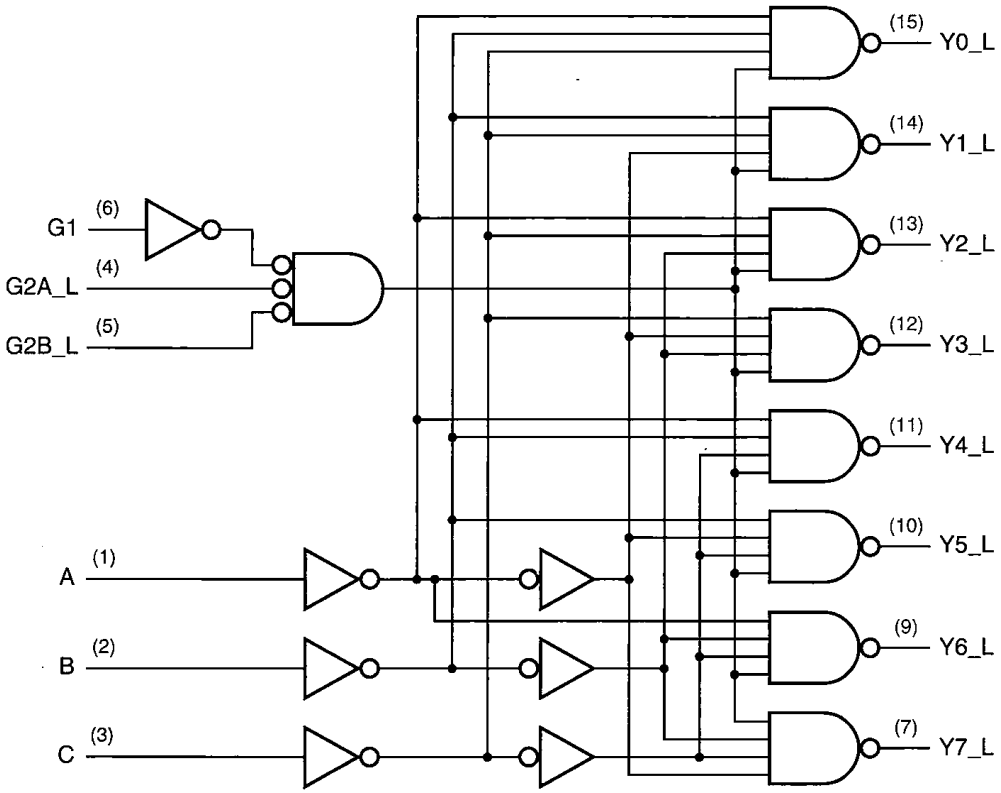
(b)

O CI 74x138

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	**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**

UnB Gama
O novo endereço da tecnologia

Internamente...



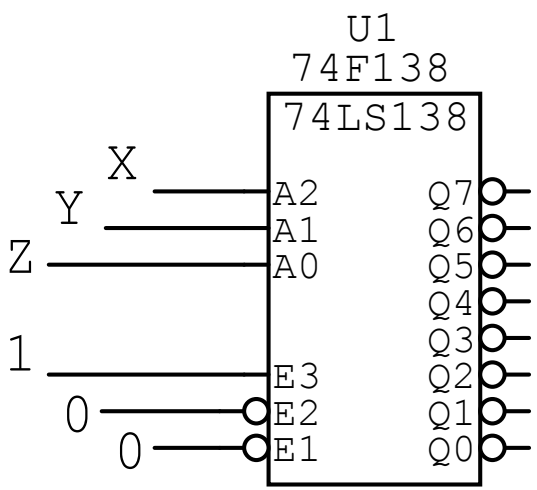
Usando o CI 74x138

Podemos usar decodificadores como **geradores de mintermos**.

Suponha que queremos realizar a função:

$$F(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \mathbf{X} \cdot \mathbf{Y} + \overline{\mathbf{X}} \cdot \mathbf{Y} \cdot \mathbf{Z}$$

Note que o 74x138 nos dá mintermos:

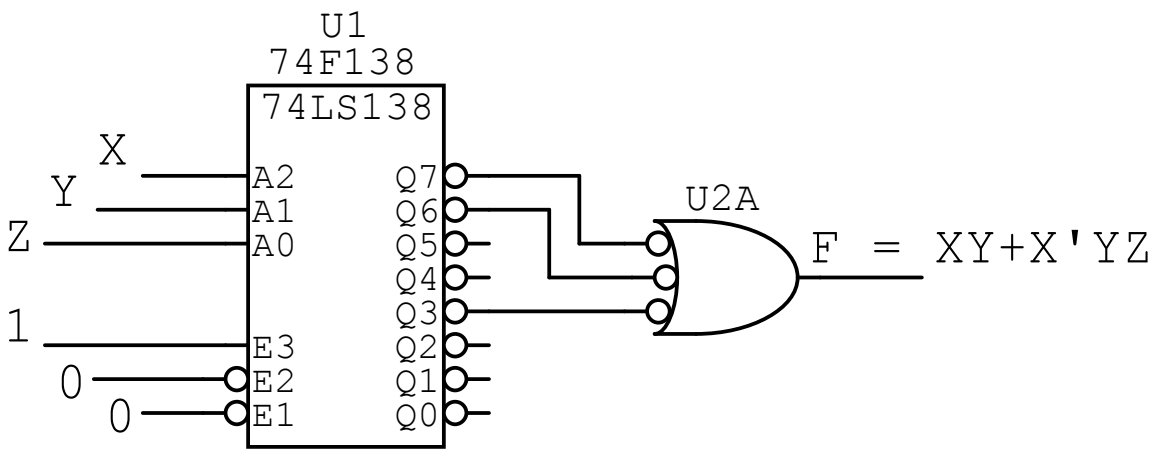


(XYZ)'
(XYZ')'
(XY'Z)'
(XY'Z')'
(X'YZ)'
(X'YZ')'
(X'Y'Z)'
(X'Y'Z')'

Usando o CI 74x138

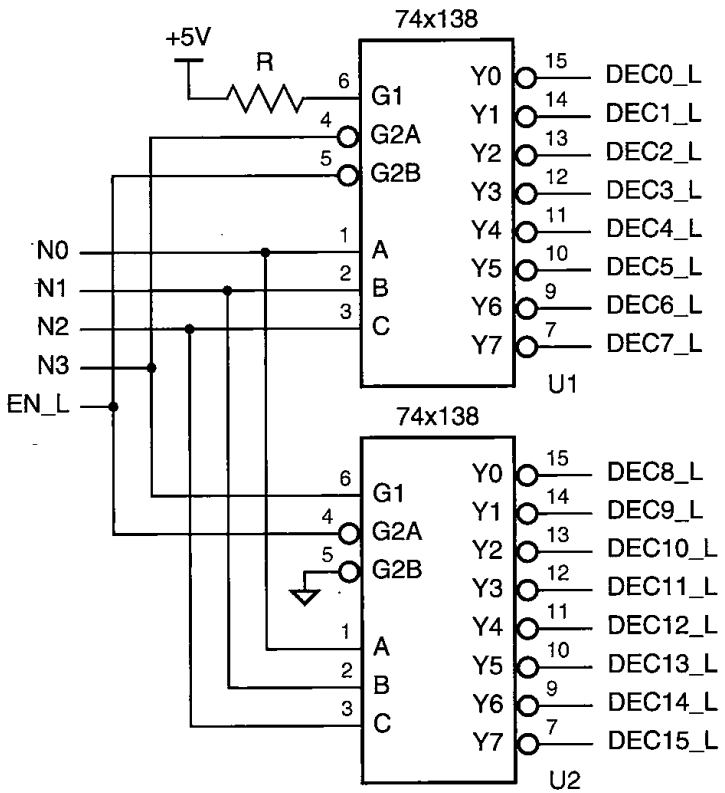
Note que podemos re-escrever a função na forma canônica:

$$
\begin{aligned}
F(\mathbf{X},\mathbf{Y},\mathbf{Z}) &= \mathbf{X}\cdot\mathbf{Y} + \overline{\mathbf{X}}\cdot\mathbf{Y}\cdot\mathbf{Z} \\
&= \mathbf{X}\cdot\mathbf{Y}\cdot\left(\mathbf{Z}+\overline{\mathbf{Z}}\right) + \overline{\mathbf{X}}\cdot\mathbf{Y}\cdot\mathbf{Z} \\
&= \mathbf{X}\cdot\mathbf{Y}\cdot\mathbf{Z} + \mathbf{X}\cdot\mathbf{Y}\cdot\overline{\mathbf{Z}} + \overline{\mathbf{X}}\cdot\mathbf{Y}\cdot\mathbf{Z}
\end{aligned}
$$



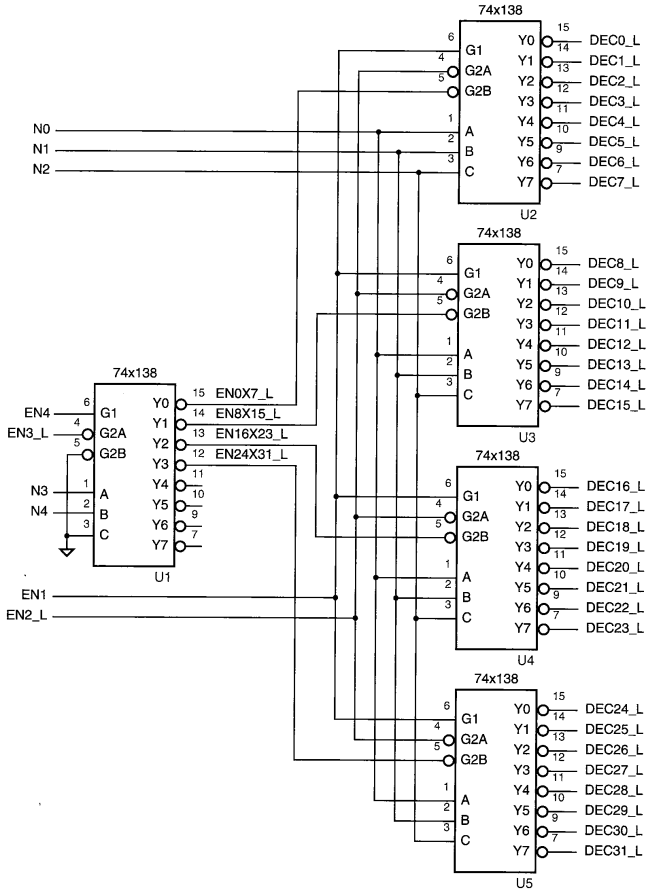
Usando o CI 74x138 para formar um decodificador 4-para-16

Cascateamento...



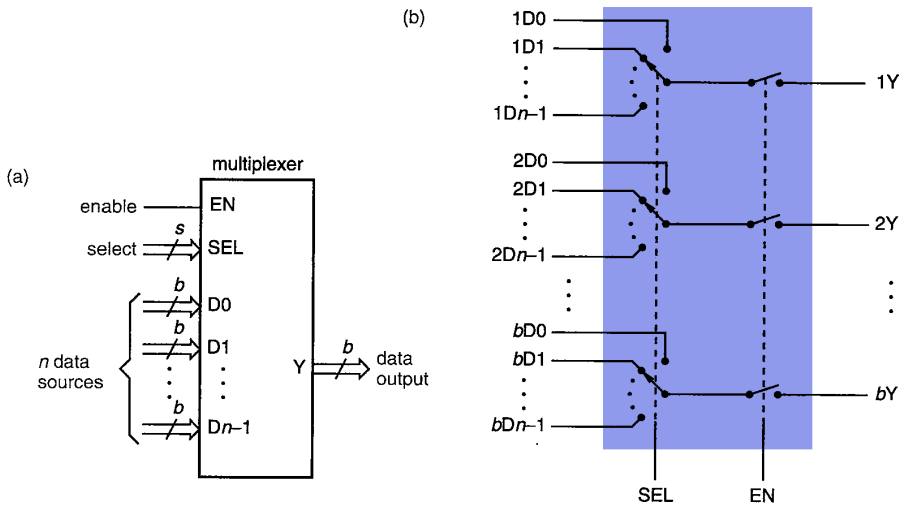
Usando o CI 74x138 para formar um decodificador 5-para-32

Basta dividir para conquistar!



Multiplexadores

Um multiplexador (ou **mux**) é um **switch/comutador** digital. Para cada saída, usamos n bits de **seleção** (em geral, com o nome de s_i) para selecionar uma dentre 2^n entradas (logo, $s = \lceil log_2 n \rceil$).

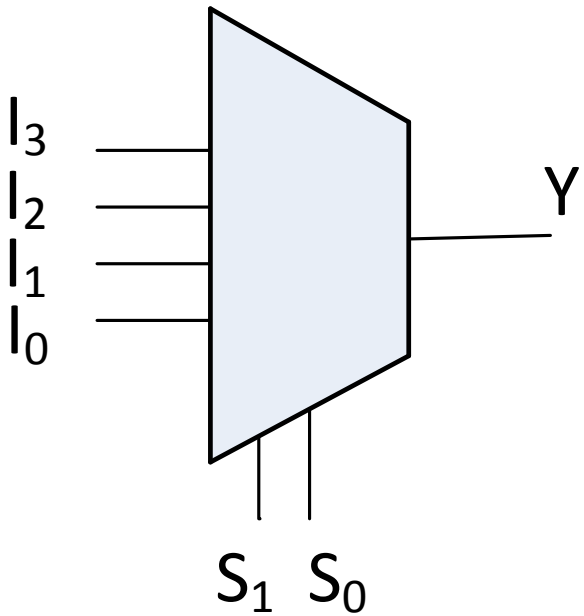


De forma geral, a equação do mux é:

$$Y = \sum_{i=0}^{n-1} EN \cdot m_i \cdot X_i$$

Multiplexador de 4 entradas

Podemos projetar um mux de 4 entradas e 1 saída. Para isso, precisamos de $s = \lceil log_2 4 \rceil = 2$ bits de seleção.

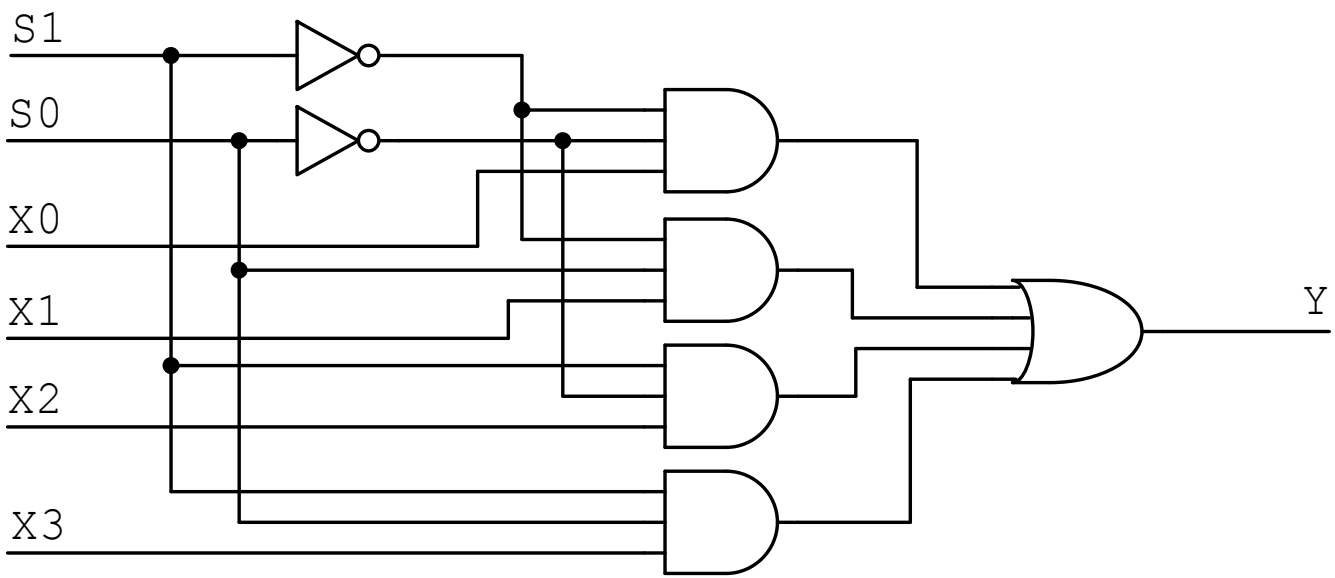


S_1	S_0	Y
0	**0**	$\mathbf{I_0}$
0	**1**	$\mathbf{I_1}$
1	**0**	$\mathbf{I_2}$
1	**1**	$\mathbf{I_3}$

Multiplexador de 4 entradas

Uma implementação básica:

$$\mathbf{Y = \overline{S_1} \cdot \overline{S_0} \cdot X_0 + \overline{S_1} \cdot S_0 \cdot X_1 + S_1 \cdot \overline{S_0} \cdot X_2 + S_1 \cdot S_0 \cdot X_3}$$



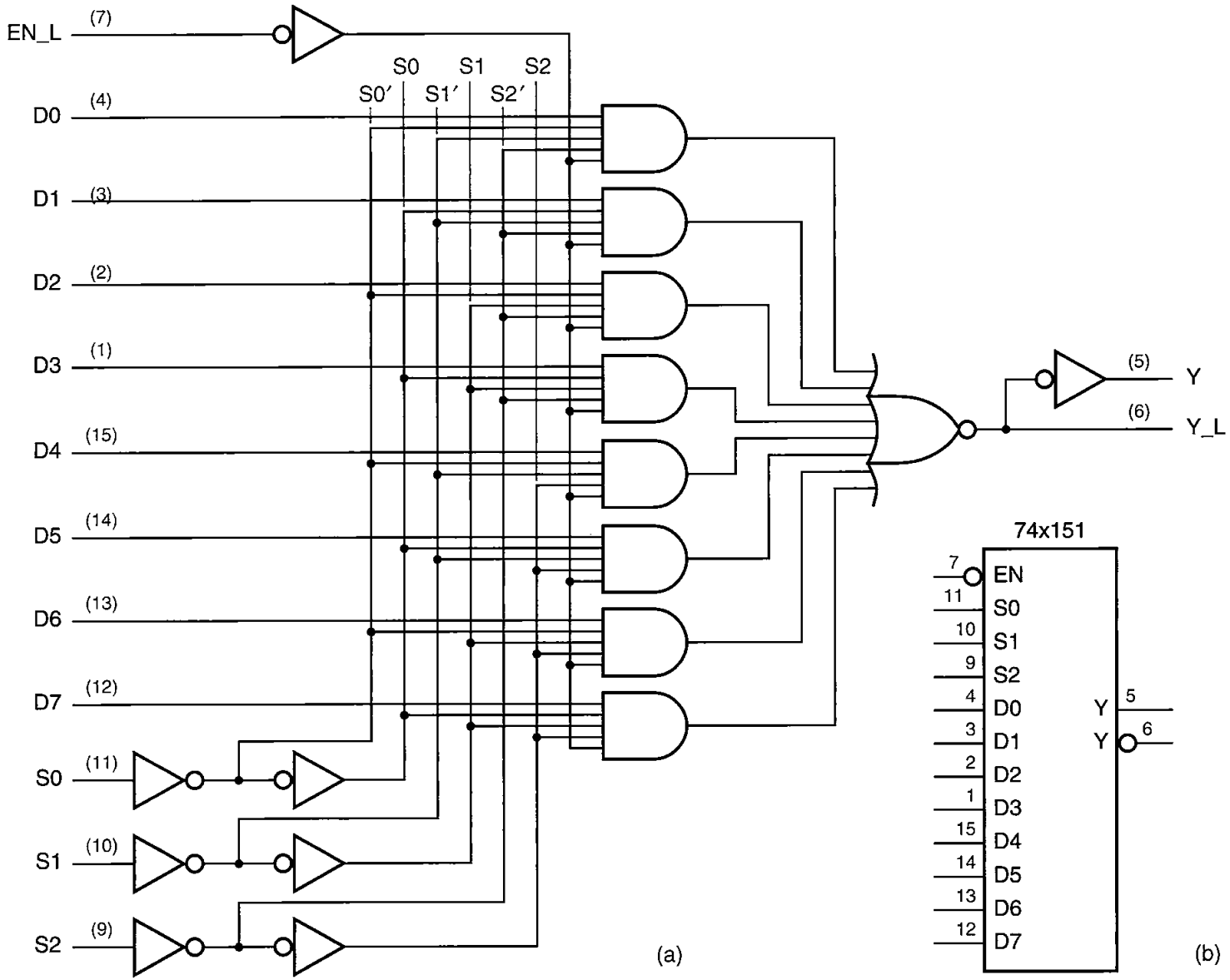
Standard MSI Multiplexers: 74x151 Mux 8-para-1

Tabela verdade:

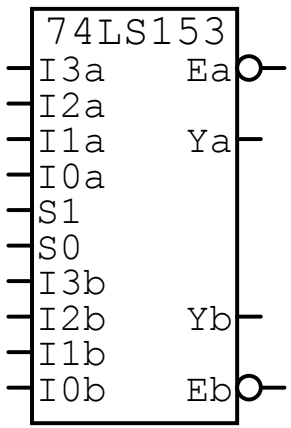
Entradas				Saídas	
EN_L	S_2	S_1	S_0	Y	Y_L
1	**x**	**x**	**x**	**0**	**1**
0	**0**	**0**	**0**	$\mathbf{D_0}$	$\overline{\mathbf{D_0}}$
0	**0**	**0**	**1**	$\mathbf{D_1}$	$\overline{\mathbf{D_1}}$
0	**0**	**1**	**0**	$\mathbf{D_2}$	$\overline{\mathbf{D_2}}$
0	**0**	**1**	**1**	$\mathbf{D_3}$	$\overline{\mathbf{D_3}}$
0	**1**	**0**	**0**	$\mathbf{D_4}$	$\overline{\mathbf{D_4}}$
0	**1**	**0**	**1**	$\mathbf{D_5}$	$\overline{\mathbf{D_5}}$
0	**1**	**1**	**0**	$\mathbf{D_6}$	$\overline{\mathbf{D_6}}$
0	**1**	**1**	**1**	$\mathbf{D_7}$	$\overline{\mathbf{D_7}}$

UnB Gama
O novo endereço da tecnologia

Standard MSI Multiplexers: 74x151 Mux 8-para-1



Standard MSI Multiplexers: 74x153 Dois Mux 4-para-1



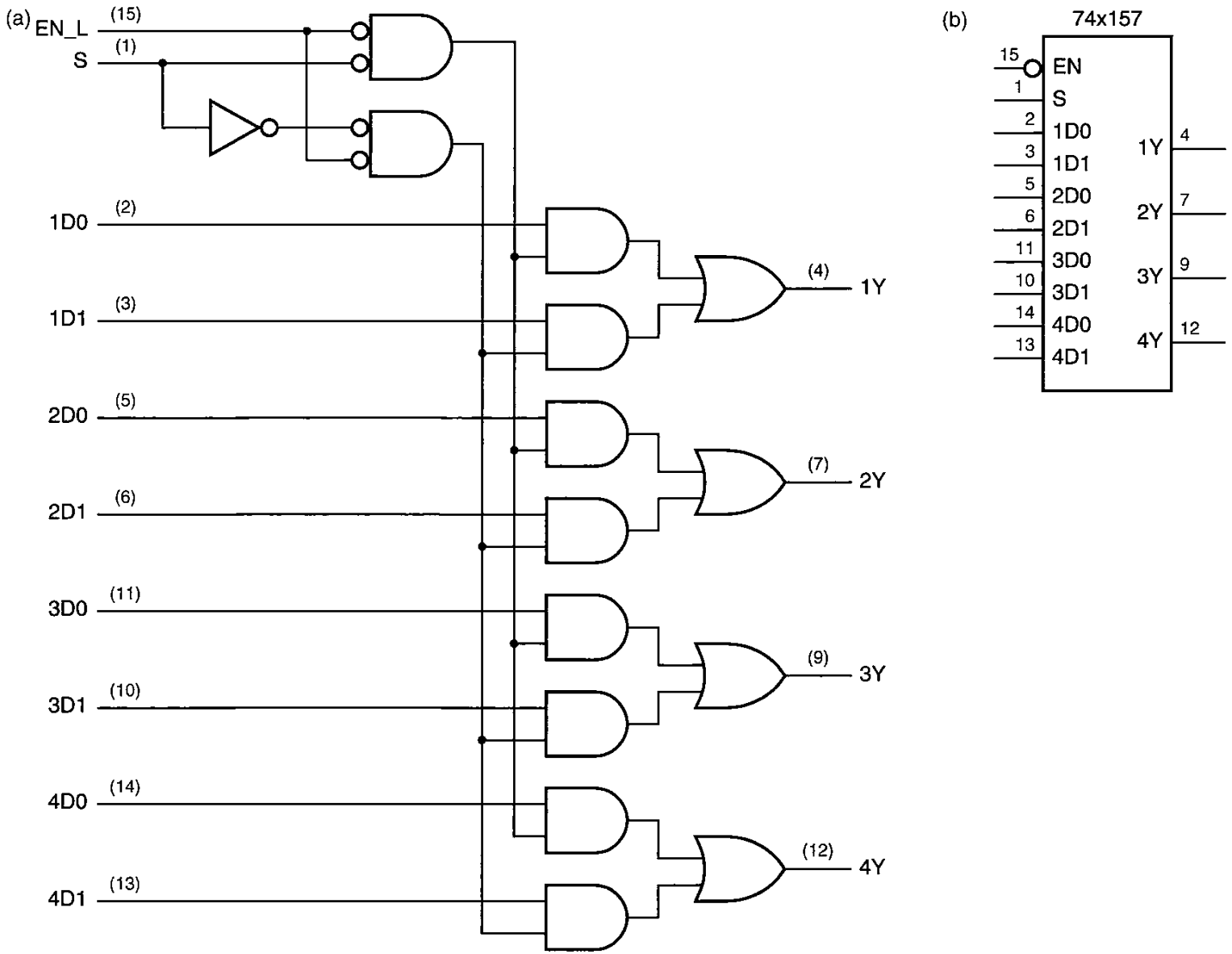
Ea	S_1	S_0	Ya
1	**x**	**x**	**0**
0	**0**	**0**	**I0a**
0	**0**	**1**	**I1a**
0	**1**	**0**	**I2a**
0	**1**	**1**	**I3a**

Eb	S_1	S_0	Yb
1	**x**	**x**	**0**
0	**0**	**0**	**I0b**
0	**0**	**1**	**I1b**
0	**1**	**0**	**I2b**
0	**1**	**1**	**I3b**

UnB Gama
O novo endereço da tecnologia

Standard MSI Multiplexers: 74x157 Quatro Mux 2-para-1

Insumo presente no laboratório:



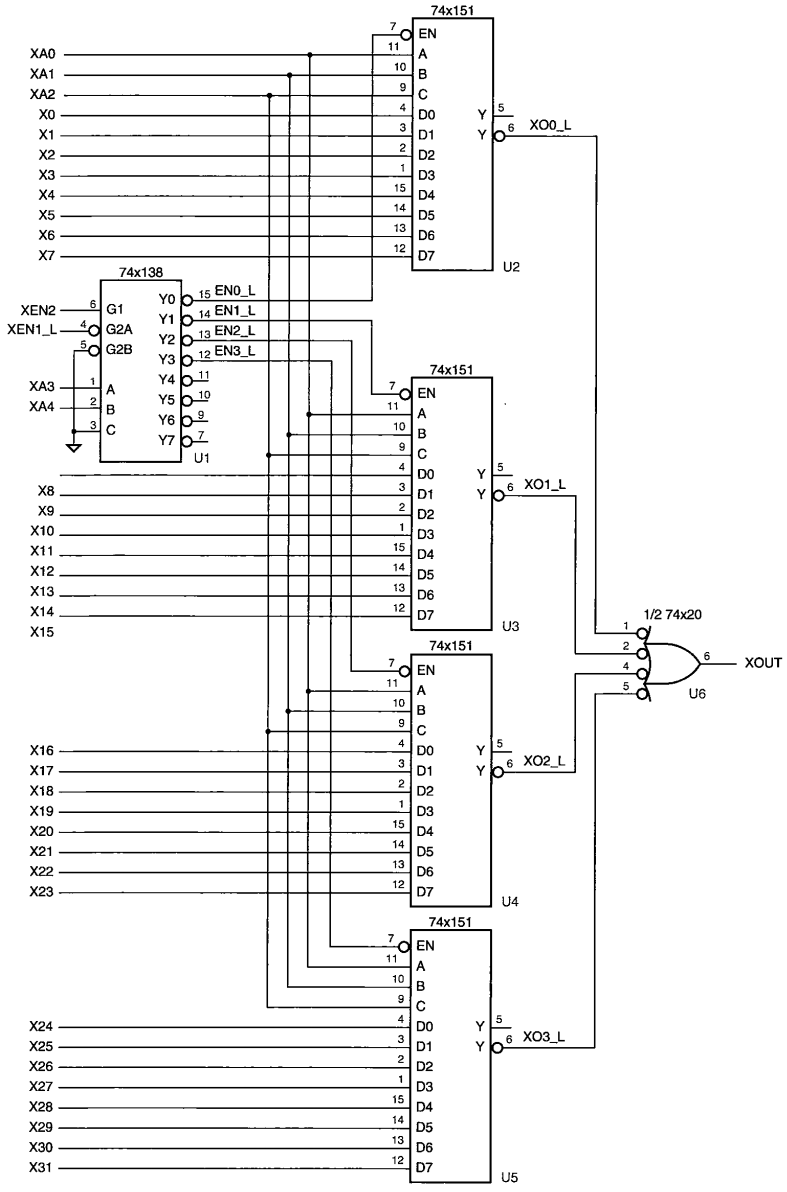
Multiplexadores com Saídas de Três Estados

Nesses multiplexadores, a entrada de **enable** força a saída para alta impedância caso esteja desativada. Temos três multiplexadores principais (com suas respectivas contrapartidas tri-state):

tri-state	$\rightarrow$	convencional
74x251	$\rightarrow$	74x151
74x253	$\rightarrow$	74x153
74x257	$\rightarrow$	74x157

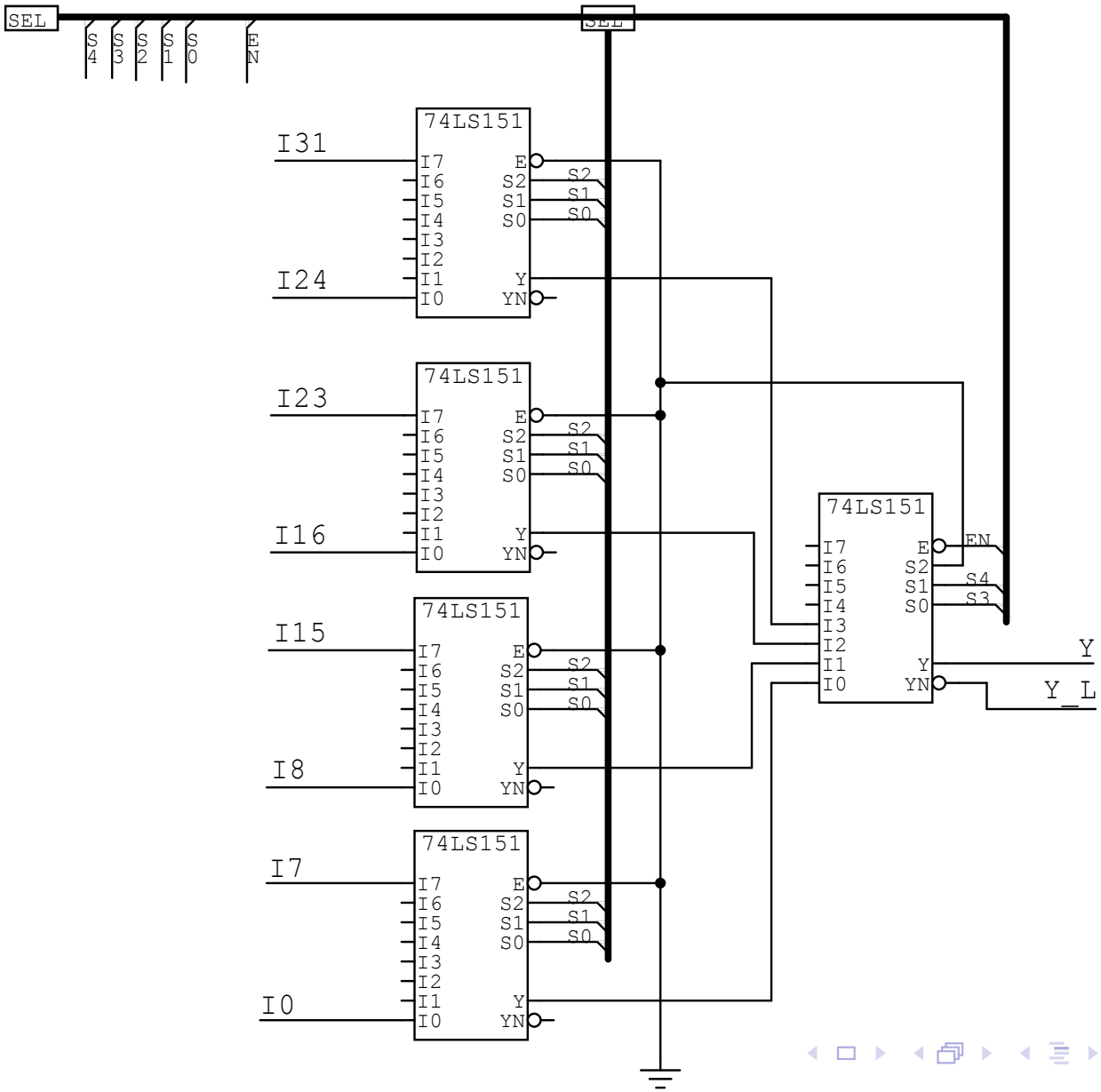
Multiplexadores em Cascata: Mux 32-para-1

Dividir para conquistar:

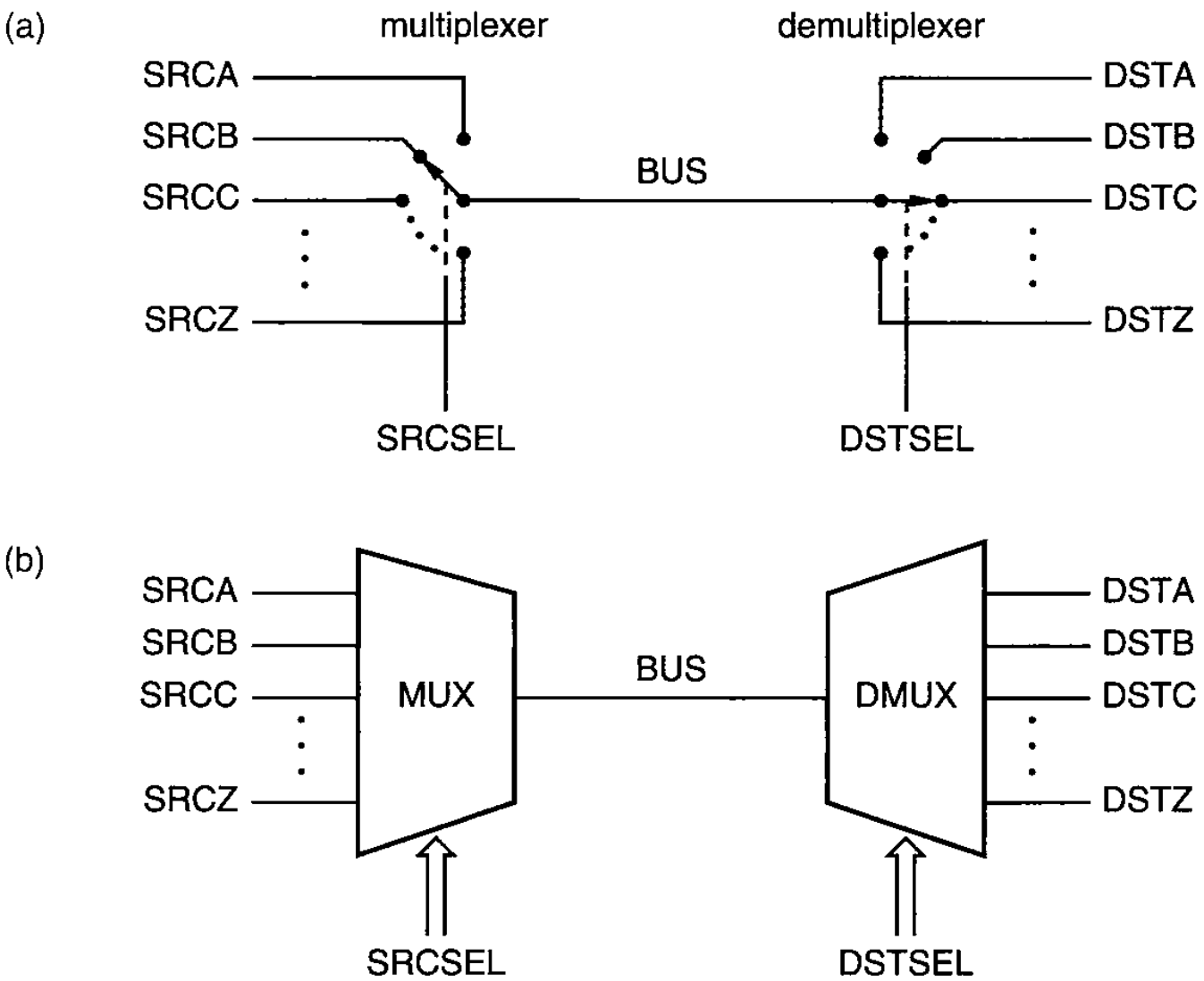


Multiplexadores em Cascata: Mux 32-para-1

Dividir para conquistar:



Demultiplexadores



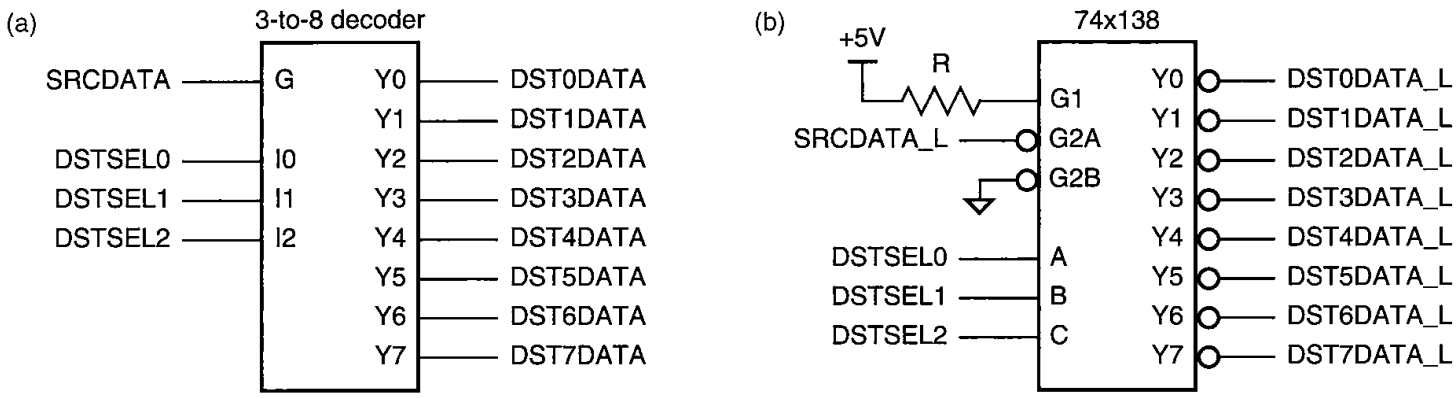
Demultiplexadores

Um demultiplexador (*demux*) tem 1 entrada, n bits de seleção e 2^n saídas.

O que é muito similar a um... **decodificador**!

UnB Gama
O novo endereço da tecnologia

Demultiplexadores



UnB Gama
O novo endereço da tecnologia