Spina

PCS3515 – Sistemas Digitais

Blocos Básicos

- Tri-State e Multiplexadores -

Seções 6.6 e 6.7 – livro texto

Com apoio do material dos demais professores

2018/1

Spina

Tri State

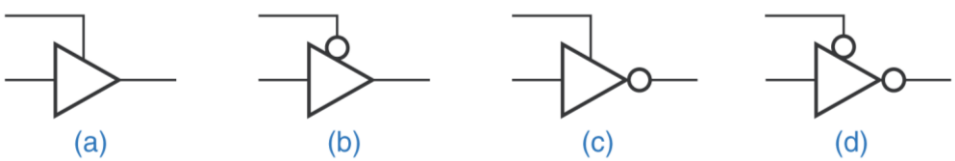


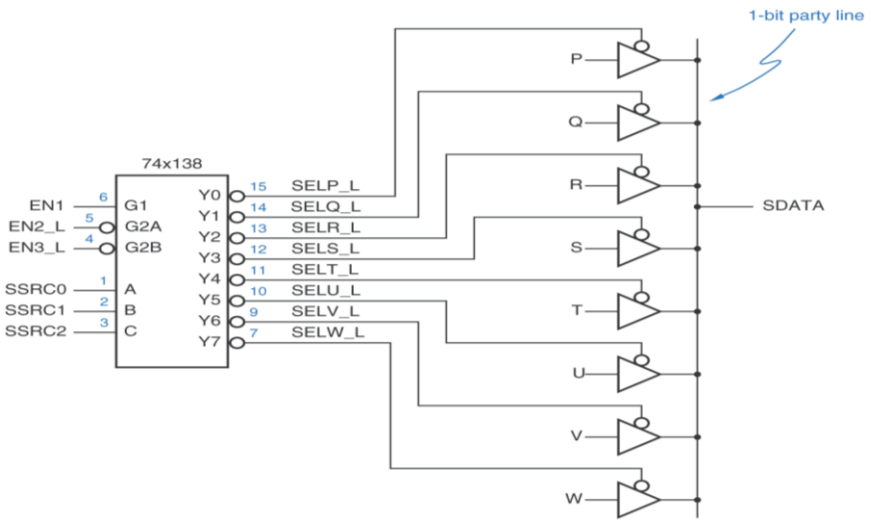
Figure 6-51

Various three-state buffers: (a) noninverting, active-high enable; (b) noninverting, active-low enable; (c) inverting, active-high enable; (d) inverting, active-low enable.

Spina

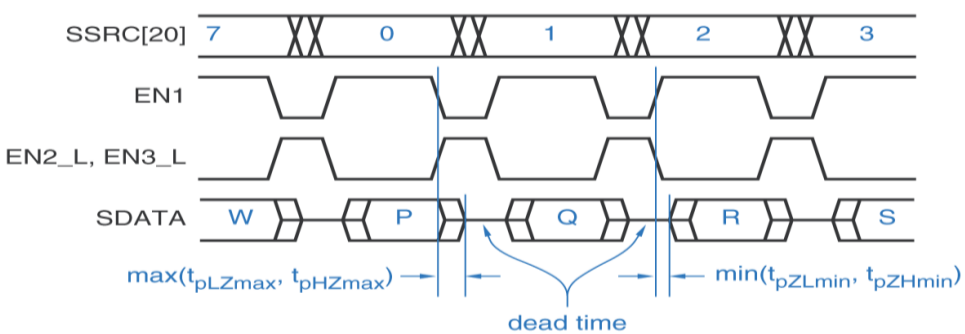
Spina

Compartilhamento de uma via



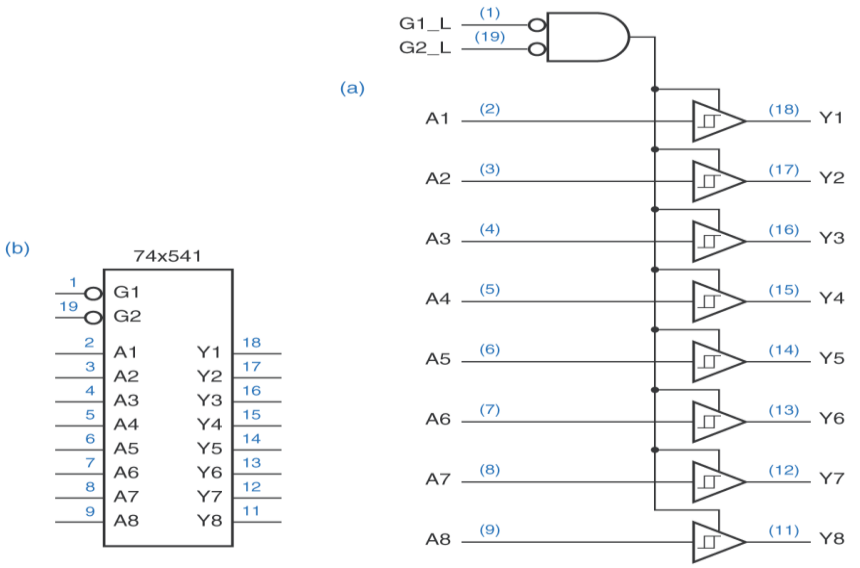
Spina

Timing para o compartilhamento



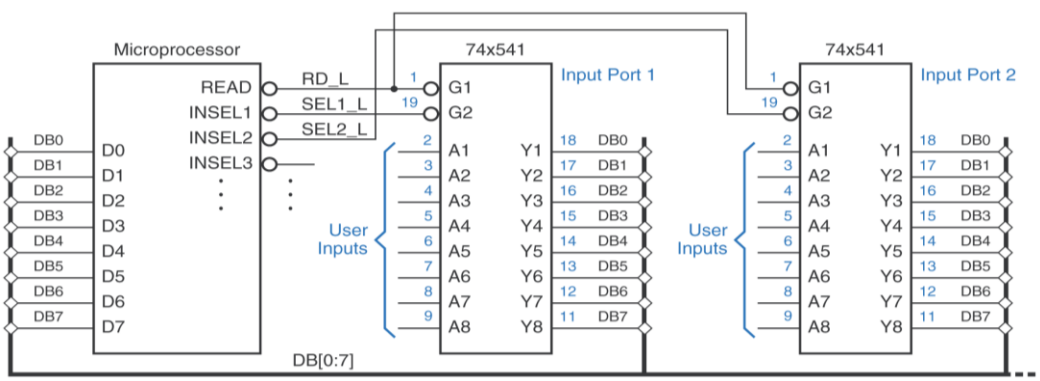
Spina

74x541



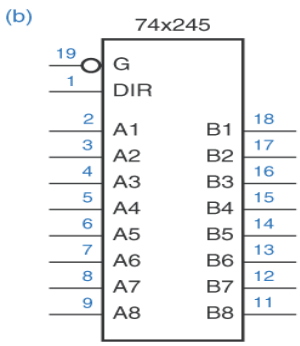
Spina

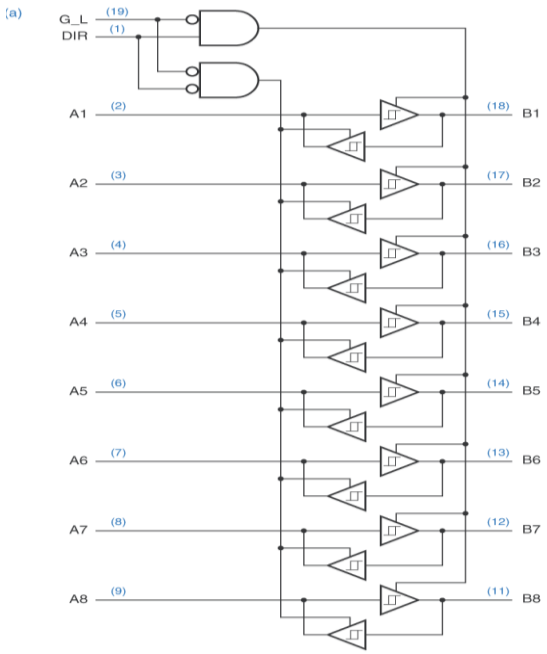
Usando com o microprocessador



Spina

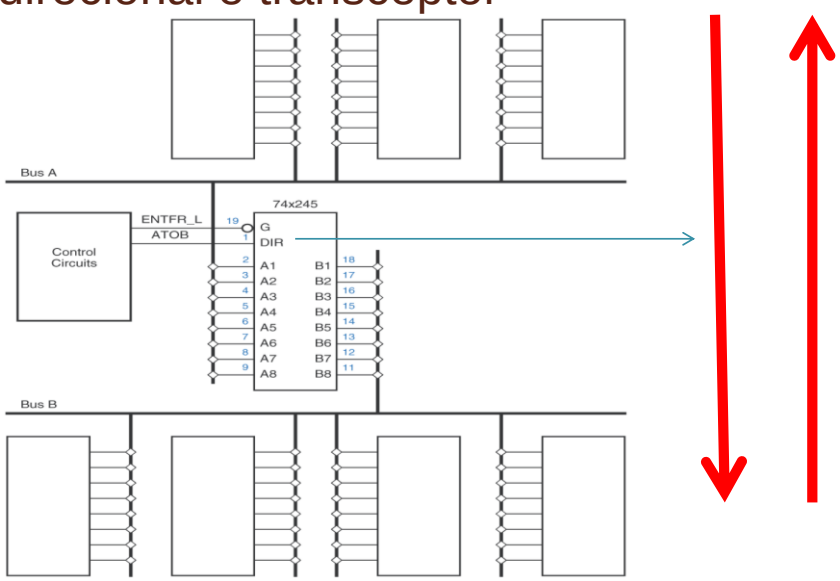
74x245





Spina

Bus bidirecional e transceptor



```
PACKAGE std_logic_1164 IS
-- logic state system (unresolved)
    TYPE std_ulogic IS ( 'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'H',  -- Weak     1
                         '-'   -- Don't care
                       );

-- unconstrained array of std_ulogic
    TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;

-- resolution function
    FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic;

-- *** industry standard logic type ***
    SUBTYPE std_logic IS resolved std_ulogic;
...
```

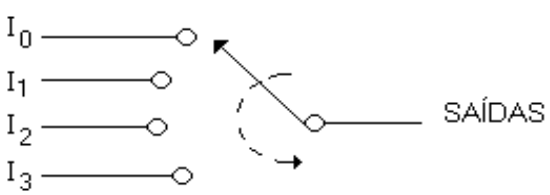
Table 6-36

IEEE 1164 package declarations for STD_ULOGIC and STD_LOGIC.

Spina

Multiplexadores – Conceito

Multiplexação – Conceito



Correspondente mecânico de um Multiplexador:

Chave seletora

Spina

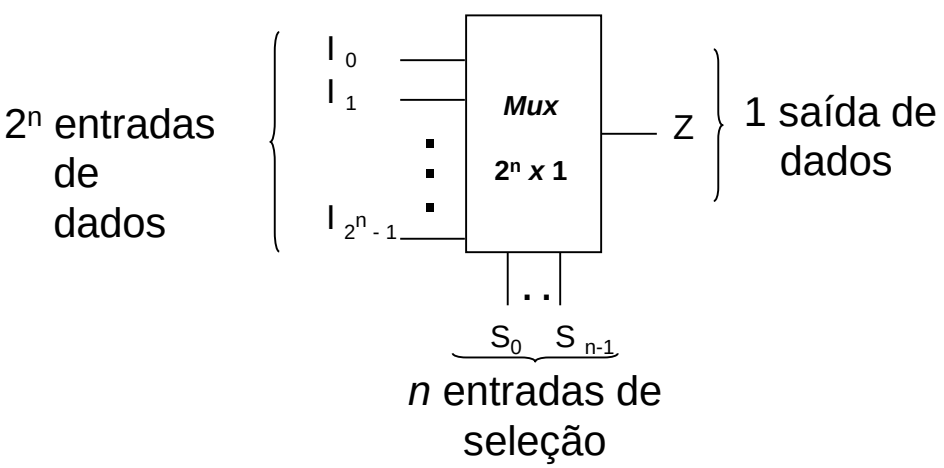
Spina

Multiplexadores

- Multiplexador ou *Multiplexer ou Mux*
  - É um Bloco Lógico Funcional (ou Bloco Combinatório Lógico) que possui 2^n entradas de dados e uma saída
  - Só uma das 2^n entradas é “conectada” à saída
  - *N* entradas adicionais de seleção indicam qual entrada de dados vai para a saída

Spina

Multiplexadores – Entradas e saídas



Spina

2. Multiplexadores

- Qual entrada de dados vai para a saída ?
  - As entradas de seleção formam uma palavra binária de *n* bits
  - A palavra pode assumir os valores de 0 a 2^n -1
  - As 2^n entradas de dados são numeradas de 0 a 2^n -1
  - A saída recebe a entrada de dados cujo índice corresponde ao valor da palavra binária das entradas de seleção
- Exemplo
  - Entradas de seleção = 011 (3_{10}); Z recebe I_3

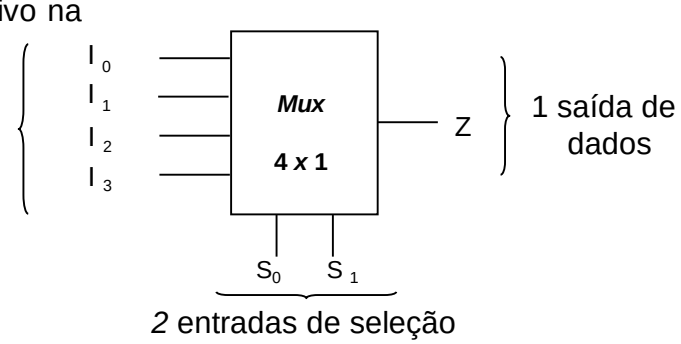
Spina

Multiplexadores – Notação

- Símbolo funcional (convenção):
  - Entradas de dados à esquerda
  - Saídas à direita
  - Índice menor indica bit menos significativo na palavra binária

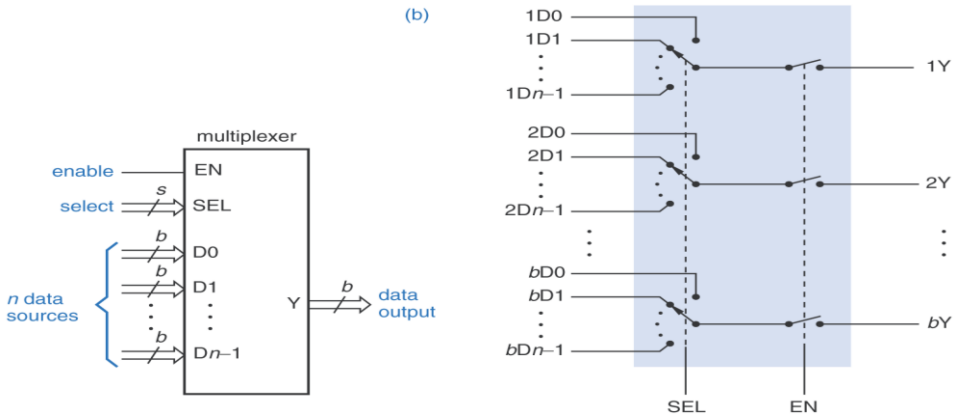
2^2 = 4 entradas de dados

- Denominação
  - Mux 8 por 1
  - Multiplexador 8 x 1



Spina

Equivalente Funcional múltiplo



Spina

Multiplexadores – Tabela da Verdade

Ex: Mux 4 por 1

Entradas de Seleção		Saída
S_1	S_0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

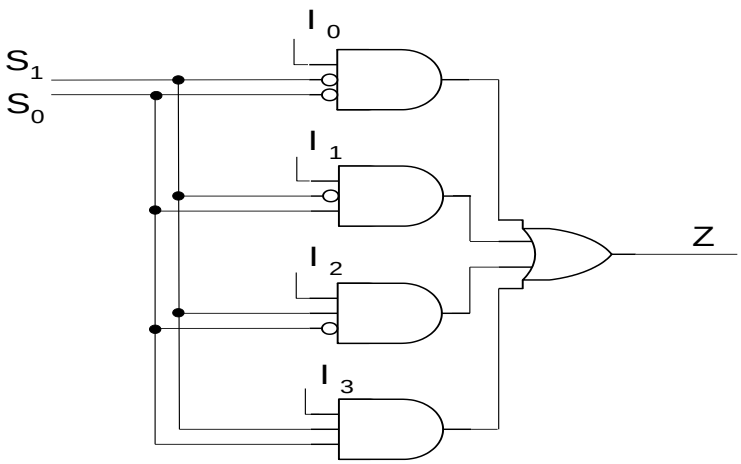
Mapa

S_0 \ S_1	0	1
0	I_0	I_2
1	I_1	I_3

$$Z = S_1'.S_0'.I_0 + S_1'.S_0.I_1 + S_1.S_0'.I_2 + S_1.S_0.I_3$$

Spina

Multiplexadores – Circuito interno



$$Z = S_1'.S_0'.I_0 + S_1'.S_0.I_1 + S_1.S_0'.I_2 + S_1.S_0.I_3$$

Cada mintermo é “multiplicado” pela entrada de dados correspondente

Spina

Multiplexadores – Entradas adicionais

- Enable
  - Permite habilitar/desabilitar o bloco todo
    - SE Enable ativo
      - Funcionamento normal
    - SE Enable inativo
      - Saída inativa independentemente do código nas entradas de seleção e das entradas de dados
  - Atua em todos os produtos (portas E)

Spina

Multiplexadores – Tabela da Verdade

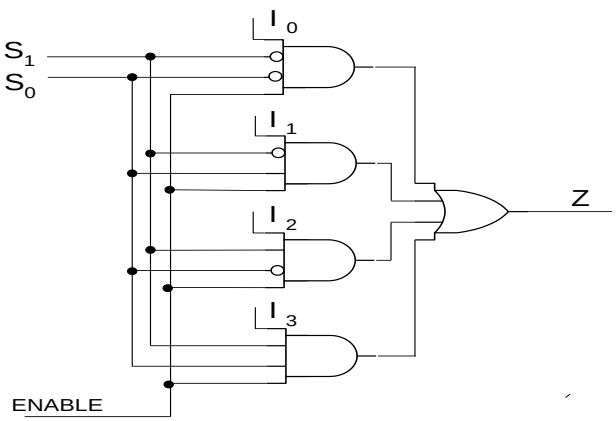
Ex: Mux 4 por 1 com *Enable* (EN)

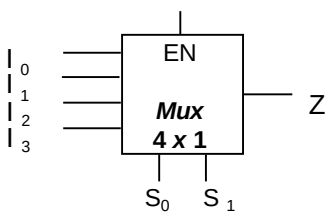
Entradas			Saída
EN	S_1	S_0	Z
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3
0	X	X	0

$$Z = EN.S_1'.S_0'.I_0 + EN.S_1'.S_0.I_1 + EN.S_1.S_0'.I_2 + EN.S_1.S_0.I_3$$

Spina

Multiplexadores – Circuito interno com *Enable*





$$Z = EN.S_1'.S_0'.I_0 + EN.S_1'.S_0.I_1 + EN.S_1.S_0'.I_2 + EN.S_1.S_0.I_3$$

Spina

Multiplexadores – Saídas invertidas

- Multiplexadores podem ter a saída invertida ou complementada
  - Diz-se saída *active-low* – Ativa em ZERO;
  - Implicações:
    - Na tabela da verdade – Inversão na saída (Z' = I_i');
    - No circuito – Uso de NOR;
    - Nomenclatura – Z' ou $\overline{Z}$

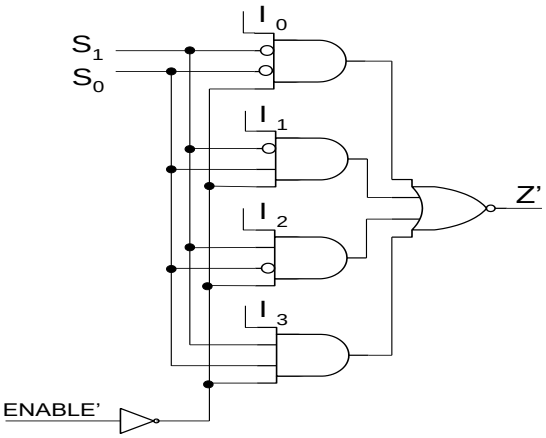
Spina

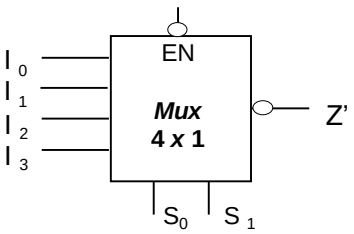
Multiplexadores – Entradas *active low*

- Entrada ENABLE também pode ser *active-low*
  - SE Enable'=0, circuito habilitado
  - SE Enable'=1, saídas todas desabilitadas
  - Implicações
    - na tabela verdade: inversão na entrada
    - no circuito: uso de inversor
    - Nomenclatura: EN' ou $\overline{EN}$

Spina

Mux – Entradas e saídas *active low*

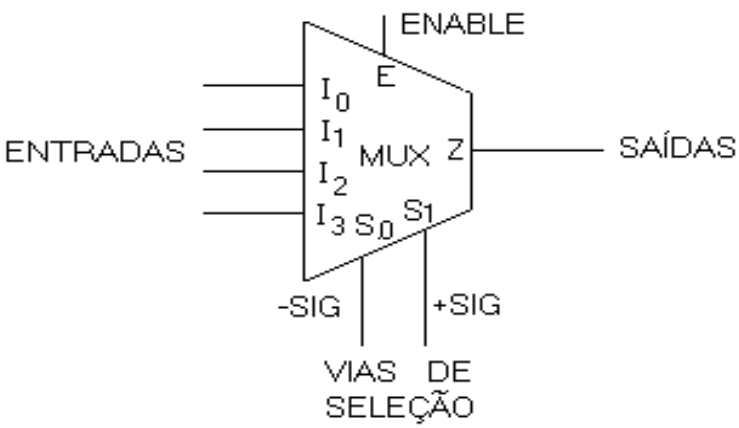




Entradas			Saída
EN'	S_1	S_0	Z'
0	0	0	I_0'
0	0	1	I_1'
0	1	0	I_2'
0	1	1	I_3'
1	X	X	1

Spina

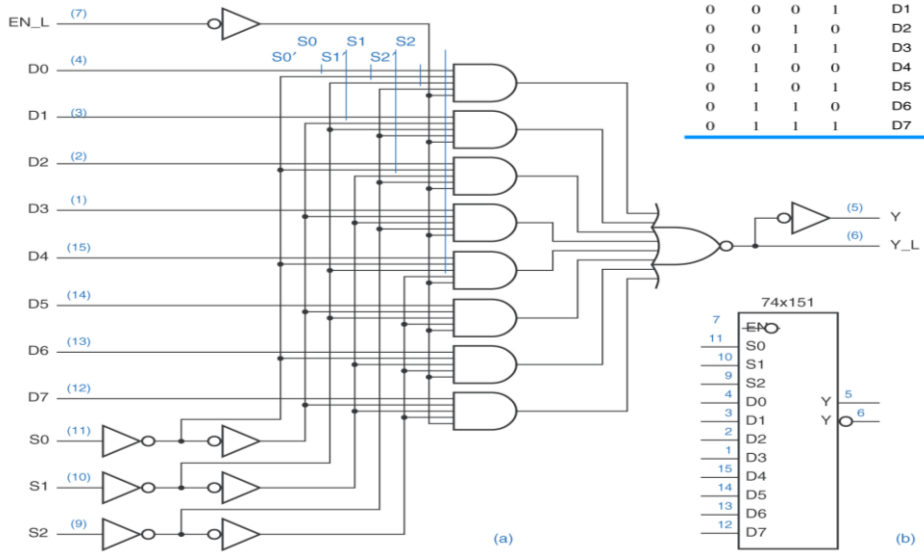
2. Multiplexadores – Símbolo alternativo



Spina

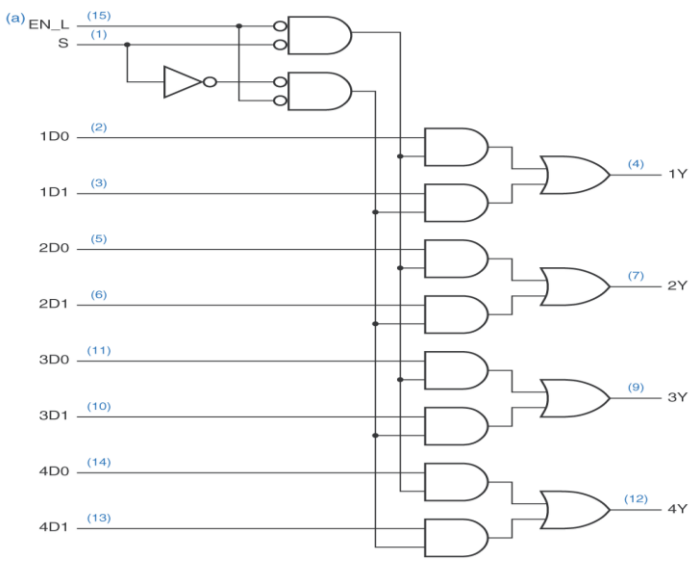
Mux 8:1 (74x151)

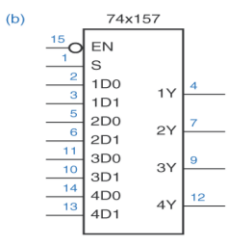
Inputs				Outputs	
EN_L	S2	S1	S0	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′



Spina

74x157

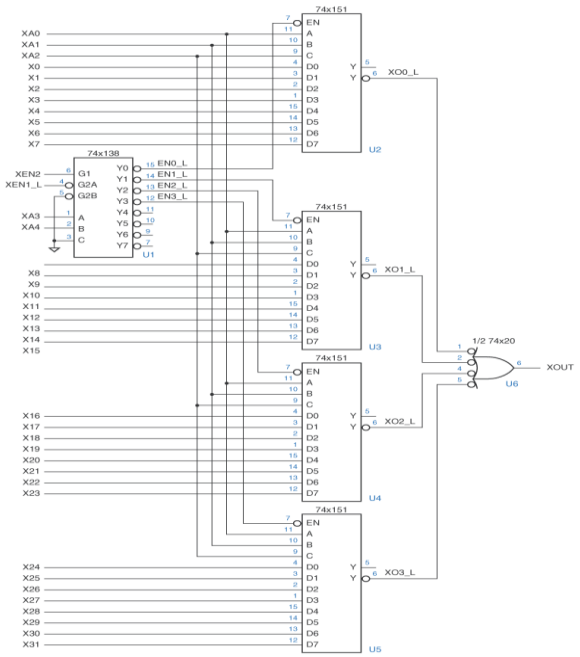




Inputs		Outputs			
EN_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1D0	2D0	3D0	4D0
0	1	1D1	2D1	3D1	4D1

Spina

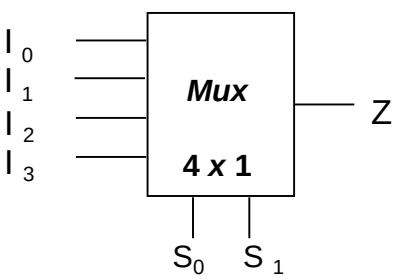
Mux 32:1



Spina

Multiplexadores – Uso

- Síntese de funções
  - Qualquer função de chaveamento de *n* variáveis pode ser implementada com um único MUX 2^n **:** 1
  - Exemplo: Função de 2 variáveis e Mux 4x1



S_0 \ S_1	0	1
0	I_0	I_2
1	I_1	I_3

Mapa do Mux 4*x*1

$$Z = S_1'.S_0'.I_0 + S_1'.S_0.I_1 + S_1.S_0'.I_2 + S_1.S_0.I_3$$

Multiplexadores – Síntese de funções

Mapa do Mux 4x1

S_0 \ S_1	0	1
0	I_0	I_2
1	I_1	I_3

Mapa da Função F(b,a)

a \ b	0	1
0		1
1	1	

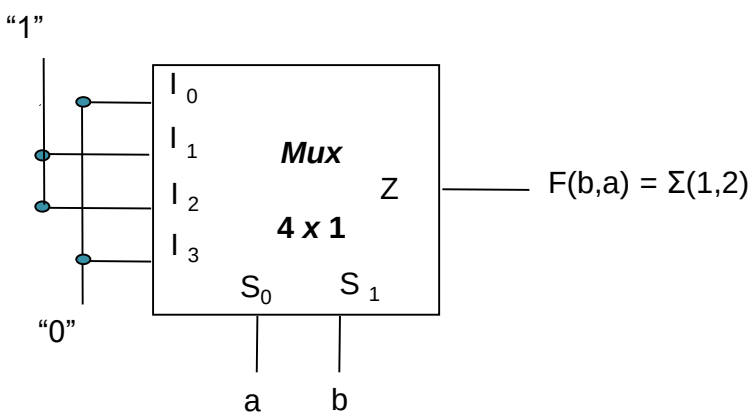
F(b,a) = Σ(1,2)

- Da comparação dos dois mapas obtém-se:

  $I_0 = 0$, $I_1 = 1$, $I_2 = 1$, $I_3 = 0$

Spina

Multiplexadores – Síntese de funções



Spina

Multiplexadores – Síntese de funções

- Síntese de funções com Mux de menor ordem
  - Implementar função de chaveamento de *n* variáveis com um único MUX 2^{n-1} **:** 1
  - Exemplo:
    - Função de 4 variáveis
    - Mux 8x1 (3 variáveis de seleção)

Spina

Multiplexadores – Síntese de funções

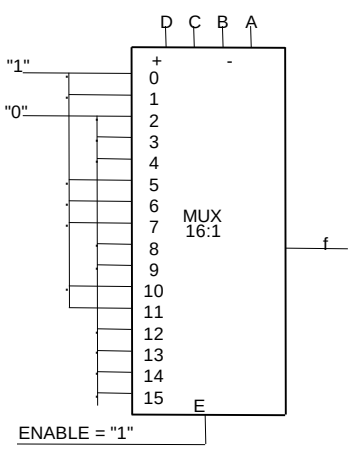
- Solução com Mux de 16x1

BA \ DC	00	01	11	10
00	I_0	I_4	I_{12}	I_8
01	I_1	I_5	I_{13}	I_9
11	I_3	I_7	I_{15}	I_{11}
10	I_2	I_6	I_{14}	I_{10}

MAPA DE MUX 16:1

BA \ DC	00	01	11	10
00	1			
01	1	1		
11		1		1
10		1		1

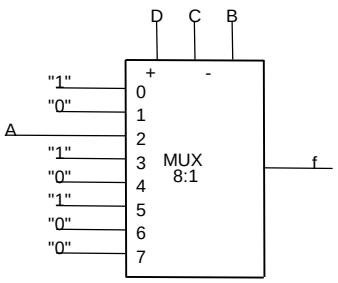
$F = \Sigma(0,1,5,6,7,10,11)$

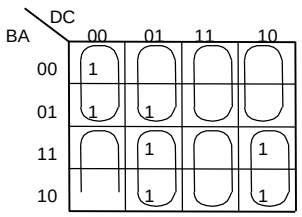


Spina

Multiplexadores – Síntese de funções

- Solução com Mux de 8x1 (solução 1)



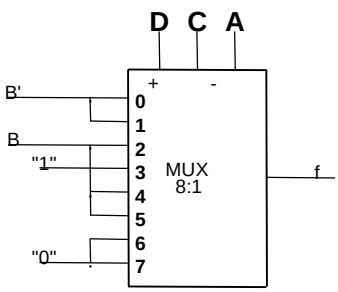


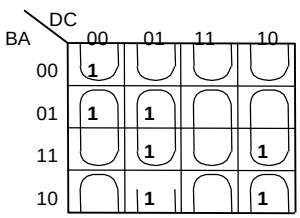
pois f = D‘C‘B' + D‘CB‘A + D‘CB + DC‘B

Spina

Multiplexadores – Síntese de funções

- Solução com Mux de 8x1 (solução 2)



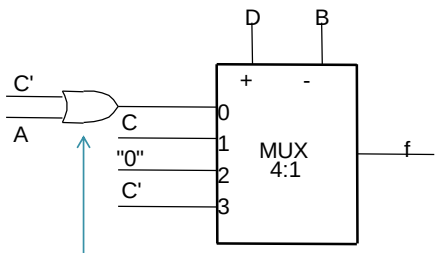


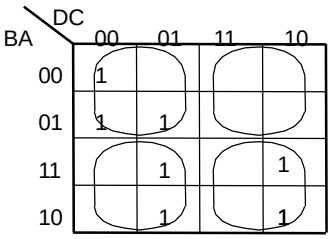
pois f = D‘C‘B‘A' + D‘C‘B’A + D‘CBA' + D‘CA
+ DC‘BA' + DC‘BA

Spina

Multiplexadores – Síntese de funções

- Solução com Mux de 4x1





Normalmente não recomendado!

Spina

Exercícios

- 1. Sintetize a função de chaveamento abaixo utilizando um decodificador 4x16.

  F(d,c,b,a) = Σ(0,1,3,8,10)

- 2. Sintetize a função de chaveamento dada pela tabela verdade ao lado utilizando um decodificador 3x8.

c	b	a	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Spina

Spina

Exercícios

- 3. Sintetize a função de chaveamento do exercício 4.1. utilizando um decodificador 4x16 com saídas *active low*.

- 4. Sintetize a função de chaveamento do exercício 4.2. utilizando um decodificador 3x8 com saídas *active low*.

Spina

Exercícios

- 7. Sintetize a função F(d,c,b,a) abaixo utilizando:
  - a. Um Mux de 16x1
  - b. Um Mux de 8x1
  - c. Um Mux de 4x1

$$F(d,c,b,a) = \Sigma(0,1,3,8,10)$$

Multiplexação do canal

12/6

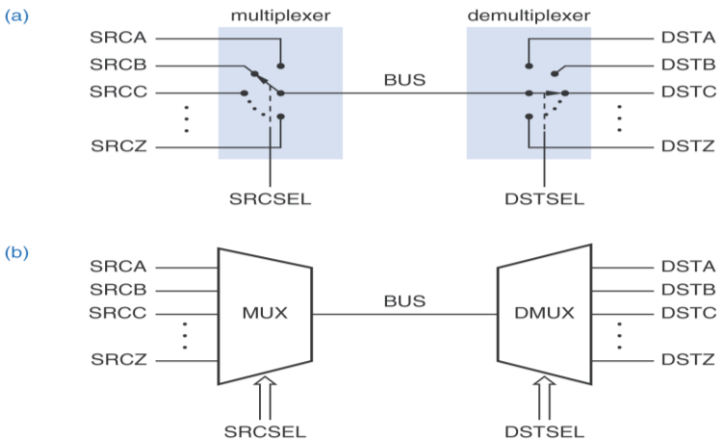


Figure 6-64

A mux driving a bus and a demultiplexer receiving the bus: (a) switch equivalent; (b) block-diagram symbols.

Spina

Dec como demultiplexer

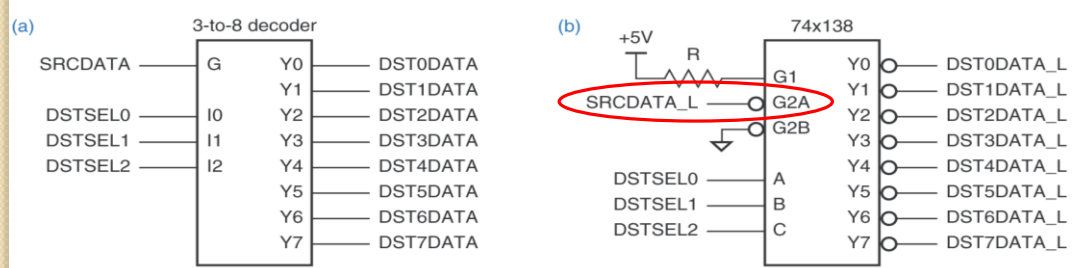


Figure 6-65

Using a 3-to-8 binary decoder as a 1-bit, 8-output demultiplexer: (a) generic decoder; (b) 74×138.

Spina

Spina

Mux 4 x 1 [1/3]

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux4x1en is
  port(I0, I1, I2, I3: in std_logic;
       S0, S1: in std_logic;
       enable: in std_logic;
       Z: out std_logic);
end mux4x1en;
```

Spina

Mux 4 x 1 [2/3]

```
architecture simpleMux of mux4x1en is
    signal inter0, inter1, inter2, inter3 : std_logic;
begin
  inter0 <= I0 and not S0 and not S1 and enable;
  inter1 <= I1 and     S0 and not S1 and enable;
  inter2 <= I2 and not S0 and     S1 and enable;
  inter3 <= I3 and     S0 and     S1 and enable;
  Z <= inter0 or inter1 or inter2 or inter3;
end simpleMux;
```

Mux 4 x 1 [3a/3]

```
architecture componentMux of mux4x1en is
    signal inter0, inter1, inter2, inter3: std_logic;
    signal NOTS0, NOTS1: std_logic;
    component \NOT\ port(\IN\: in std_logic; \OUT\: out std_logic);
    end component;
    component and4 port(IN1, IN2, IN3, IN4: in std_logic; \OUT\:
    out std_logic); end component;
    component or4 port(IN1, IN2, IN3, IN4: in std_logic; \OUT\:
    out std_logic); end component;
```

Spina

Mux 4 x 1 [3b/3]

```
begin
  inv_0: \NOT\ port map(S0, NOTS0);
  inv_1: \NOT\ port map(S1, NOTS1);
  and_0: and4 port map(I0, NOTS0, NOTS1, enable, inter0);
  and_1: and4 port map(I1,    S0, NOTS1, enable, inter1);
  and_2: and4 port map(I2, NOTS0,    S1, enable, inter2);
  and_3: and4 port map(I3,    S0,    S1, enable, inter3);
orFinal: or4 port map(inter0, inter1, inter2, inter3, Z);
end componentMux;
```

Spina

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux4in8b is
    port (
        S: in STD_LOGIC_VECTOR (1 downto 0);       -- Select inputs, 0-3 ==> A-D
        A, B, C, D: in STD_LOGIC_VECTOR (1 to 8); -- Data bus input
        Y: out STD_LOGIC_VECTOR (1 to 8)          -- Data bus output
    );
end mux4in8b;

architecture mux4in8b of mux4in8b is
begin
    with S select Y <=
      A when "00",
      B when "01",
      C when "10",
      D when "11",
      (others => 'U') when others; -- this creates an 8-bit vector of 'U'
end mux4in8b;
```

Table 6-48

Dataflow VHDL program for a 4-input, 8-bit multiplexer.

Mux com 4 entradas de 8 bits cada para uma saída de 8 bits !

Spina

```
architecture mux4in8p of mux4in8b is
begin
process(S, A, B, C, D)
  begin
    case S is
      when "00" => Y <= A;
      when "01" => Y <= B;
      when "10" => Y <= C;
      when "11" => Y <= D;
      when others => Y <= (others => 'U');  -- 8-bit vector of 'U'
    end case;
  end process;
end mux4in8p;
```

Table 6-49

Behavioral architecture for a 4-input, 8-bit multiplexer.

Spina

```
library IEEE;
use IEEE.std_logic_1164.all;
entity mux4in3b is
    port (
        S: in STD_LOGIC_VECTOR (2 downto 0);        -- Select inputs, 0-7 ==> ABACADAB
        A, B, C, D: in STD_LOGIC_VECTOR (1 to 18); -- Data bus inputs
        Y: out STD_LOGIC_VECTOR (1 to 18)           -- Data bus output
    );
end mux4in3b;
architecture mux4in3p of mux4in3b is
begin
process(S, A, B, C, D)
variable i: INTEGER;
  begin
    case S is
      when "000" | "010" | "100" | "110" => Y <= A;
      when "001" | "111" => Y <= B;
      when "011" => Y <= C;
      when "101" => Y <= D;
      when others => Y <= (others => 'U'); -- 18-bit vector of 'U'
    end case;
  end process;
end mux4in3p;
```

Table 6-50

Behavioral VHDL program for a specialized 4-input, 3-bit multiplexer.

Só usa 4 das possíveis 8 entradas

Spina

Lição de Casa

- Leitura Obrigatória:
  - Capítulo 6 do Livro Texto, ênfase em 6.4, 6.5, 6.7, 6.8.
- Exercícios obrigatórios:
  - Capítulo 6 do Livro Texto.

Spina

Spina

Livro Texto

- Wakerly, J.F.; *Digital Design – Principles & Practices;* Fourth Edition, ISBN: 0-13-186389-4, Pearson & Prentice-Hall, Upper Saddle, River, New Jersey, 07458, 2006.

Spina

Bibliografia Adicional

- Fregni, Edson; Saraiva, Antônio. *Engenharia do Projeto Lógico Digital*. Editora Edgard Blücher Ltda. São Paulo, SP, Brasil, 1.995;

- Ranzini, Edith; Fregni, Edson. Teoria da Comutação: Introdução aos Circuitos Digitais (Partes 1 e 2). Notas de Aula de PCS214, PCS/EPUSP, Agosto de 1.996.

Spina

Bibliografia Adicional

- Tocci, Ronald; Widmer, Neal S. Sistemas Digitais – Princípios e Aplicações. 8ª Edição, Pearson/Prentice-Hall, São Paulo, 2.003.