PCS 3115

Sistemas Digitais I

Circuitos Combinatórios

Blocos Básicos:

(De)Multiplexadores e Dispositivos tri-state

Prof. Dr. Marcos A. Simplicio Jr.

versão: 3.0 (Jan/2016)

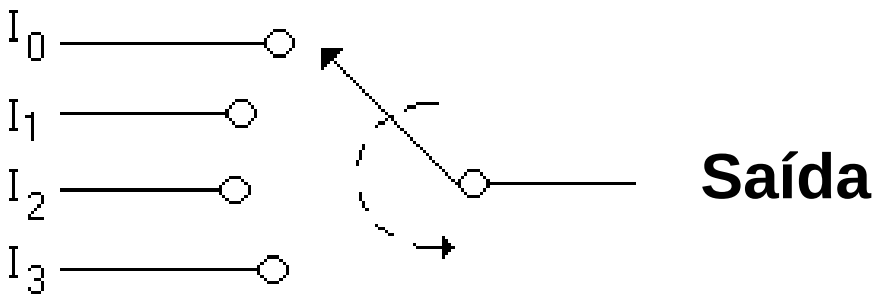
Adaptado por Glauber (2018)

Blocos básicos

- Codificadores e Decodificadores
- (De) Multiplexadores (HOJE!)
- Portas tri-state (HOJE?)
- Comparadores
- Somadores/Subtratores
- **16 de maio: P2**
- Multiplicadores
- ULA
- Gerador/Detector de Paridade

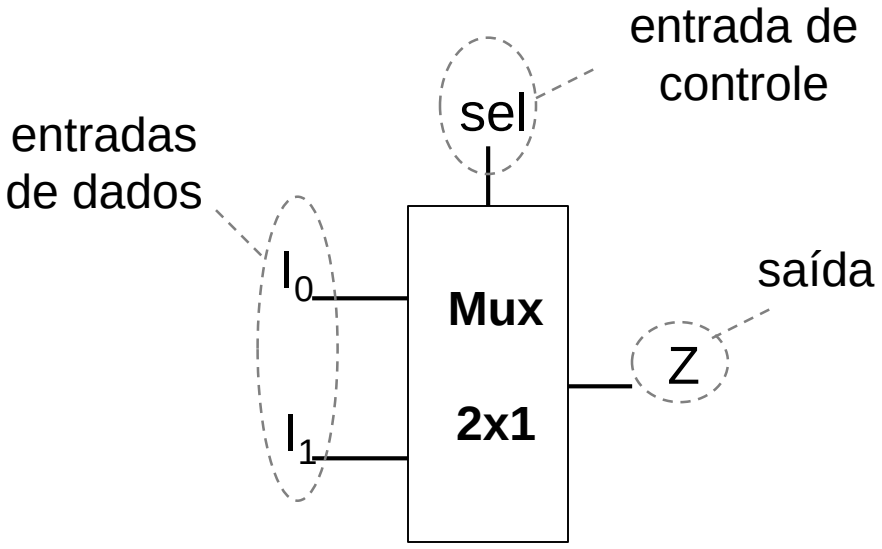
Multiplexador: Conceito

- Multiplexação: seleção da entrada que deve ir para a saída
  - Operação de uma chave seletora



Multiplexador: Conceito

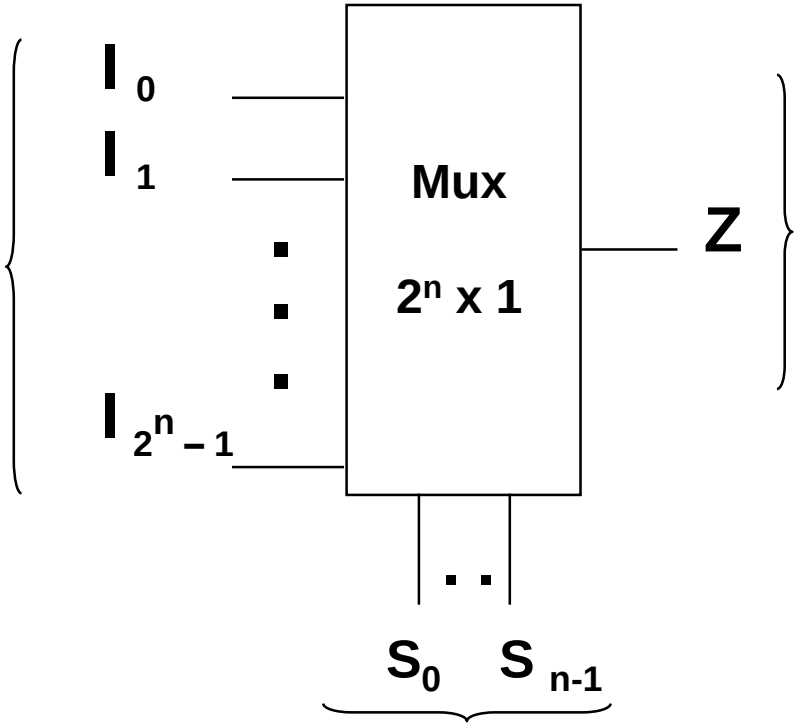
- Multiplexador ou Multiplexer ou Mux
  - Possui 2^n entradas de dados e uma saída: só uma das 2^n entradas é “conectada” à saída
  - Um total de **n entradas de seleção** indicam qual entrada de dados vai para a saída



sel	I_0	I_1	Z
0	**0**	0	**0**
0	**0**	1	**0**
0	**1**	0	**1**
0	**1**	1	**1**
1	0	**0**	**0**
1	0	**1**	**1**
1	1	**0**	**0**
1	1	**1**	**1**

Multiplexador: visão geral

2^n entradas de dados

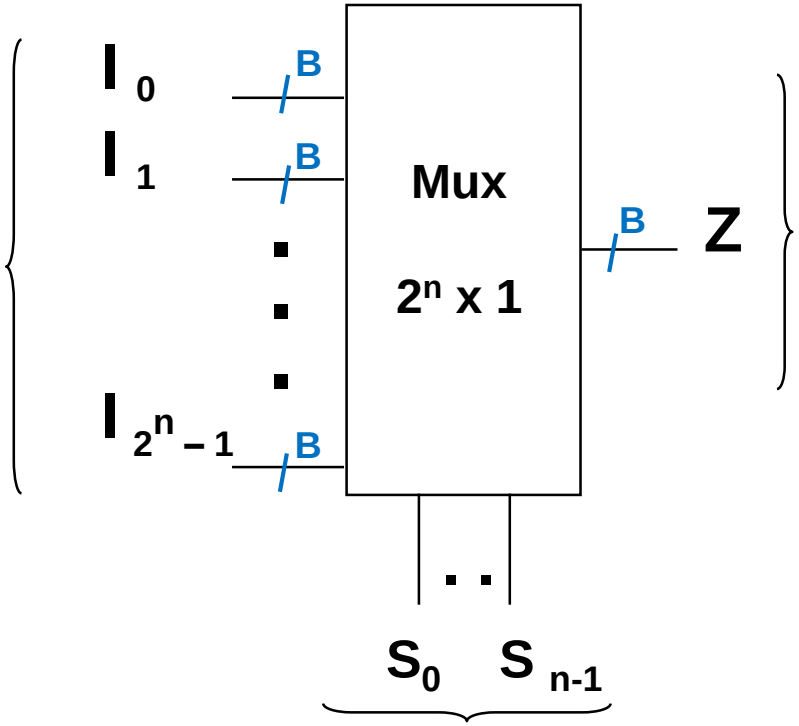


1 saída de dados

n entradas de seleção

Multiplexador: visão geral

2^n entradas de dados



1 saída de dados

n entradas de seleção

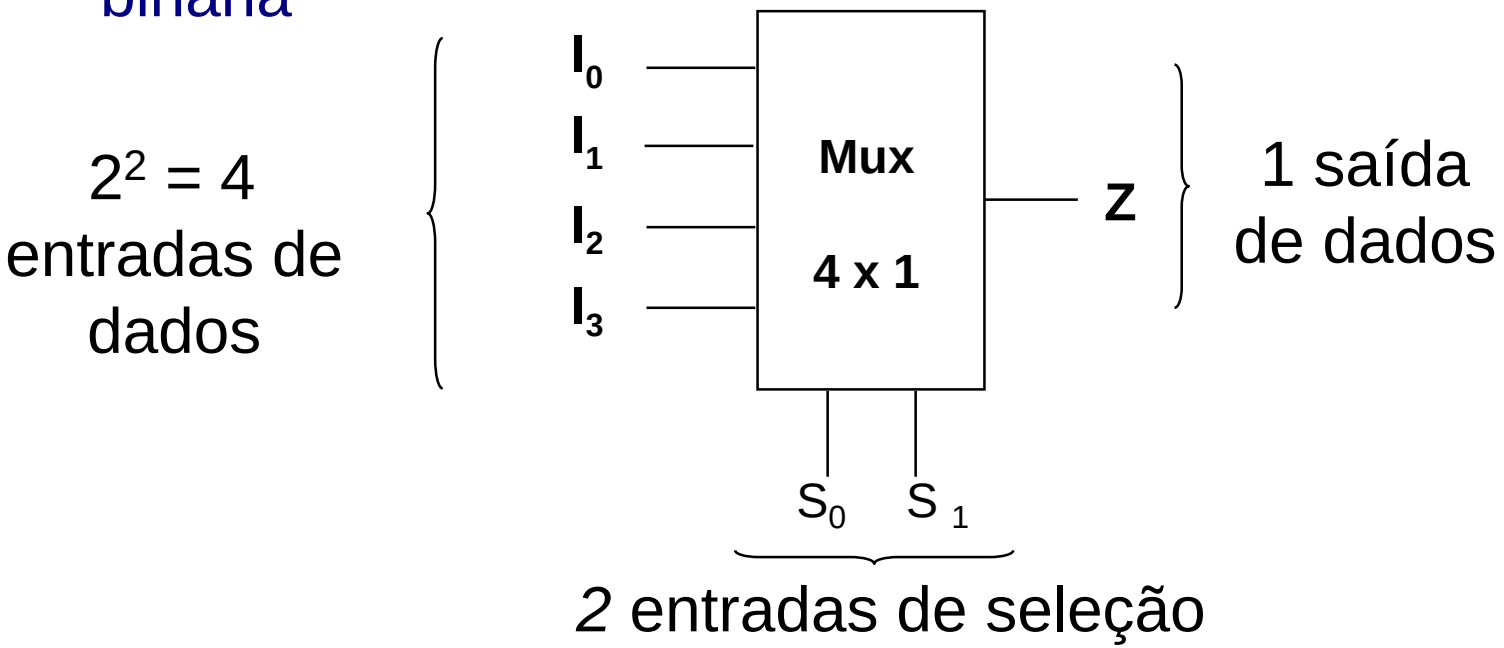
Nota: cada entrada/saída pode ser na realidade um barramento com B bits ➔ todos são selecionados ao mesmo tempo. NÃO vamos explorar este conceito nos slides a seguir

Multiplexador: visão geral

- 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

Multiplexador de 2 bits

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



Multiplexador de 2 bits

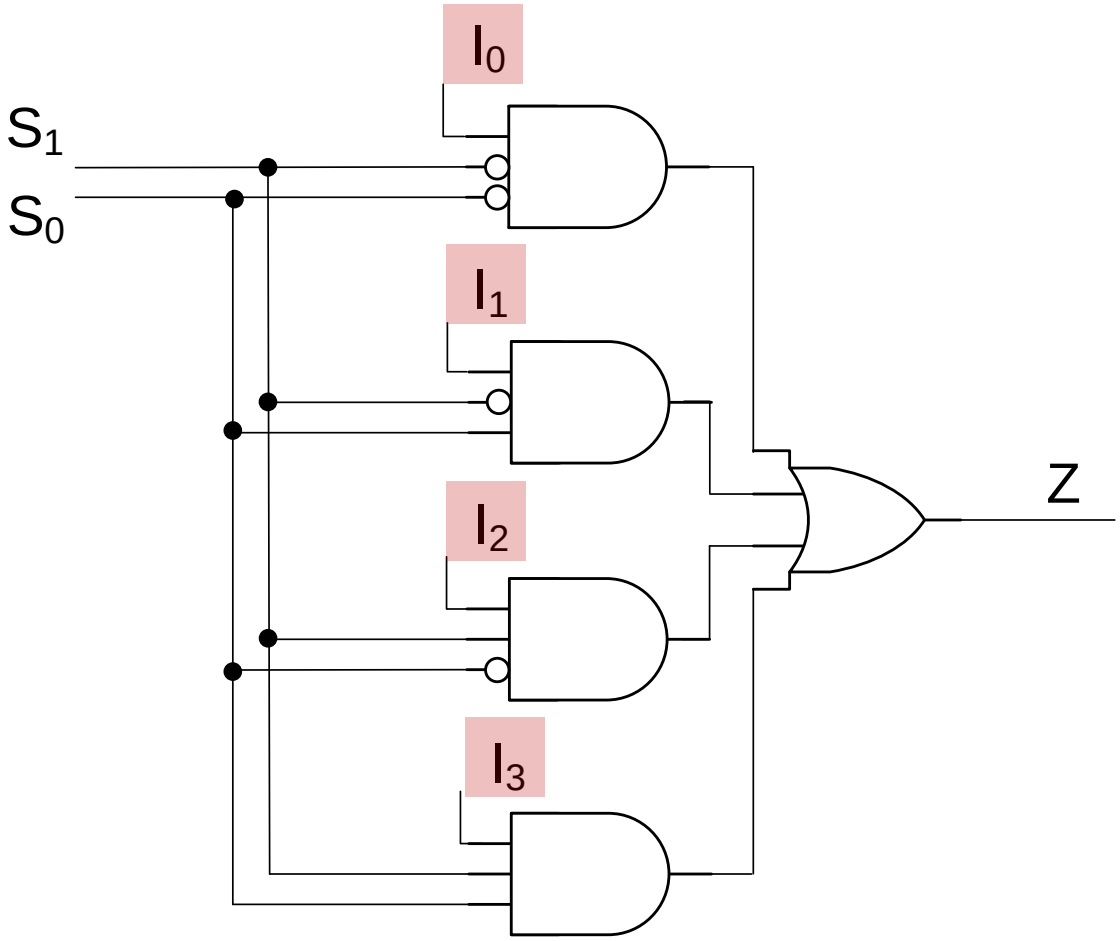
Ex: Mux 4 por 1

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

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

mintermos

Multiplexador de 2 bits: circuito interno



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

mintermos

Multiplexador com Enable

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

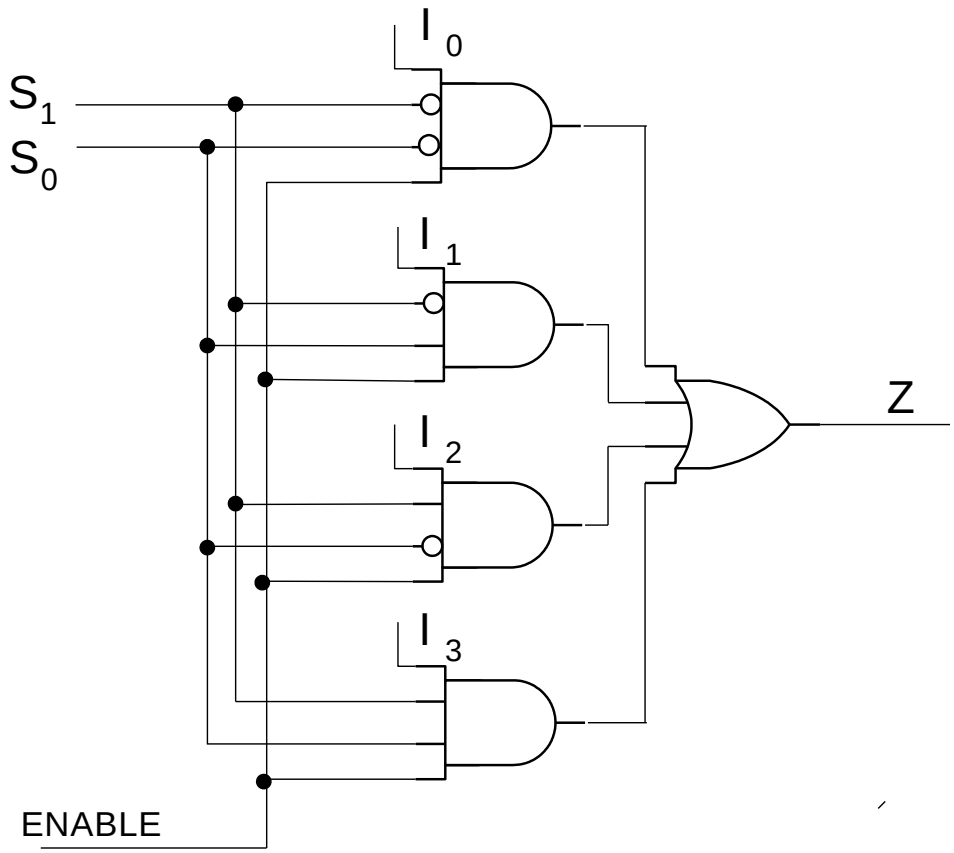
Multiplexador com Enable

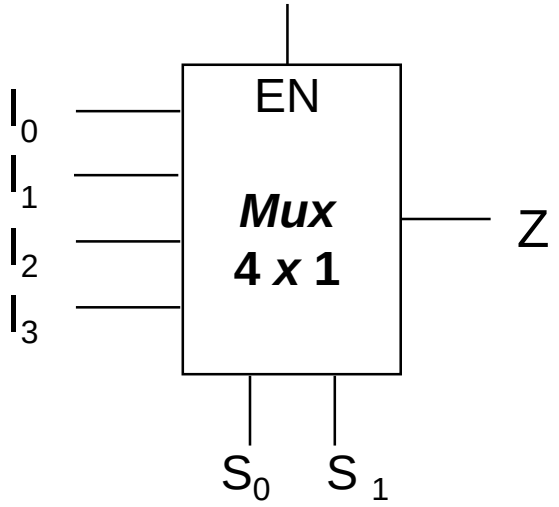
- Tabela verdade: Mux 4 por 1 com *Enable* (EN)

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

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

Multiplexador com Enable





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

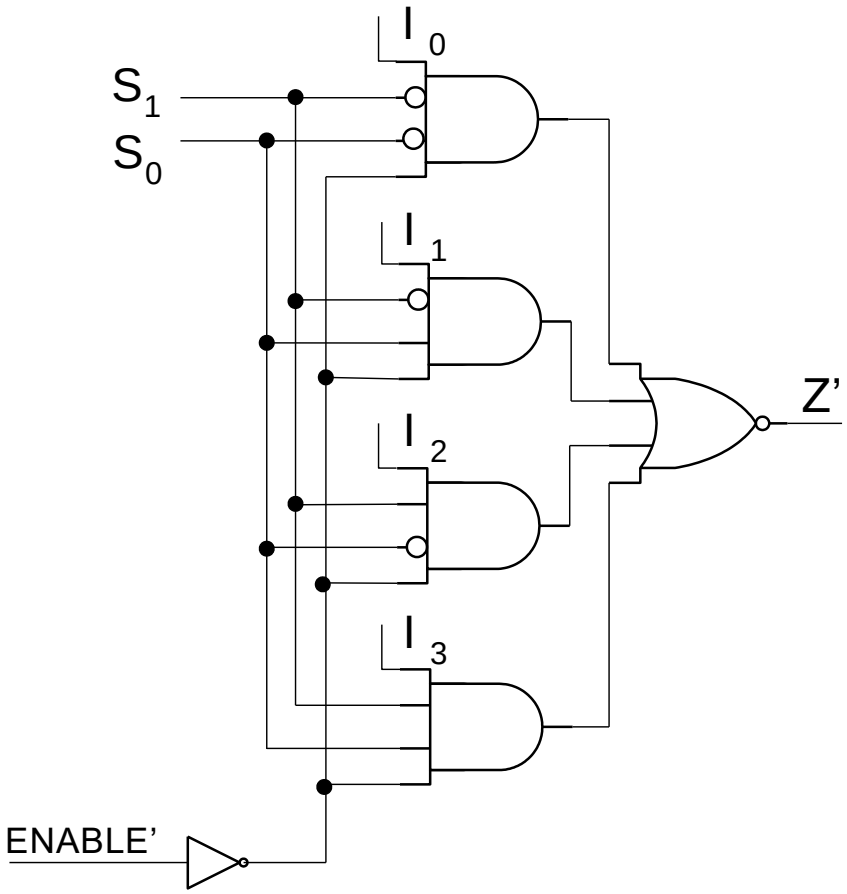
Multiplexador: *active low*

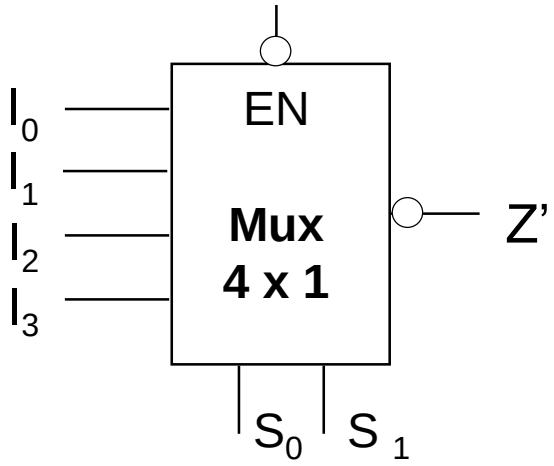
- Multiplexadores podem ter a saída invertida (complementada)
  - Diz-se que as saídas são *active-low* – ativas em ZERO
- Implicações:
  - Na tabela verdade: inversão nas saídas ($Z' = I_i'$)
  - No circuito: uso de **NOR** no lugar de OR
  - Nomenclatura: Z’

Multiplexador: *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'

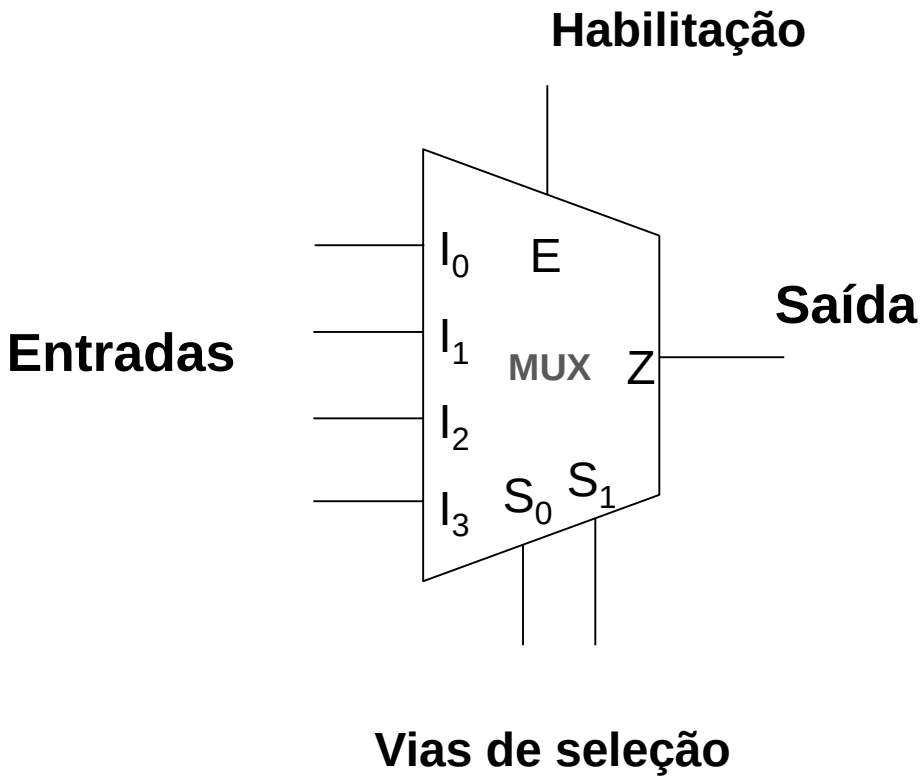
Multiplexador: *active-low*





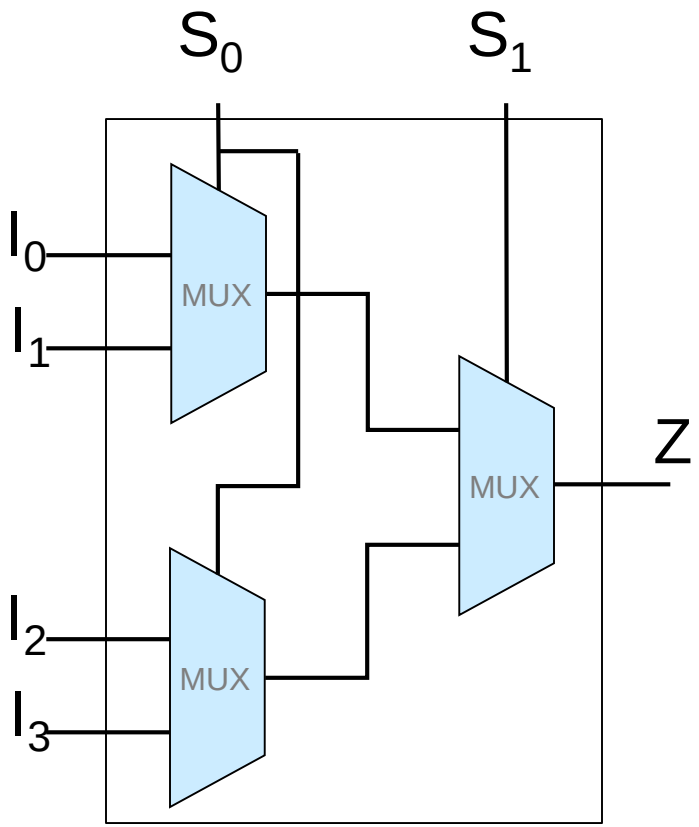
Entradas			Saída
EN'	S_1	S_0	Z'
0	0	0	I_0'
0	0	1	I_1'
0	1	0	I_2'
0	1	1	I_3'
1	X	X	1

Multiplexador: Símbolo alternativo



Multiplexador: Cascateamento

- Multiplexador 4:1 implementação usando Mux 2:1



S_1	S_0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Multiplexadore 4x1 : VHDL

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

entity mux4_1 is
    port (S           : in  STD_LOGIC_VECTOR (1 downto 0);
          A,B,C,D     : in  STD_LOGIC;
          Y           : out STD_LOGIC);
end mux4_1;

architecture arch_mux of mux4_1 is
begin
    with S select
       Y <= A  when "00",
            B  when "01",
            C  when "10",
            D  when "11",
            '0' when others; -- "catch all"
end arch_mux;
```

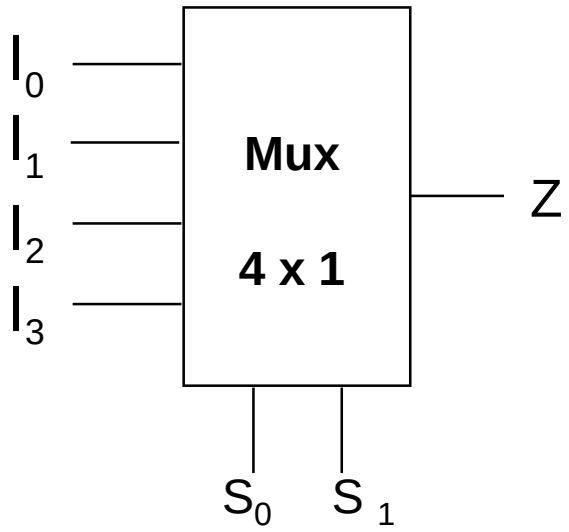
Multiplexadores multibit: VHDL

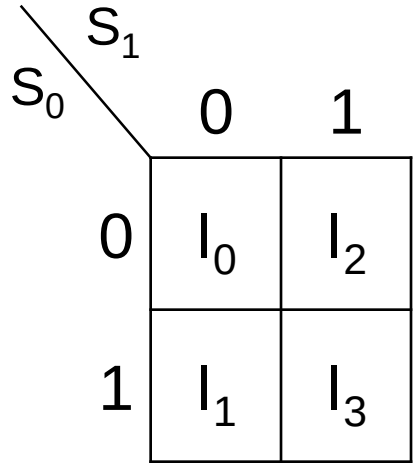
```
library IEEE;
use IEEE.std_logic_1164.all;

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;
```

Obs.: No código, entradas/saídas são barramentos de 8 bits

Multiplexadores: Uso

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



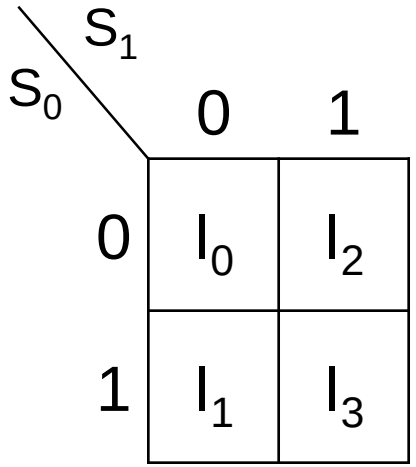


Mapa do Mux 4*x*1

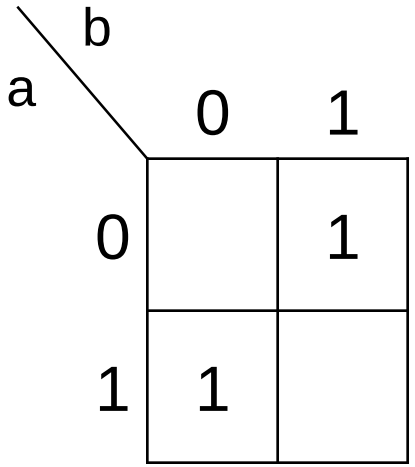
$$Z = S_1'.S_0'.I_0 + S_1'.S_0.I_1 + S_1.S_0'.I_2 + S_1.S_0.I_3$$

Multiplexadores: Síntese

Mapa do
Mux 4*x*1



Mapa da Função
F(b,a)

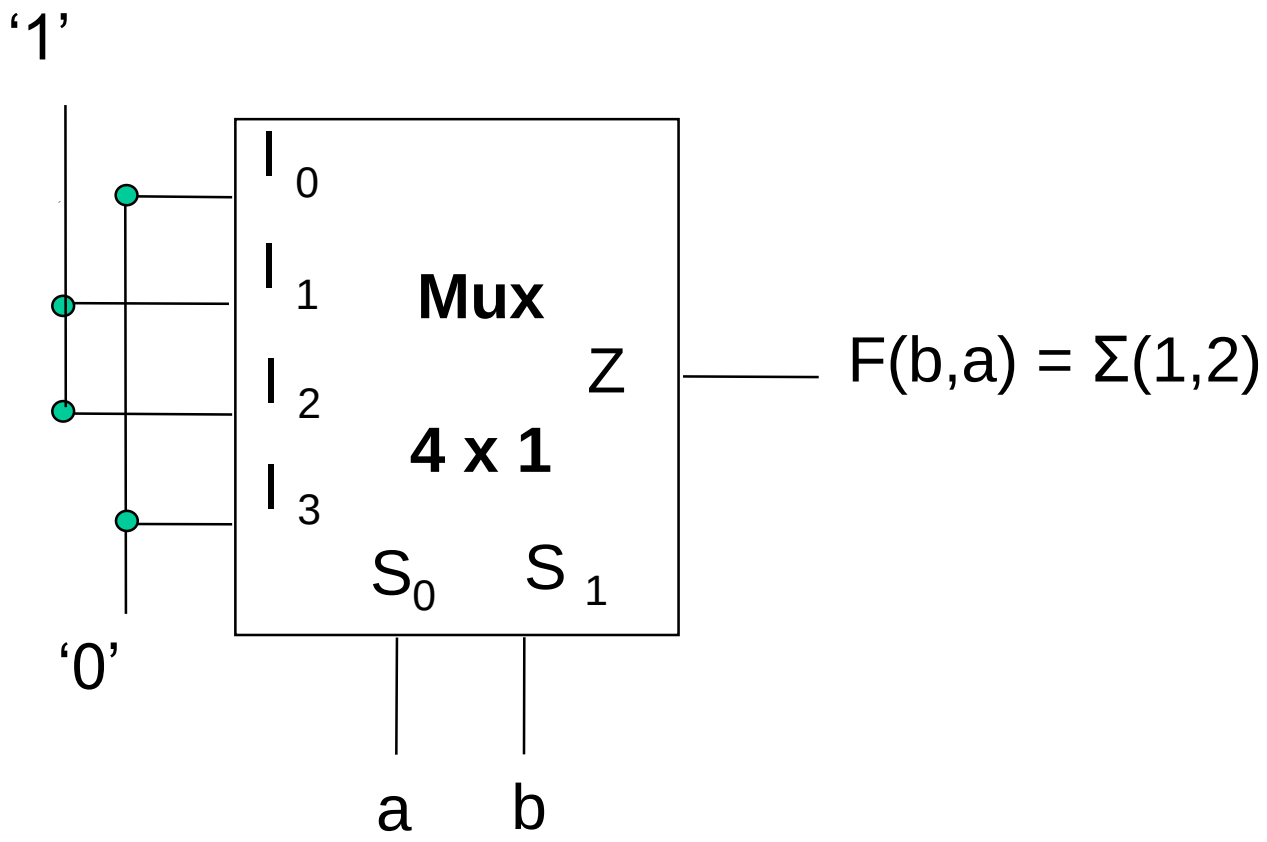


$F(b,a) = \Sigma(1,2)$

- Da comparação dos dois mapas obtém-se:

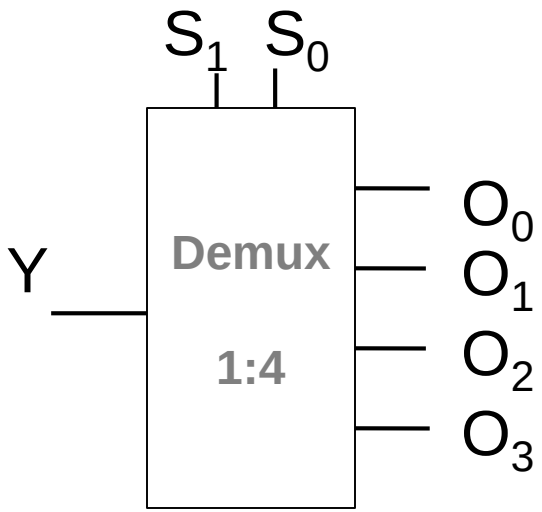
  $I_0 = 0, I_1 = 1, I_2 = 1, I_3 = 0$

Multiplexadores: Síntese



Demultiplexadores

- Função inversa ao multiplexador:
  - Demux 1:2^n direciona a entrada para uma dentre 2^n saídas de dados
  - Considerando Y=EN, tem-se um Decodificador 2x4!



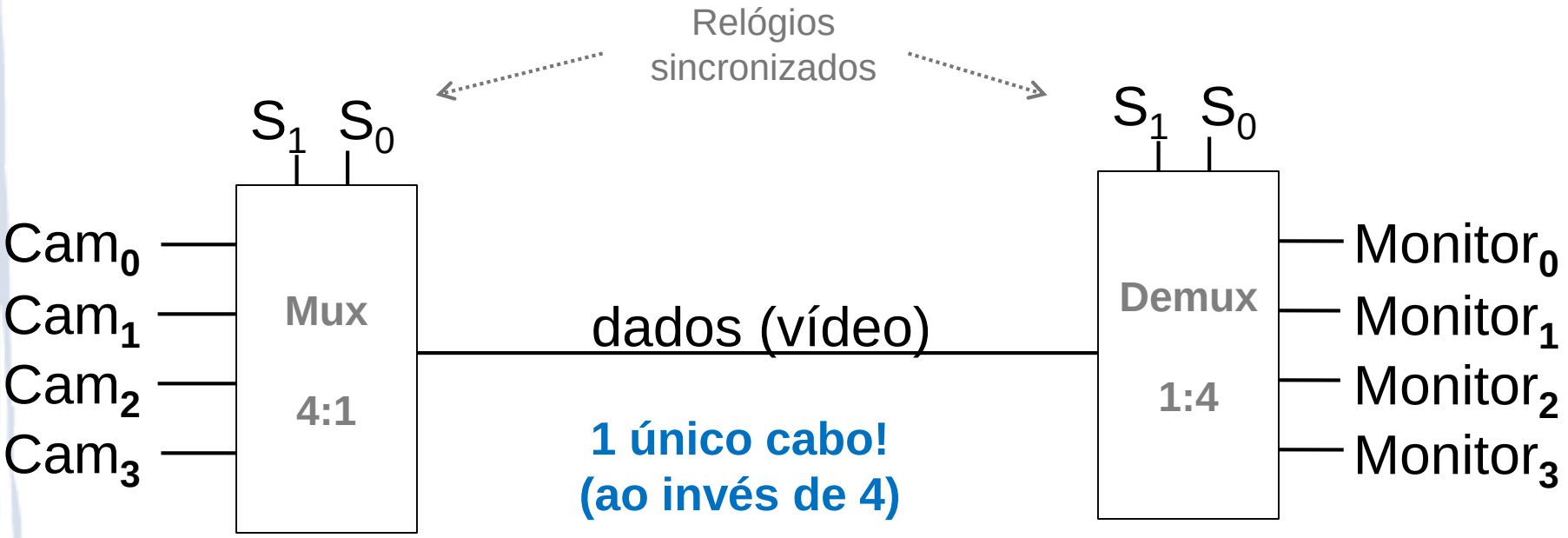
S_1	S_0	O_1	O_0	O_2	O_3
0	0	**Y**	0	0	0
0	1	0	**Y**	0	0
1	0	0	0	**Y**	0
1	1	0	0	0	**Y**

Demultiplexador 4x1, 8 bits: VHDL

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

entity demux4_1 is
    port (S              : in STD_LOGIC_VECTOR (1 downto 0)
          Y              : in STD_LOGIC_VECTOR (7 downto 0);
          O0,O1,O2,O3    : out STD_LOGIC_VECTOR (7 downto 0));
end demux4_1;

architecture arch_demux0 of demux4_1 is -- desabilitado com 0s
begin
    O0 <= Y when S = "00" else "00000000"; -- código alternativo:
    O1 <= Y when S = "01" else "00000000"; -- else (others => '0');
    O2 <= Y when S = "10" else "00000000";
    O3 <= Y when S = "11" else "00000000";
end arch_demux0;


architecture arch_demuxZ of demux4_1 is -- com alta impedância
begin
    O0 <= Y when S = "00" else "ZZZZZZZZ"; -- código alternativo:
    O1 <= Y when S = "01" else "ZZZZZZZZ"; -- else (others => 'Z');
    O2 <= Y when S = "10" else "ZZZZZZZZ";
    O3 <= Y when S = "11" else "ZZZZZZZZ";
end arch_demuxZ;
```

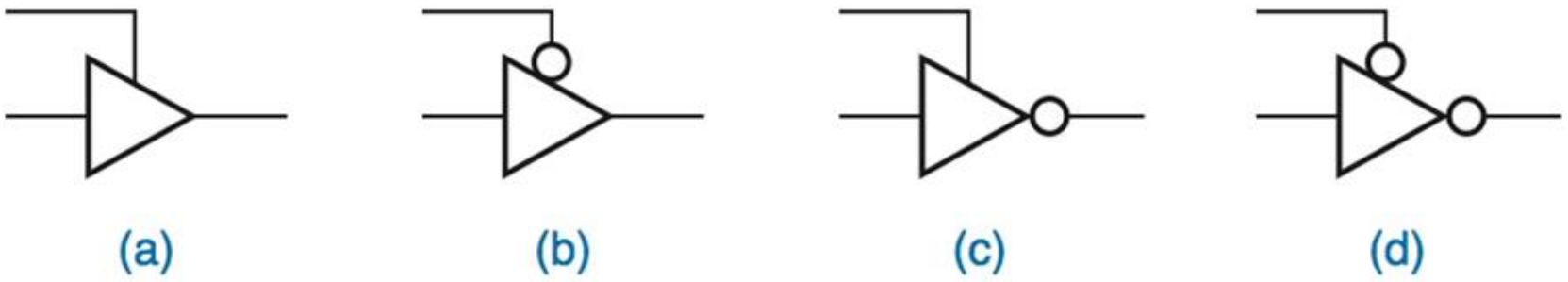
(De)multiplexadores: Uso

- Pode ser usado para compartilhamento de canal entre diversos fluxos (multiplexação de vídeo)
  - Ex.: envio serial de vídeo de diferentes câmeras para uma central de vigilância



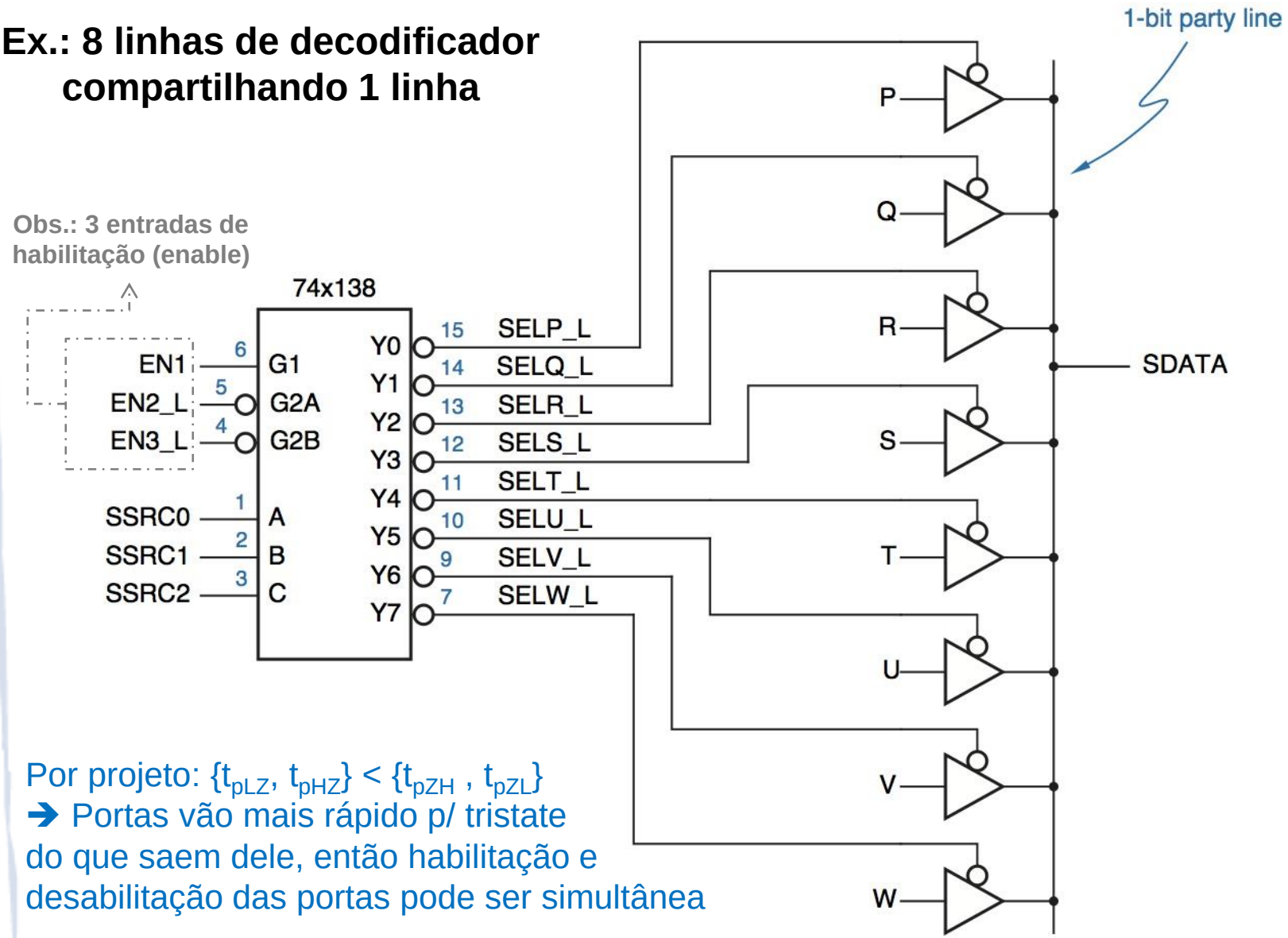
Portas tristate

- Além de saída em 1 e 0, pode-se ter um terceiro estado de alta impedância (HI-Z).
- Usado para conexão em barramentos: apenas circuitos ativos acessam barramento, prevenindo curtos
  a) Buffer (sem inversão): ativo alto
  b) Buffer (sem inversão): ativo-baixo
  c) Inversor tristate: ativo-alto
  d) Inversor tristate: ativo-baixo



Portas tristate

Ex.: 8 linhas de decodificador compartilhando 1 linha

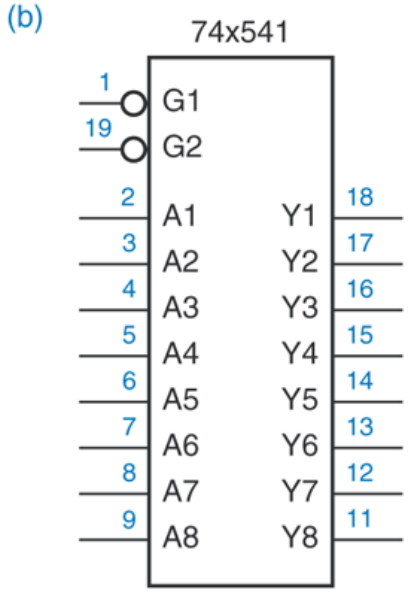


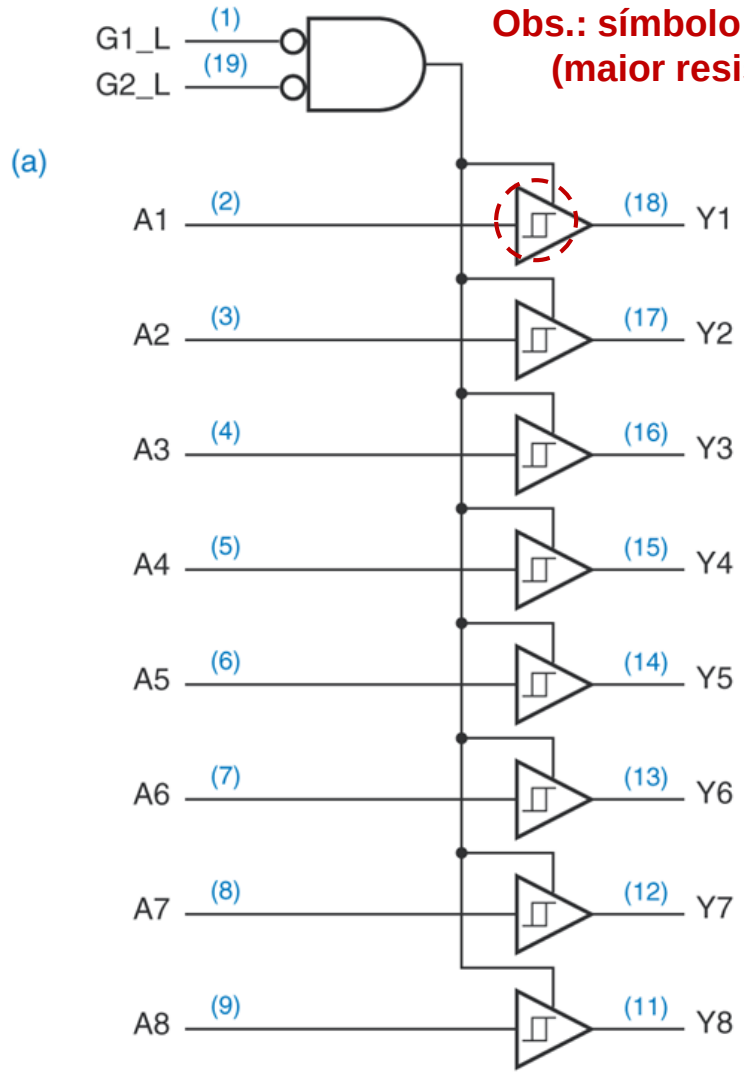
Por projeto: $\{t_{pLZ}, t_{pHZ}\} < \{t_{pZH}, t_{pZL}\}$

➔ Portas vão mais rápido p/ tristate do que saem dele, então habilitação e desabilitação das portas pode ser simultânea

Portas tristate

Circuito tristate comercial: para barramentos de 8 bits

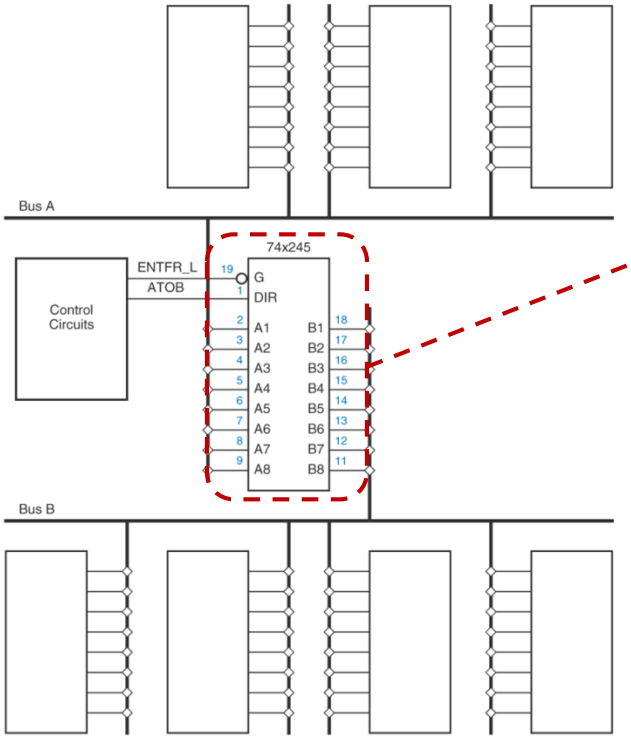


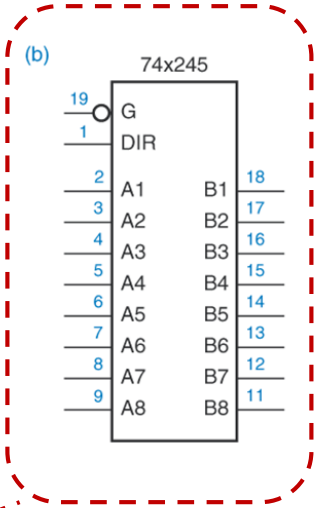


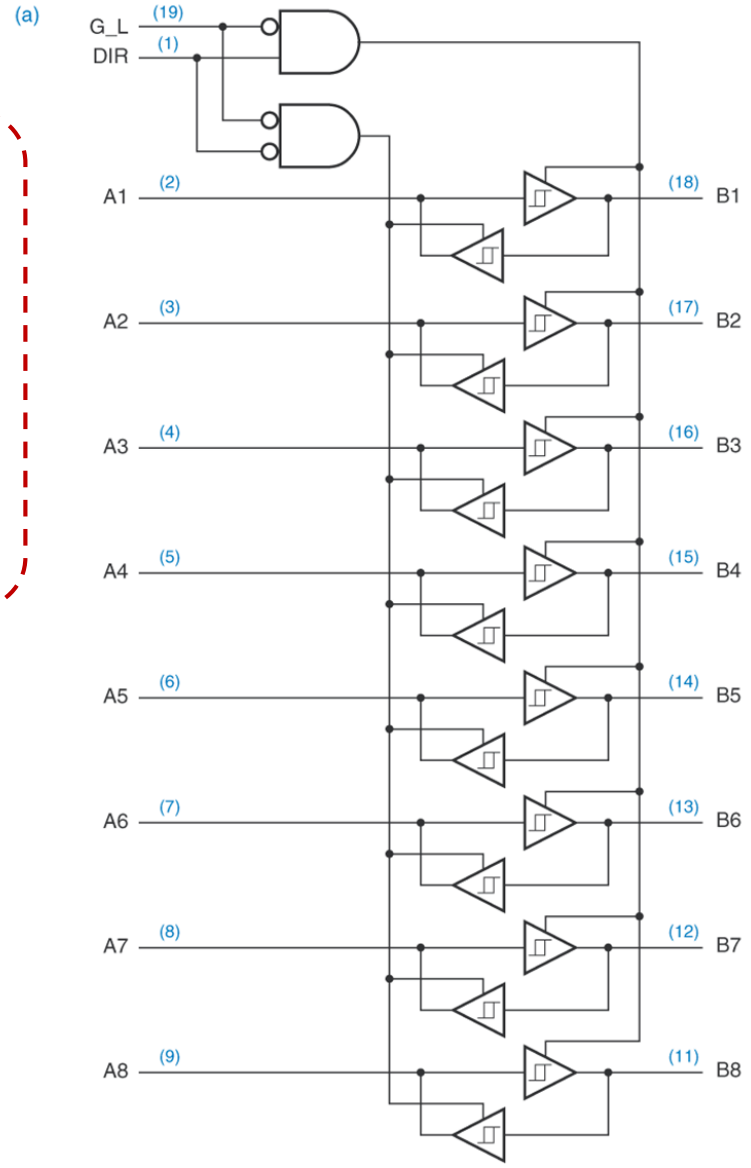
Obs.: símbolo significa histerese (maior resistência a ruído)

Portas tristate

Circuito tristate comercial para conexão de barramentos de 8 bits: direção do fluxo de dados definida por entrada “DIR”

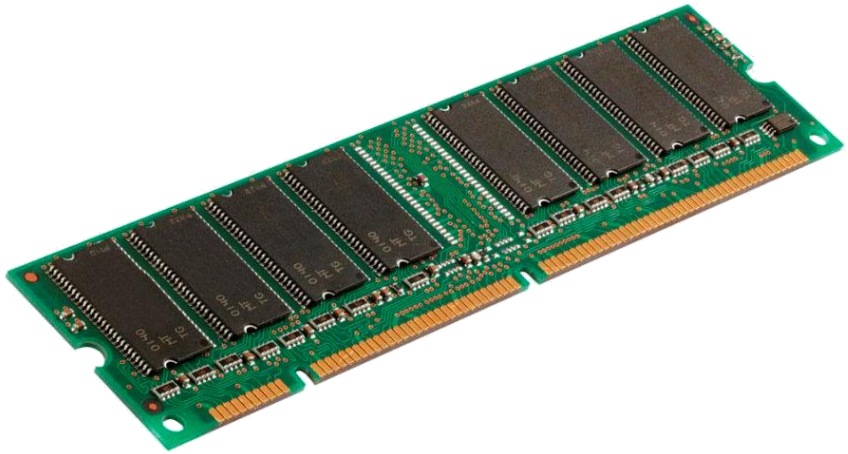






Portas tristate

- Uso comum em computadores modernos: **circuitos de memória** compartilhando o mesmo barramento
  - Apenas um chip de memória pode **escrever** no barramento a qualquer momento
  - Os outros podem **ler** do barramento em paralelo
  - ➔ Não aumenta “capacidade de transmissão”, mas permite uso de menos fios para comunicação



Lição de Casa

- Leitura sugeridos:
  - Capítulo 6 do Livro Texto, ênfase em 6.7, 6.8.
- Exercícios sugeridos:
  - Capítulo 6 do Livro Texto – (de)mux e tri-state.

Multiplexadores: Exercício

- Usando apenas um Mux 8x1, implemente a função de 4 variáveis abaixo.
- Considere que, além das entradas A,B,C e D, temos acesso ao 1 (Vcc) e ao 0 (GND).
- Como lógica extra, além do Mux 8x1, uma porta inversora é permitida.

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

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

Multiplexadores: Síntese

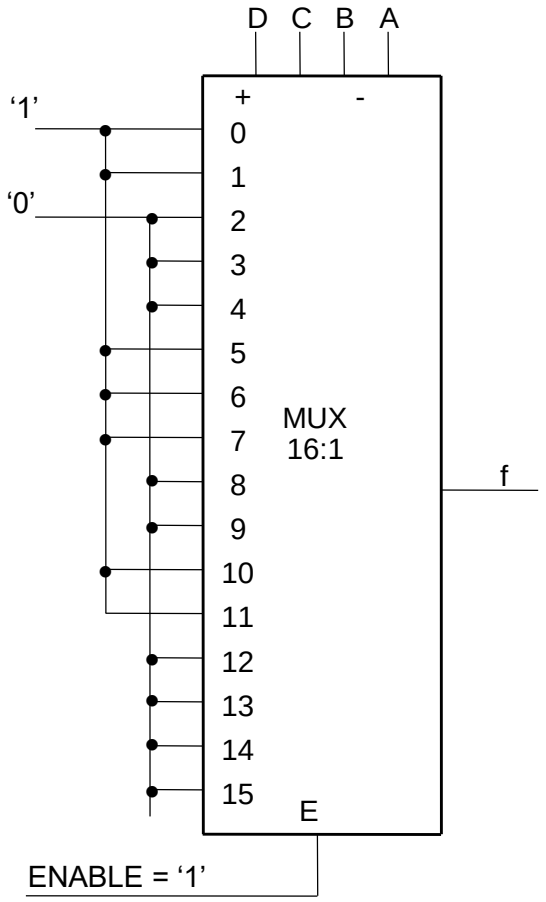
- Solução simples: com Mux de 16x1

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

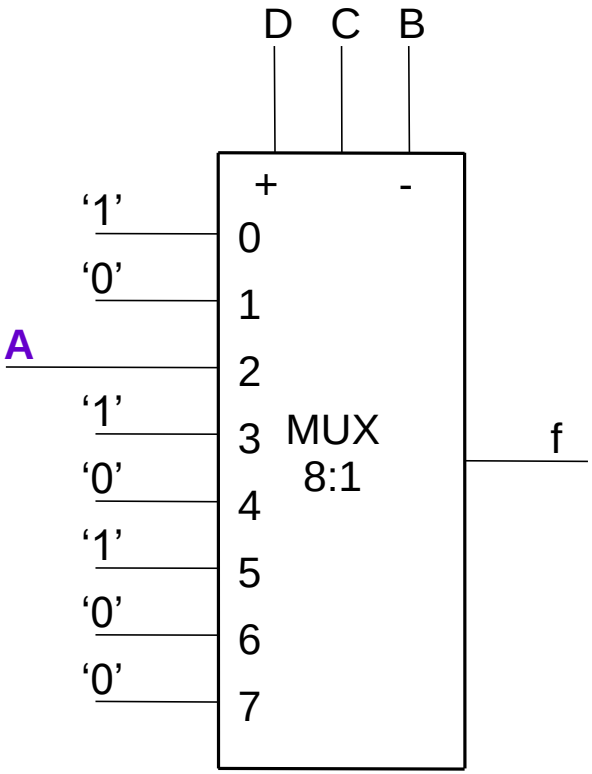
BA \ DC	00	01	11	10
00	1			
01	1	1		
11		1		1
10		1		1

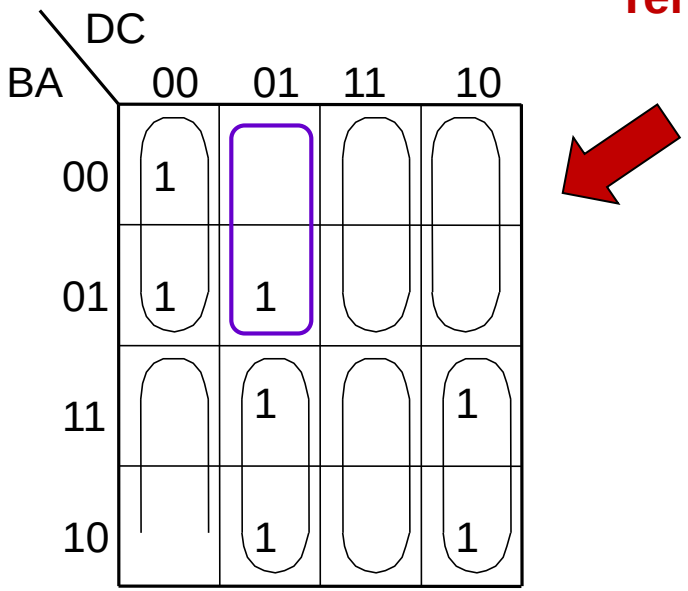
$F = \Sigma(0,1,5,6,7,10,11)$



Multiplexadores: Síntese

- Solução 1: Mux de 8x1 e **A na entrada**





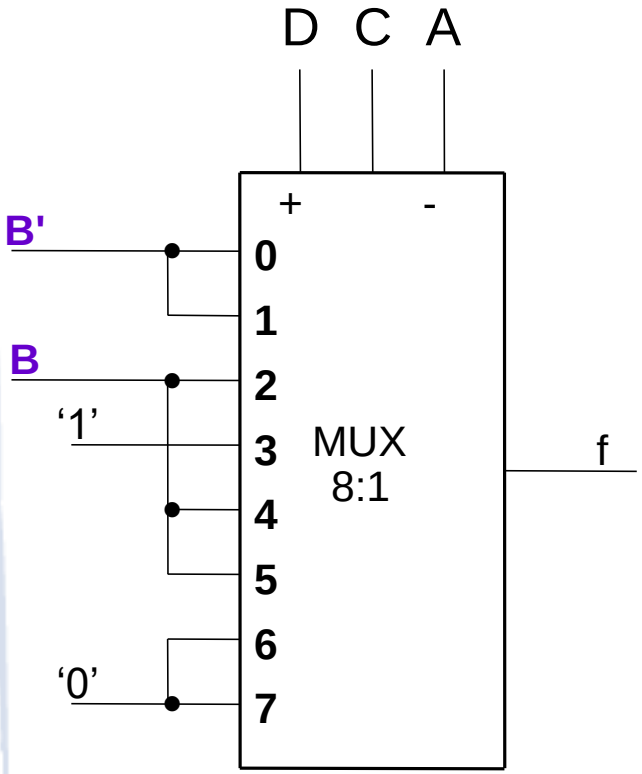
Termos sem A

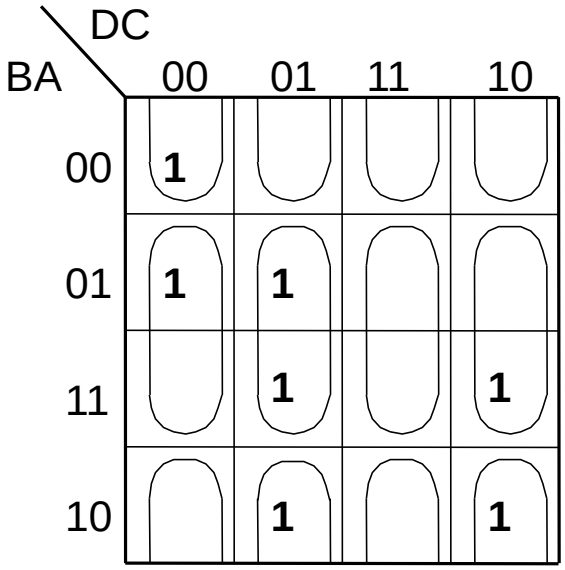
2. Função f: D'C'B' + D'CB'**A** + D'CB + DC'B

0 **2** **3** **5**

Multiplexadores: Síntese

- Solução 2: Mux de 8x1 e **B na entrada**





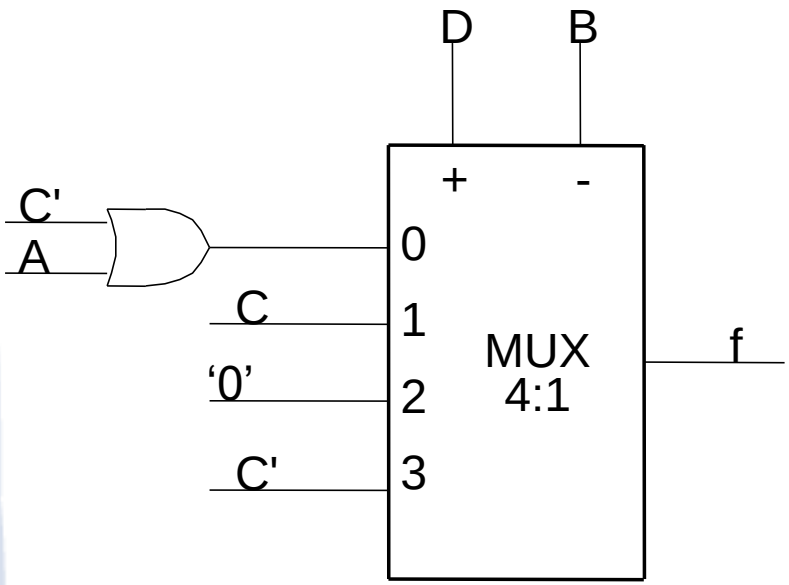
Termos sem B

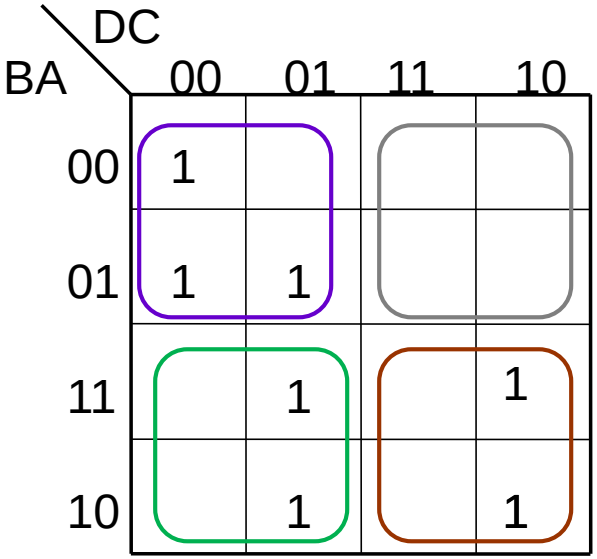
2. Função f = D'C'A'**B'** + D'C'A**B'** + D'CA'**B** + D'CA + DC'A'**B** + DC'A**B**

0 1 2 3 4 5

Multiplexadores: Síntese

- Se temos apenas um Mux de 4x1, precisamos de portas extras





Termos sem A ou C

2. Função f = **D'B'**(A+C') + **D'B**C + **DB**C'

Exercícios

- 6.1. Sintetize a função de chaveamento abaixo utilizando

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

a. Um Mux de 16x1

b. Um Mux de 8x1

c. Um Mux de 4x1

- 6.2. Sintetize a função de chaveamento dada pela tabela verdade ao lado utilizando um decodificador binário 3:8.

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