# Sistemas Digitais

## Circuitos Combinatórios Típicos

João Paulo Baptista de Carvalho

joao.carvalho@inesc.pt


# Descodificadores

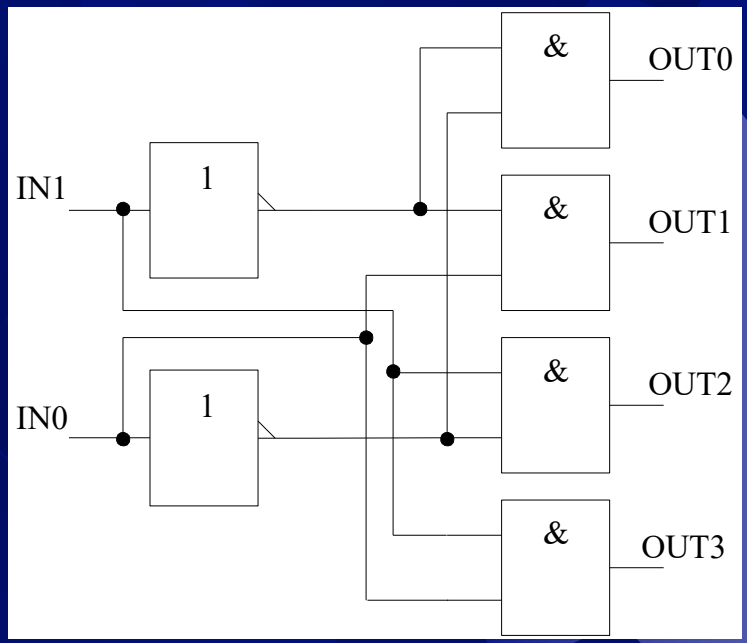
- Imaginem 4 linhas de comboio que convergem numa só.
- Cada uma delas tem um semáforo que pode estar a vermelho ou a verde.
- Existe um funcionário que em cada instante indica qual a linha que deve estar a verde.
- Para não haver colisões é preciso garantir que em cada instante há só um único semáforo a verde.
- Como projectar um circuito que garanta isso?

| IN 1 | IN 0 | Out 0 | Out 1 | Out 2 | Out 3 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

# Descodificadores

- Um descodificador é um circuito combinatório que permite, perante uma combinação de entradas, activar uma e só uma saída.
- Descodificador 2:4 (2 entradas, 4 saídas)

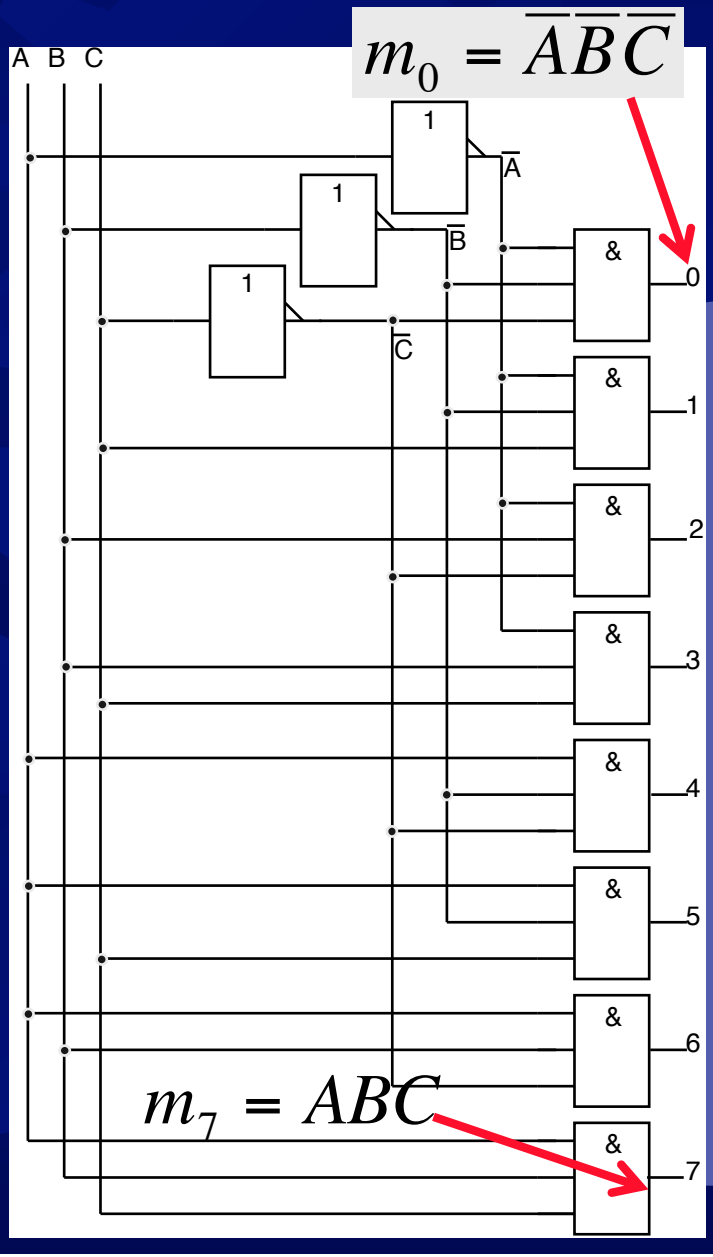
| IN1 IN0 | Out0 | Out1 | Out2 | Out3 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 |



# Descodificadores (II)

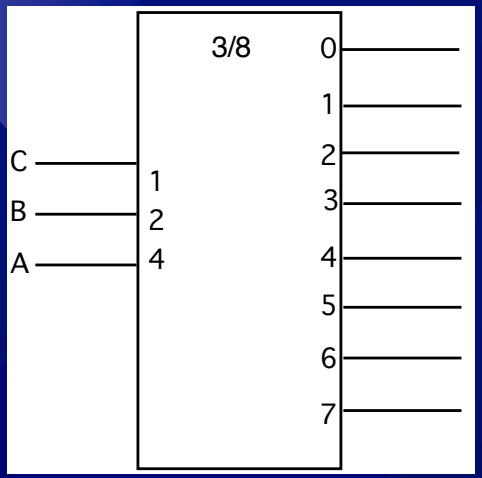
- Um descodificador binário de 3 entradas (3:8)

| ABC | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|---|---|---|---|---|---|---|---|---|
| 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 010 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

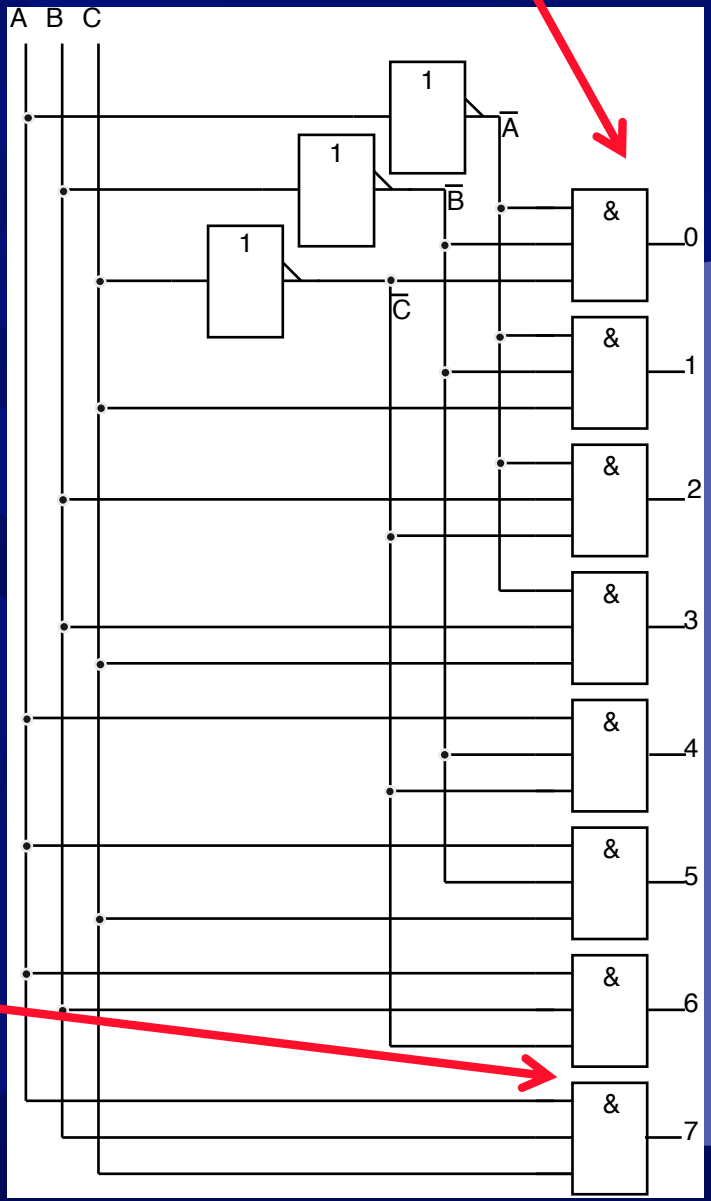


# Descodificadores (III)

- Estrutura interna de um descodificador binário de 3 entradas




$m_0 = \overline{A}\,\overline{B}\,\overline{C}$

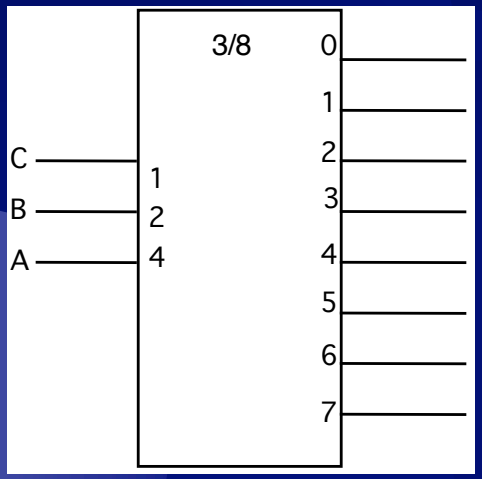



$m_7 = ABC$

# Descodificadores (IV) - Exemplos

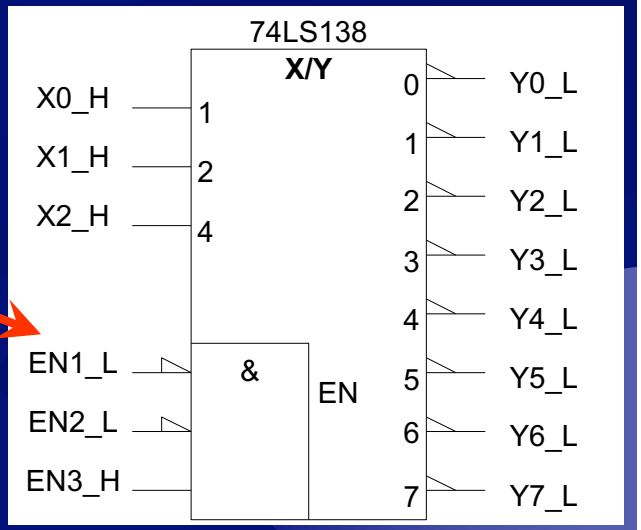
## Simbologia

### Descodificador 3:8

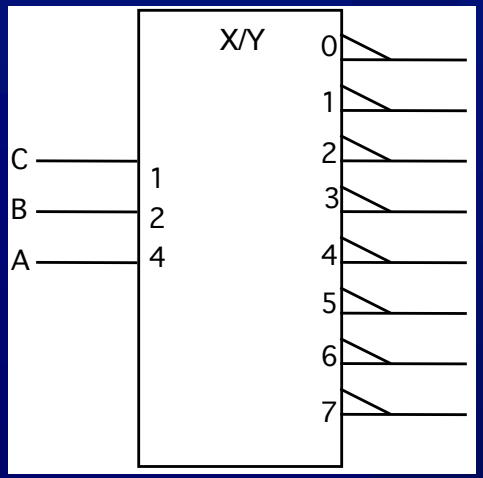



**Enable: entrada(s) que tem que estar activa(s) para que o descodificador funcione**

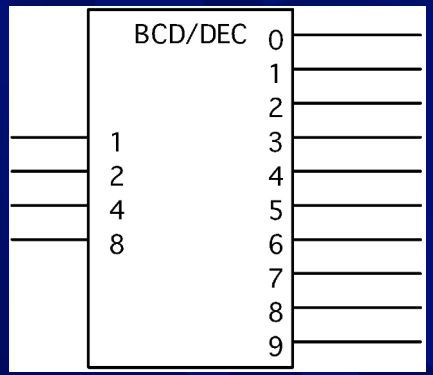
### Descodificador 3:8



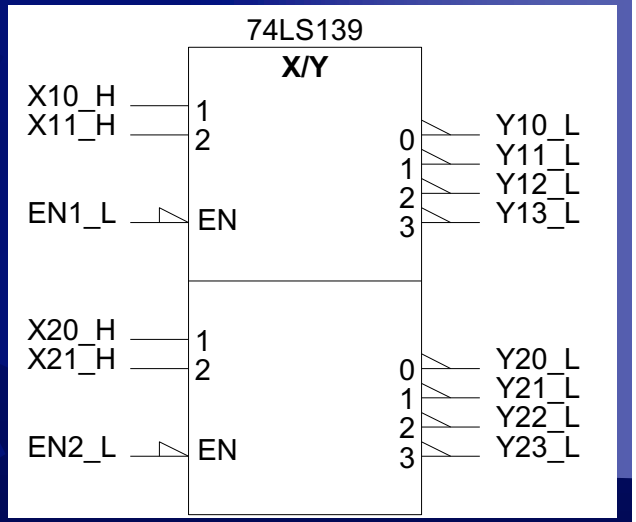

### Descodificador 3:8 com saídas activas a LOW (construído a partir de NANDs)




### Descodificador BCD/Decimal

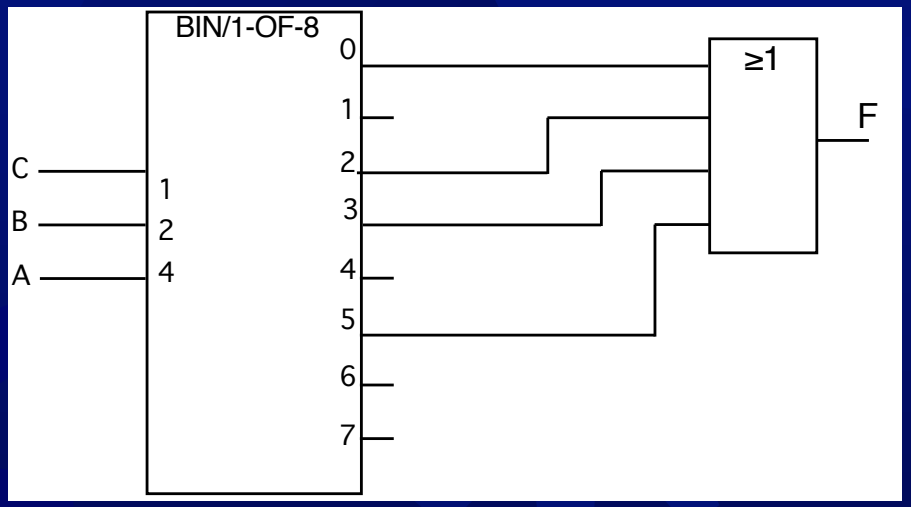



### Dual Decoder 2:4



# Descodificadores (V)

- Implementação de funções utilizando descodificadores
  - Cada saída do descodificador corresponde a um mintermo das variáveis de entrada
  - Como cada função pode ser expressa sob a forma de uma soma de mintermos, a implementação é directa.
  - Exemplo: $f(A,B,C)=\Sigma m(0,2,3,5)$

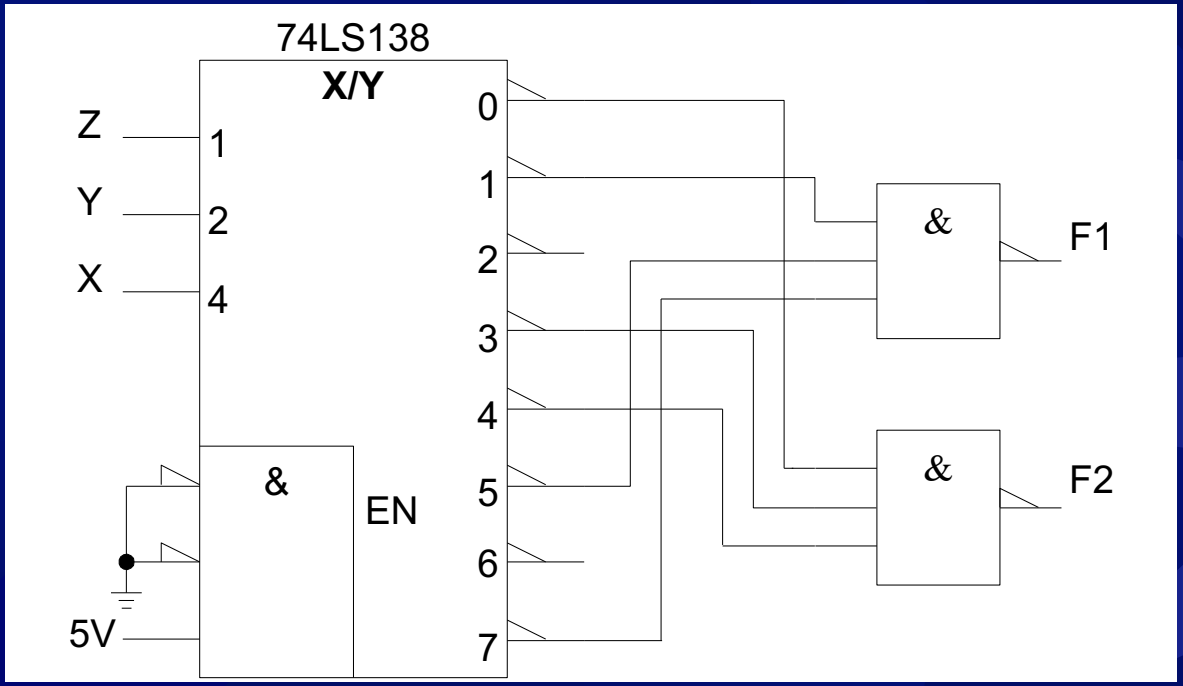


# Descodificadores (VI)

- Implementação de mais do que uma função utilizando descodificadores. Exemplo:

$$f_1(X,Y,Z)=\Sigma m(1,5,7)$$

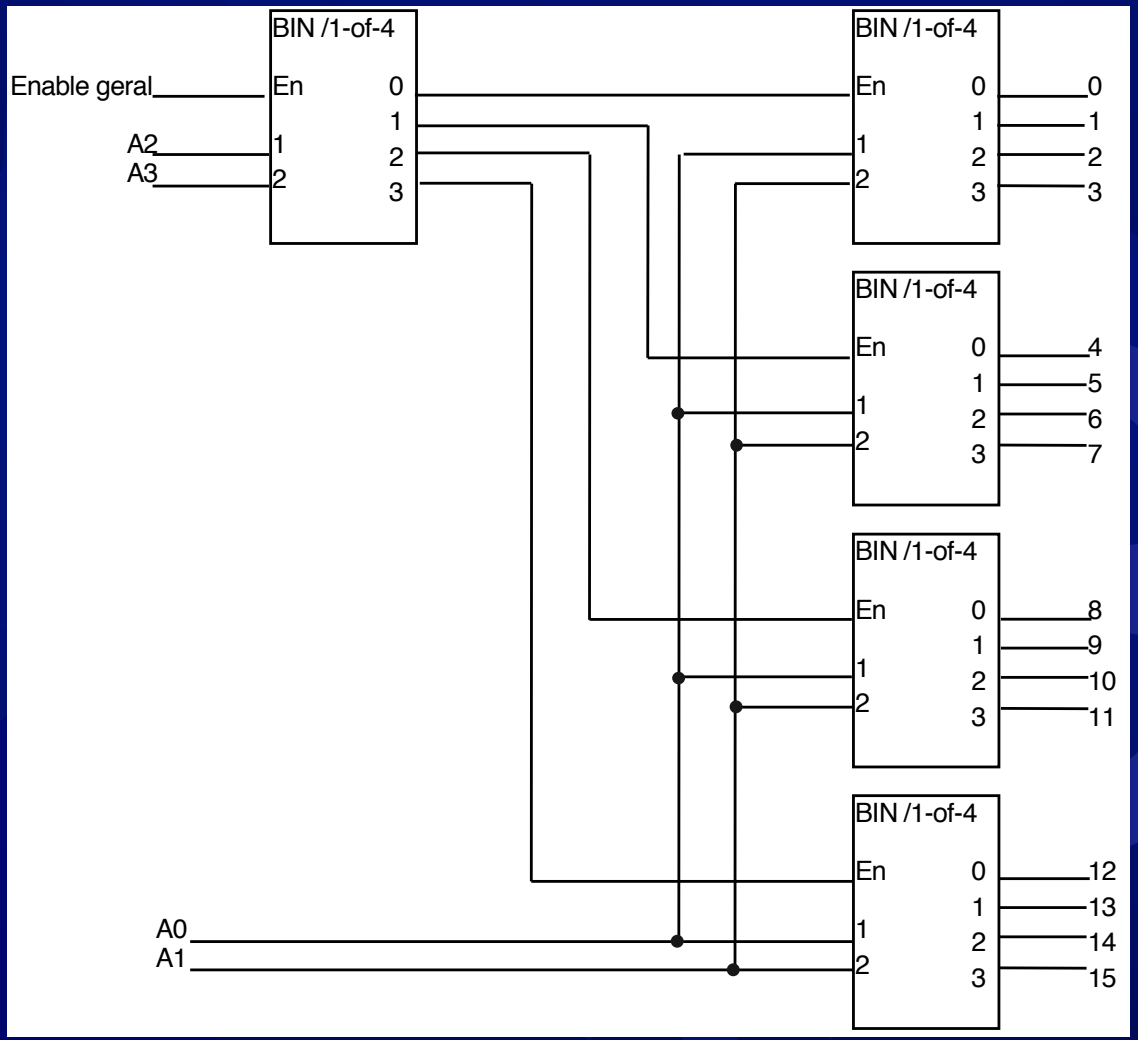
$$f_2(X,Y,Z)=\Sigma m(0,3,4)$$




- Q: Porque motivo se utiliza um NAND, se o que se pretende implementar é uma soma (OR)?

# Descodificadores (VII)

- Obtenção de um descodificador 4:16 a partir de descodificadores 2:4 com enable

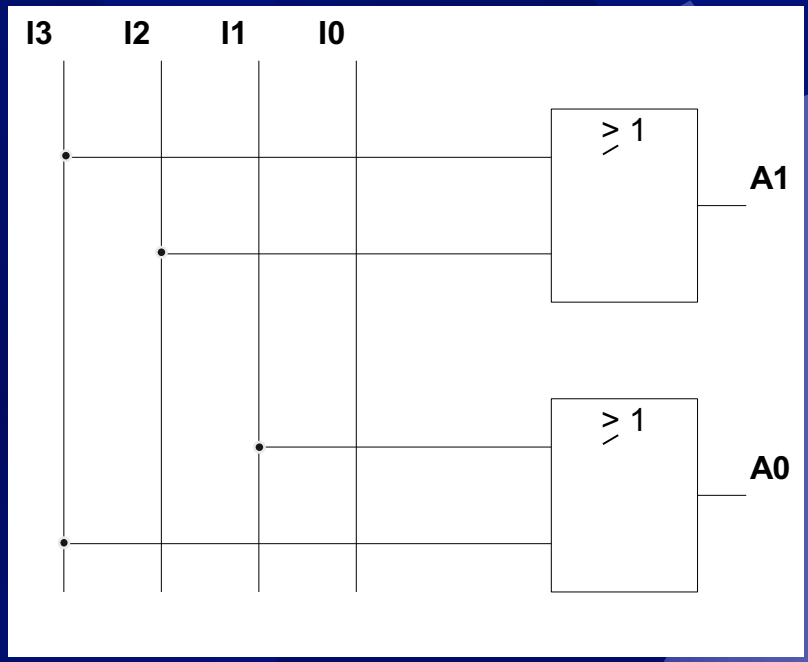


# Codificadores

- O codificador é um circuito combinatório que permite, perante a activação de uma entrada, gerar a palavra de código correspondente a essa entrada (codificação). Um codificador de palavras de *n* bits tem $2^n$ entradas (correspondentes ao número de palavras) e *n* saídas)

**CODIFICADOR 4:2**

| I0 | I1 | I2 | I3 | A1 | A0 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |




- Codificadores de Prioridade: Permitem a activação de mais de uma entrada simultaneamente gerando a codificação correspondente à entrada activa mais prioritária

# Codificador de Prioridade

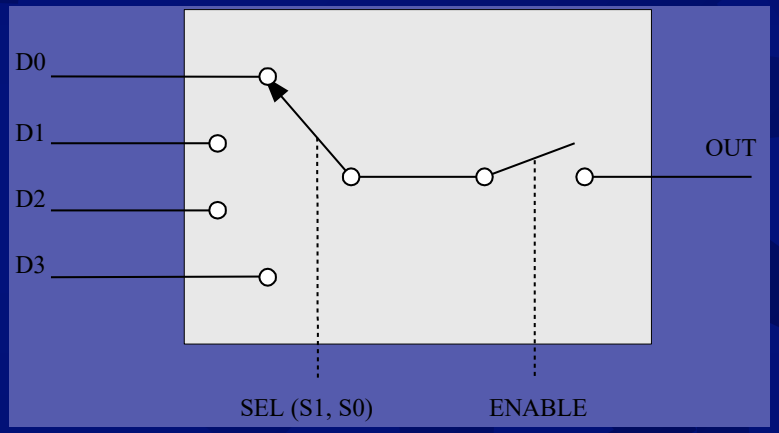
- Codificadores de Prioridade: Permitem a activação de mais de uma entrada simultaneamente gerando a codificação correspondente à entrada activa mais prioritária

| I3 | I2 | I1 | I0 | A1 | A0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | x | x |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

- Por inspecção visual (método de Olhovsky):
  - $A1 = I3 + I2$
  - $A0 = I3 + \overline{I2} I1$

# Multiplexeres

- Uma fábrica tem 4 depósitos de combustível. Em cada depósito existe um sensor que indica o estado do nível de combustível:1 se o combustível está acima de um certo nível, e 0 se estiver abaixo.
- Para saber se cada depósito está acima ou abaixo do nível, um operador dispõe de um circuito com botões de selecção que permite ver qual o estado do nível no depósito seleccionado.

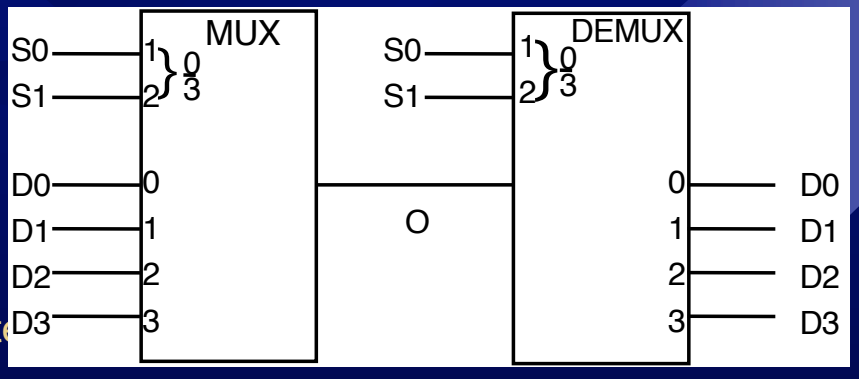



| S1 | S0 | OUT |
|---|---|---|
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |

- A um circuito deste tipo chama-se Multiplexer.
- Mas não seria melhor ter os 4 indicadores visíveis?

# Multiplexeres (II)

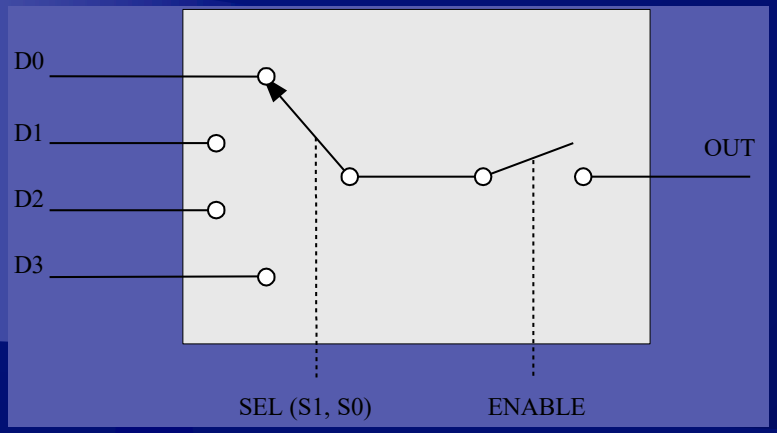
- Mas não seria melhor ter os 4 indicadores visíveis?
- A resposta é sim... Mas imaginem a seguinte situação:
  - A fábrica tem 64 depósitos que estão situados a 1Km da cabine de controlo
  - Para se ter a indicação dos níveis dos 64 depósitos necessitaria de 64 cabos com 1Km cada (i.e, 64Km de cabos de cobre)
  - Se utilizasse um circuito multiplexer, necessitava apenas de 7Km de cabos:
    - 6 cabos de 1 Km para os selectores (64=$2^6$)
    - 1 cabo de 1 Km para o indicador
  - Como o cobre é caro, existe uma poupança significativa...
- E se quiséssemos a conveniência de ter os 64 indicadores sempre visíveis poderíamos utilizar um demultiplexer na cabine do operador:
  - (falaremos sobre isto mais à frente)



# Multiplexeres (III)

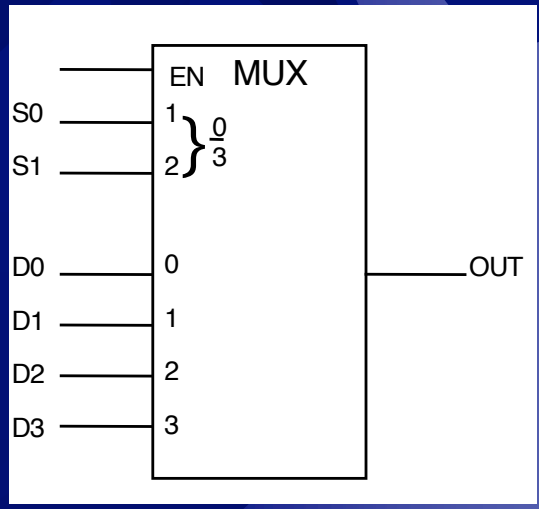
- O Multiplexer é um circuito combinatório que permite encaminhar uma de *n* entradas de dados para a saída.
- A entrada a encaminhar é especificada através de sinais de selecção (que funcionam como um descodificador)

**MULTIPLEXER 4:1**




**Entradas de Dados** **Entradas de Controlo** **(Selecção e Habilitação)** **Saída**

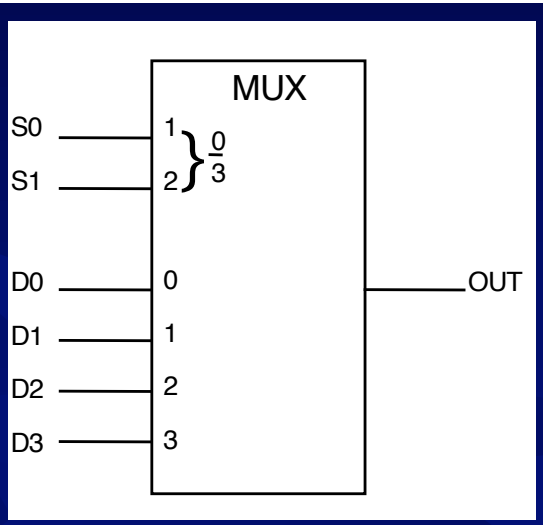
| S1 | S0 | OUT |
|---|---|---|
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |



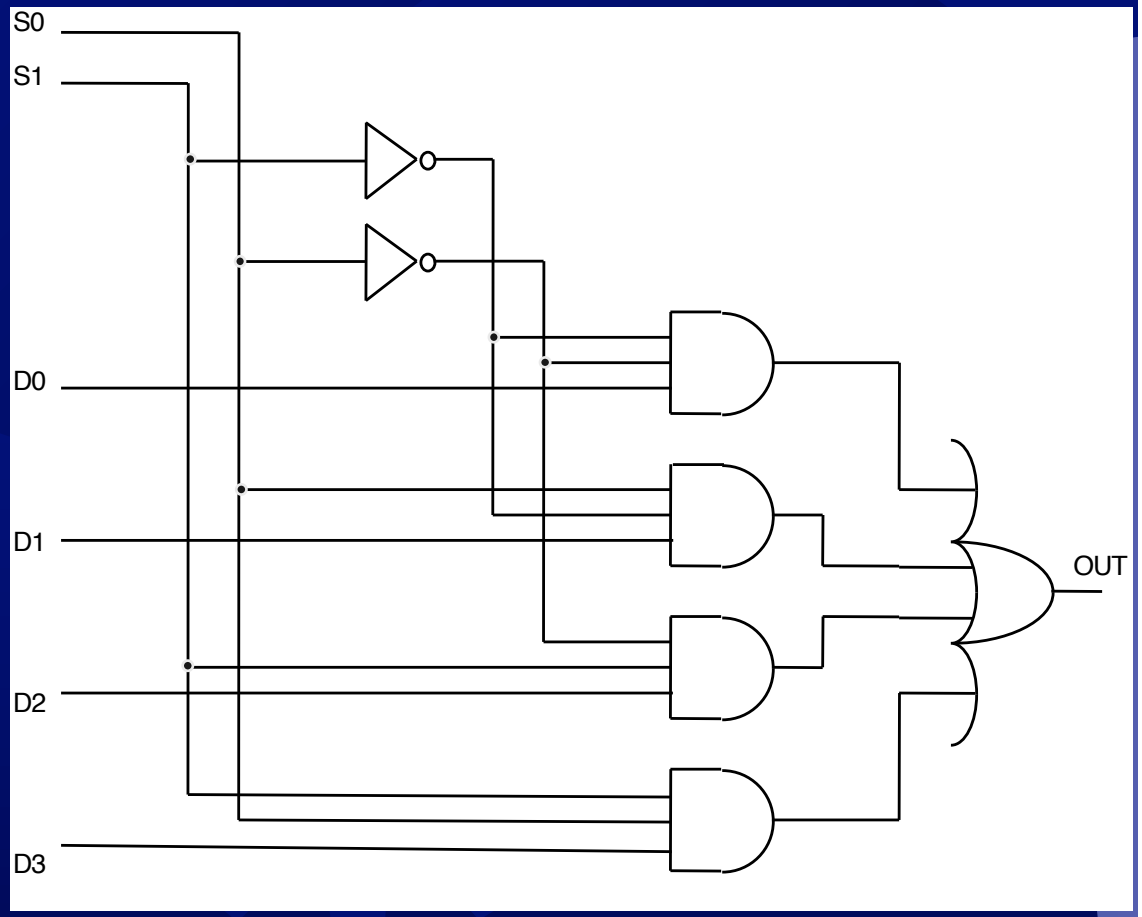
# Multiplexeres (IV)

- Estrutura Interna

| S1 | S0 | OUT |
|---|---|---|
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |

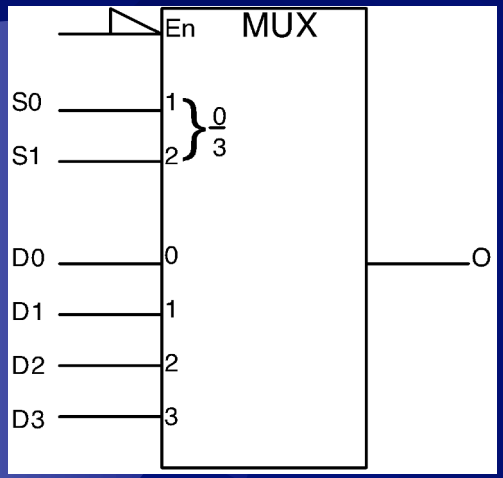


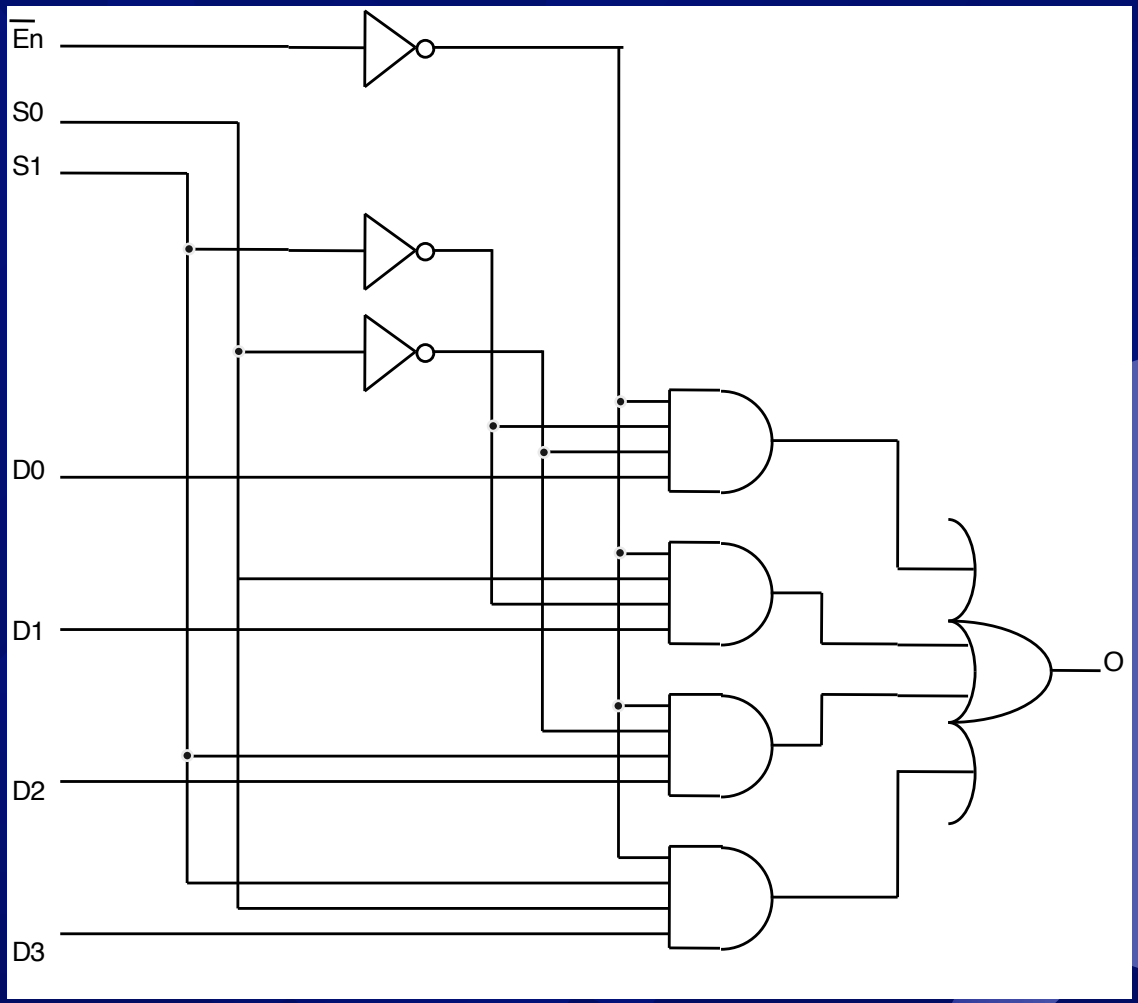

| D3 | D2 | D1 | D0 | S1 | S0 | OUT |
|---|---|---|---|---|---|---|
| X | X | X | 0 | 0 | 0 | 0 |
| X | X | X | 1 | 0 | 0 | 1 |
| X | X | 0 | X | 0 | 1 | 0 |
| X | X | 1 | X | 0 | 1 | 1 |
| X | 0 | X | X | 1 | 0 | 0 |
| X | 1 | X | X | 1 | 0 | 1 |
| 0 | X | X | X | 1 | 1 | 0 |
| 1 | X | X | X | 1 | 1 | 1 |



# Multiplexeres (V)

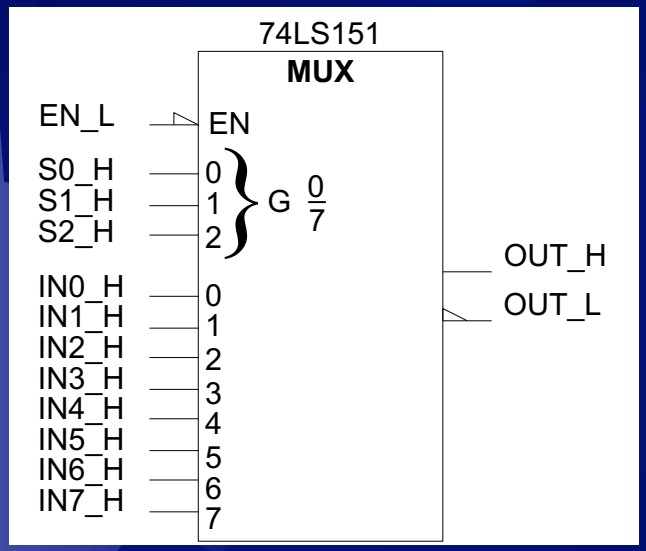
- Estrutura Interna c/ Enable activo a Low



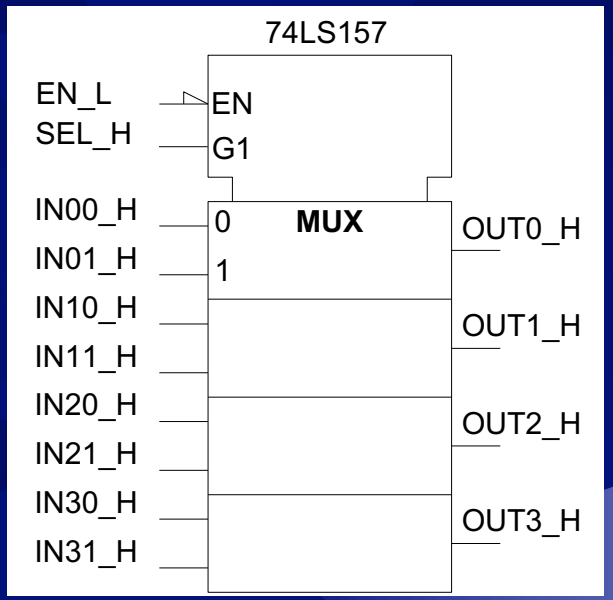



# Multiplexeres (VI)

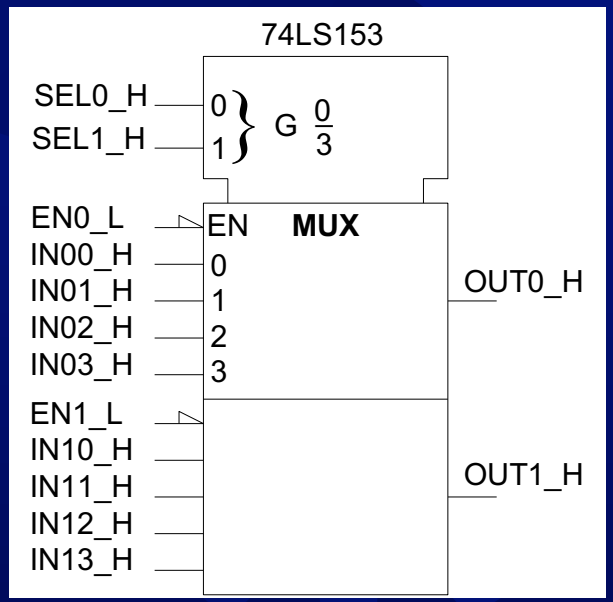
- Simbologia




## QUAD MUX 2:1

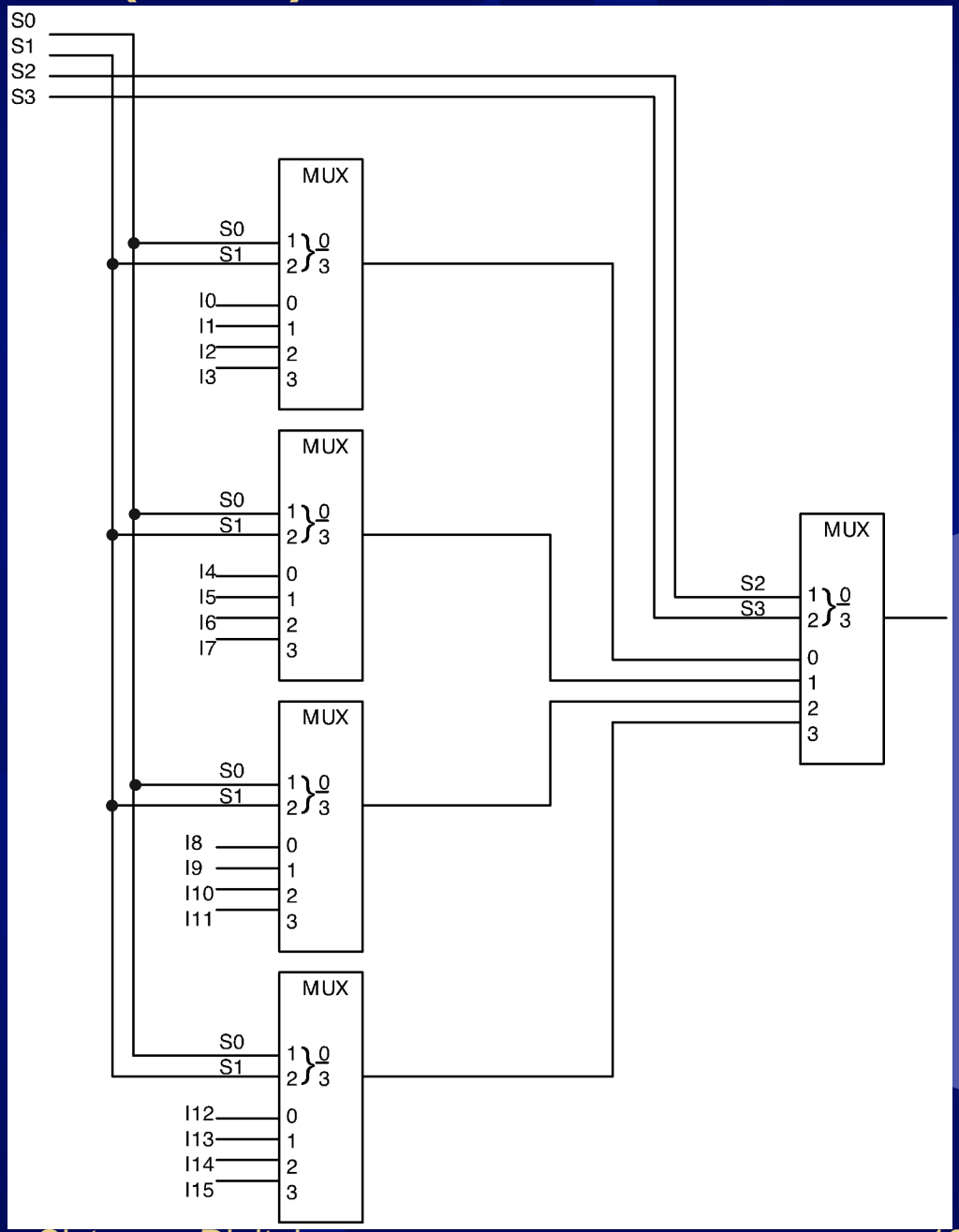



## DUAL MUX 4:1



# Multiplexeres (VII)

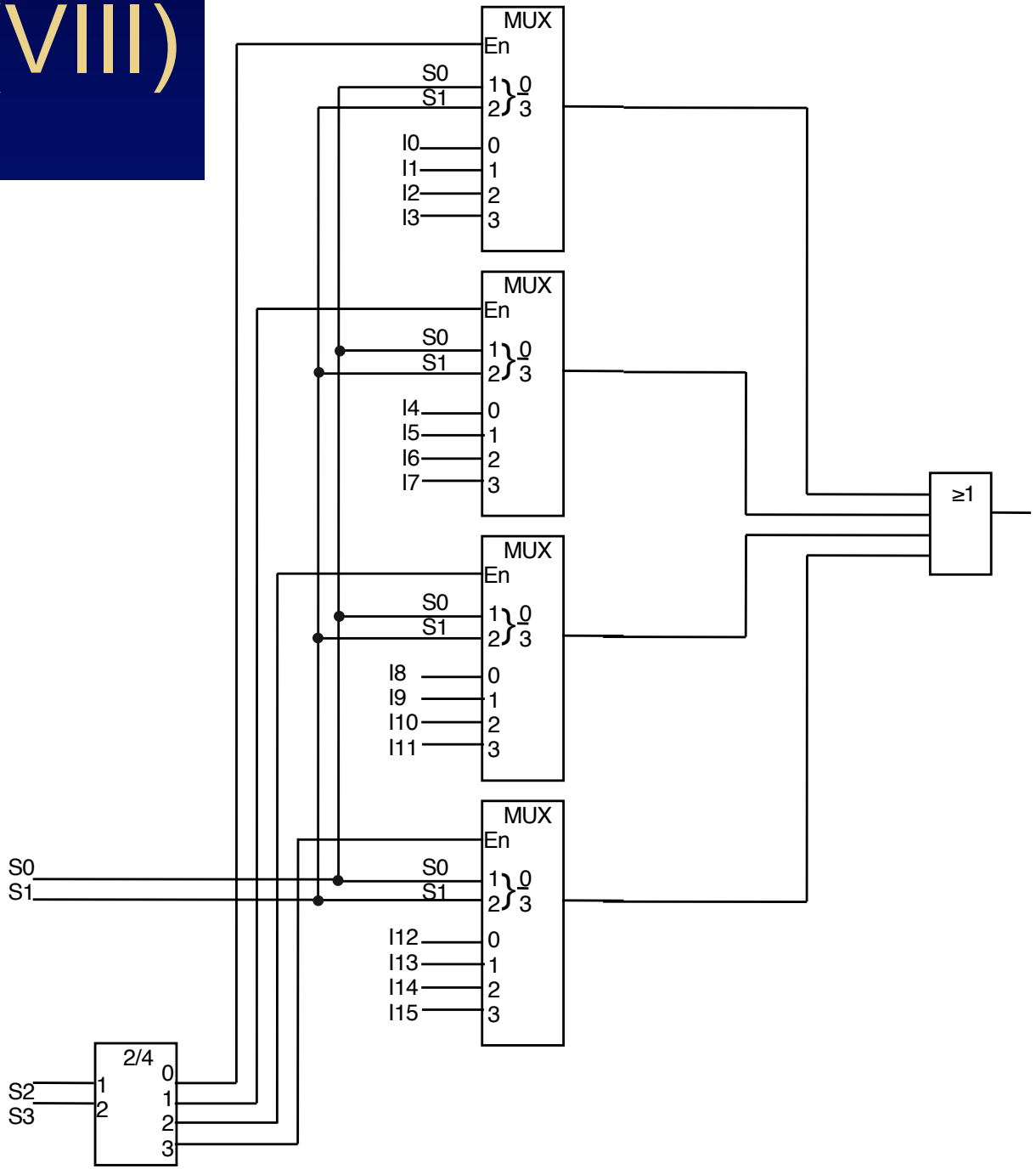
- Expansão de Multiplexers: Obtenção de um Mux 16:1 a partir de 5 Mux 4:1



# Multiplexeres (VIII)

- Expansão de Multiplexers:

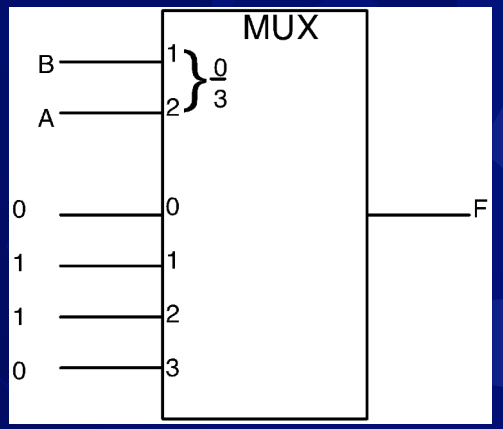
Obtenção de um Mux 16:1 a partir de 4 Mux 4:1 (com Enable) e um descodificador



# Multiplexeres (IX)

- Implementação de funções
- Um mux é construído internamente como uma soma de produtos de todas as variáveis de selecção, ou seja, por uma soma de mintermos, pelo que pode ser facilmente utilizado para implementar funções

| AB | f(A,B) |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 0 |




- Com um multiplexer de *n* variáveis de selecção, pode-se construir qualquer função de *n* variáveis

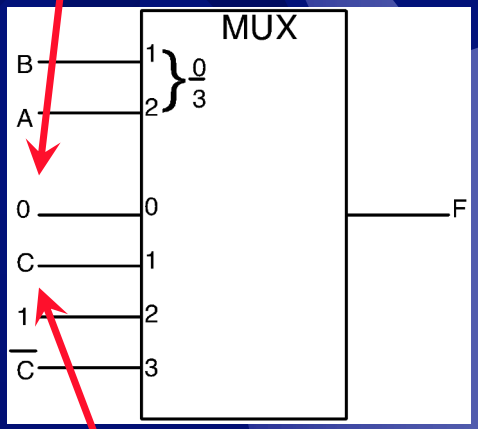
# Multiplexeres (X)

- Implementação de funções (cont.)
- Mas é possível ir mais longe: Com o mesmo MUX e um NOT, pode-se implementar qualquer função de 3 variáveis

| ABC | F(A,B,C) |
|---|---|
| 000 | 0 |
| 001 | 0 |
| 010 | 0 |
| 011 | 1 |
| 100 | 1 |
| 101 | 1 |
| 110 | 1 |
| 111 | 0 |

| ABC | F(A,B,C) | |
|---|---|---|
| 000 | 0 | F = 0 |
| 001 | 0 | |
| 010 | 0 | F = C |
| 011 | 1 | |
| 100 | 1 | F = 1 |
| 101 | 1 | |
| 110 | 1 | F = $\overline{C}$ |
| 111 | 0 | |

Para A= 0 e B=0, a função é sempre 0



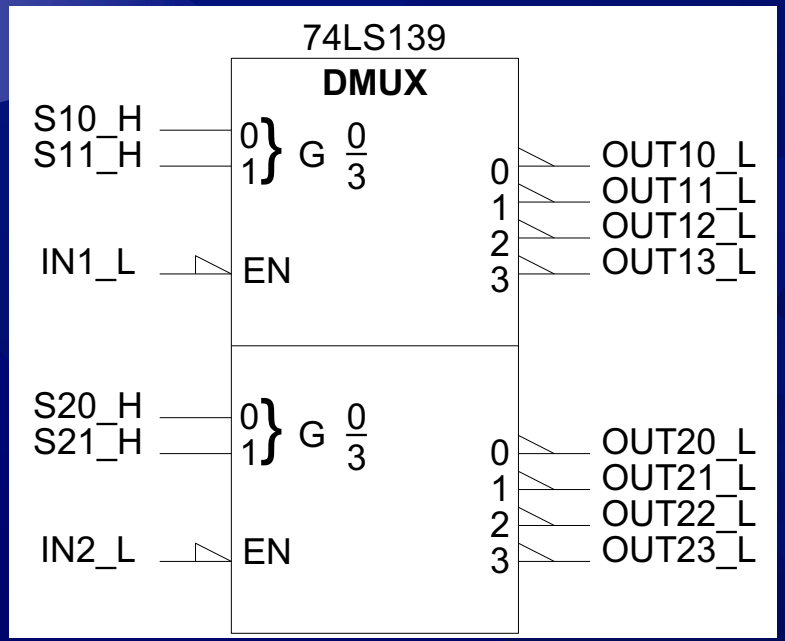

Para A= 0 e B=1, a função toma o valor de C

- Com um multiplexer de *n* variáveis de selecção, e um inversor, pode-se construir qualquer função de *n*+1 variáveis

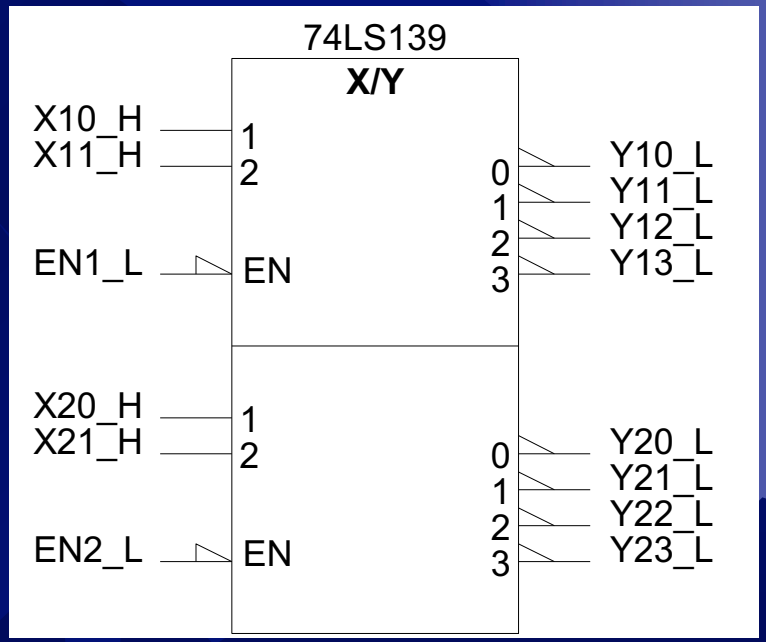
# Demultiplexeres e Descodificadores

- Um Descodificador com Enable é equivalente a um Demultiplexer, sendo as entradas de dados do DEC as entradas de selecção do DMUX, e a entrada de Enable do DEC a entrada de dados do DMUX.
- Nota: a simbologia altera-se de acordo com a funcionalidade do circuito.

**DUAL DMUX 1:4**

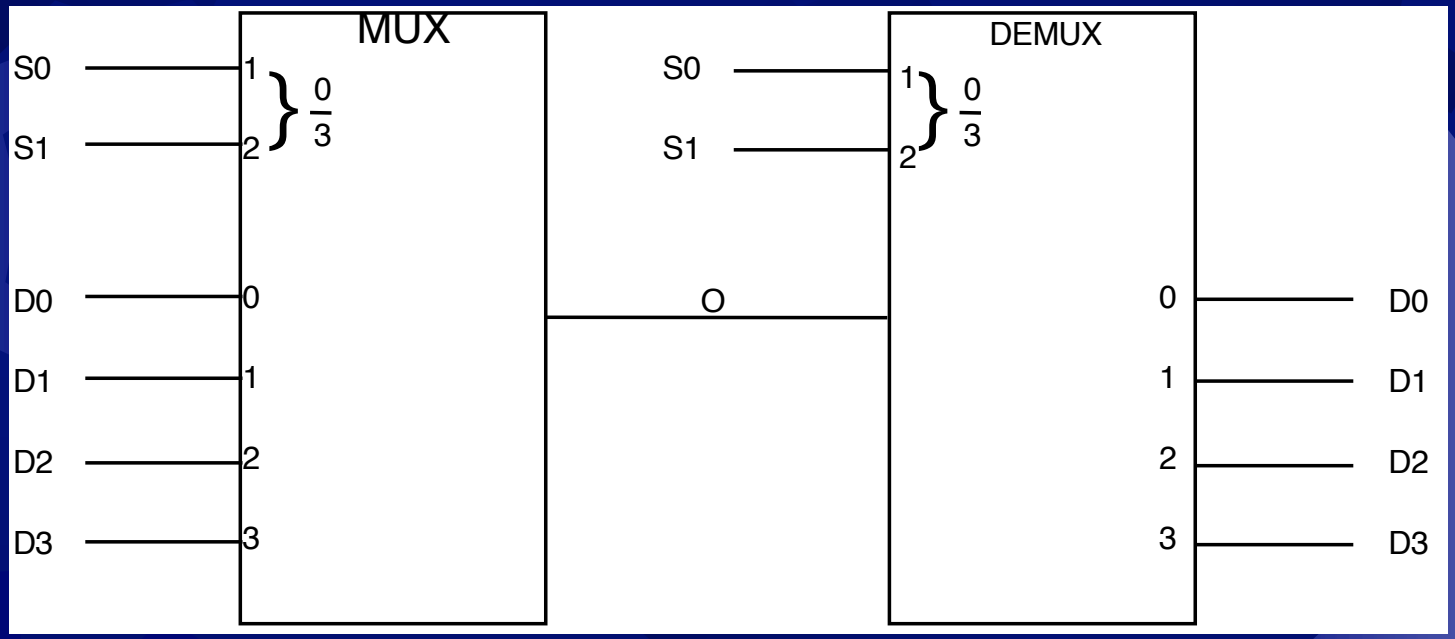



**DUAL DECODER 2:4**



# Multiplexagem e Desmultiplexagem

- Multiplexagem e Desmultiplexagem

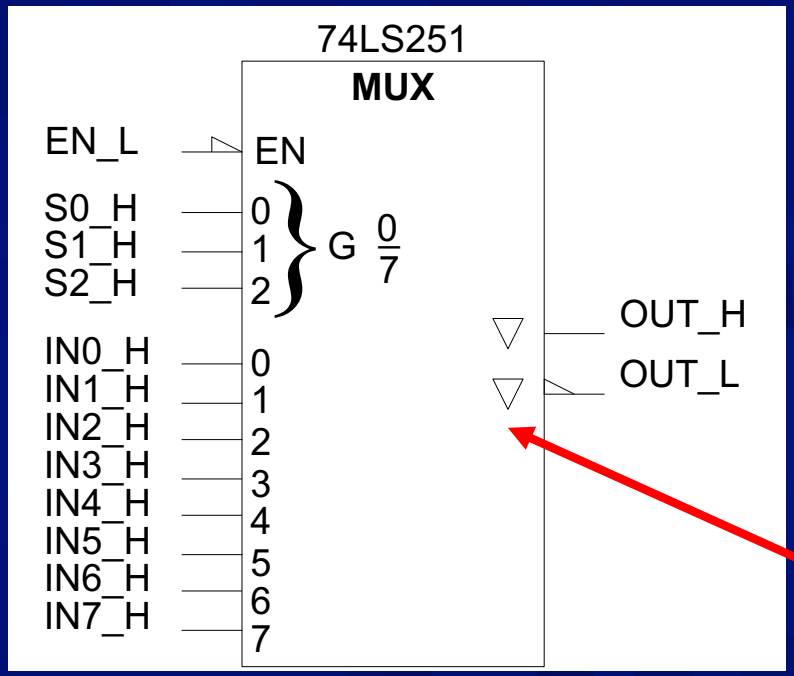


# Saídas Tri-State

- As saídas das portas e circuitos lógicos estudados até agora NÃO podem ser ligadas entre si.
- Mas existem alguns circuitos que possuem características que permitem tal possibilidade:
  - Tecnologia Open Collector (não vai ser estudada)
  - Saídas Tri-State
- Os circuitos com saída Tri-State possuem um enable que quando não está activo coloca as saídas num estado de alta impedância que impede a passagem de corrente.
  - Na prática o circuito comporta-se como se as saídas estivessem fisicamente desligadas dos restantes circuitos.

# Saídas Tri-State (II)

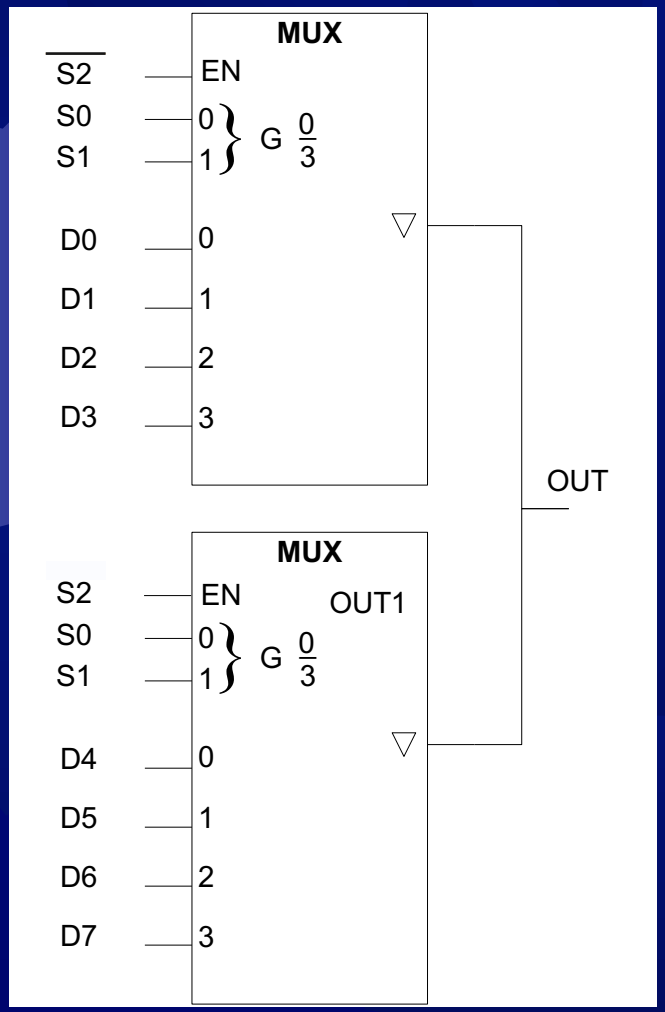
- Exemplo: 74LS251 - Multiplexer 8:1 c/ enable e Tristate

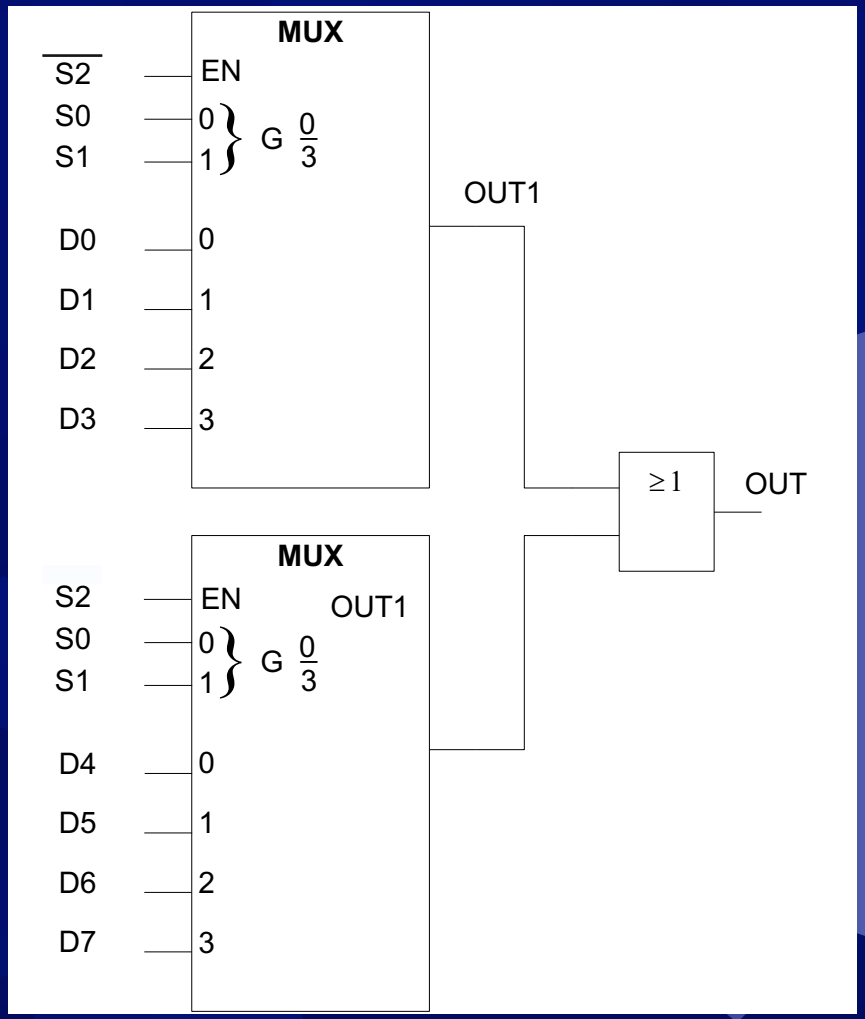



| S2 | S1 | S0 | EN_L | OUT_H |
|---|---|---|---|---|
| - | - | - | 1 | HI-Z |
| 0 | 0 | 0 | 0 | IN0_H |
| 0 | 0 | 1 | 0 | IN1_H |

# Saídas Tri-State (III)

- Exemplo: Implementação de um MUX 8:1 com base em 2 MUX 4:1 com e sem saída Tri-State



# Bibliografia


- Arroz,G., Monteiro,J.C., Oliveira,A., "Arquitectura de Computadores, dos Sistemas Digitais aos Microprocessadores", Capítulo 4, 2ª Edição, 2009
- Mano,M., Kime,C. – "Logic and Computer Design Fundamentals", Prentice Hall, secções 3.4 a 3.6