

# Sistemas Reconfiguráveis

## Eng. de Computação

Profs. Francisco Garcia e Antônio Hamilton Magalhães

Aula 5 – Exemplos de uso do código concorrente:

- Codificadores com prioridade
  - Multiplexadores


# Códigos em VHDL



Na última aula, vimos que existem dois tipos de código em VHDL:

- Código concorrente (paralelo)
  - Uso direto dos operadores
  - Construções com **WHEN**:
    - **WHEN / ELSE**
    - **WITH / SELECT / WHEN**
- Código sequencial
  - Tem de estar dentro de um **PROCESS**

# Construções do código concorrente



Vimos também que, no código concorrente, existem duas construções que usam **WHEN**:

Construção **WHEN** / **ELSE**:

```
signal <= expression1 WHEN condition1 ELSE  
    expression2 WHEN condition2 ELSE  
    .....  
    expressionN;
```

Construção **WITH** / **SELECT** / **WHEN**:

```
WITH identifier SELECT  
    signal <= expression1 WHEN value1,  
    expression2 WHEN value2,  
    .....  
    expressionN WHEN valueN;
```

Vamos continuar a ver exemplos de descrição de circuitos combinacionais de média complexidade usando essas construções

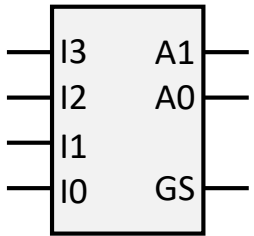
# Codificadores com prioridade (*priority encoder*)

---

- Muitas entradas e poucas saídas
- A saída(N bits) indica qual entrada está ativada
- Pode ter até  $2^N$  entradas
- As entradas podem ser ativas em nível lógico alto ou baixo
- Se mais de uma entrada estiver ativada, a saída será correspondente à entrada com maior prioridade.
- Pode ter uma saída para indicar que existe alguma entrada ativa
- Pode ter uma ou mais entradas de habilitação (*enable*), que pode(m) ser ativa(s) em nível alto ou baixo.
- No caso de usar *enable* , pode ter uma saída para propagação do estado.

# Exemplo enc\_4x2

**Exemplo enc\_4x2:** codificador com quatro entradas ativas em nível alto e saída de código com dois bits  
A[1..0] = saída de código; GS = saída que indica alguma entrada ativa



Entradas				Saídas		
I3	I2	I1	I0	A1	A0	GS
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

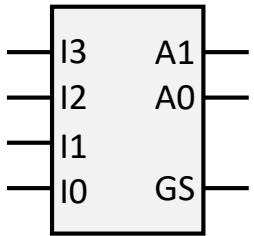
Nenhuma entrada ativa.  
Por isso, GS = '0'

Pelo menos uma  
entrada ativa

A saída A[1..0] indica  
a entrada ativa com  
maior prioridade

# Exemplo enc\_4x2

**Exemplo enc\_4x2:** codificador com quatro entradas ativas em nível alto e saída de código com dois bits  
A[1..0] = saída de código; GS = saída que indica alguma entrada ativa



Entradas				Saídas		
I3	I2	I1	I0	A1	A0	GS
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

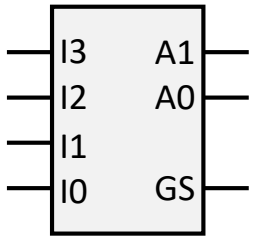
ENTITY enc_4x2 IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        gs: OUT STD_LOGIC;
        a: OUT STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
END ENTITY;

.....
```

Continua na próxima página

# Exemplo enc\_4x2

**Exemplo enc\_4x2:** codificador com quatro entradas ativas em nível alto e saída de código com dois bits  
A[1..0] = saída de código; GS = saída que indica alguma entrada ativa



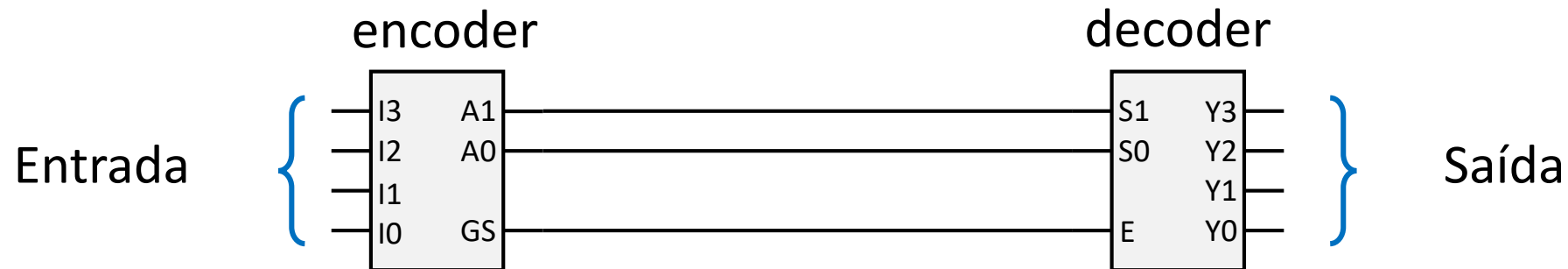
Entradas				Saídas		
I3	I2	I1	I0	A1	A0	GS
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

```
ARCHITECTURE arch OF enc_4x2 IS
BEGIN
    a <= "11" WHEN i(3) = '1' ELSE
        "10" WHEN i(2) = '1' ELSE
        "01" WHEN i(1) = '1' ELSE
        "00";

    gs <= '0' WHEN i = "0000" ELSE -- nenhuma entrada
        '1';
END arch;
```

# Ligação *encoder* / *decoder*

**Pergunta:** A saída desse circuito será sempre igual à entrada?



**Resposta:** Se tiver nenhuma ou apenas uma entrada ativa (alto): sim  
Se tiver mais de uma entrada ativa (alto), não, pois o *decoder* pode ter somente uma saída ativa

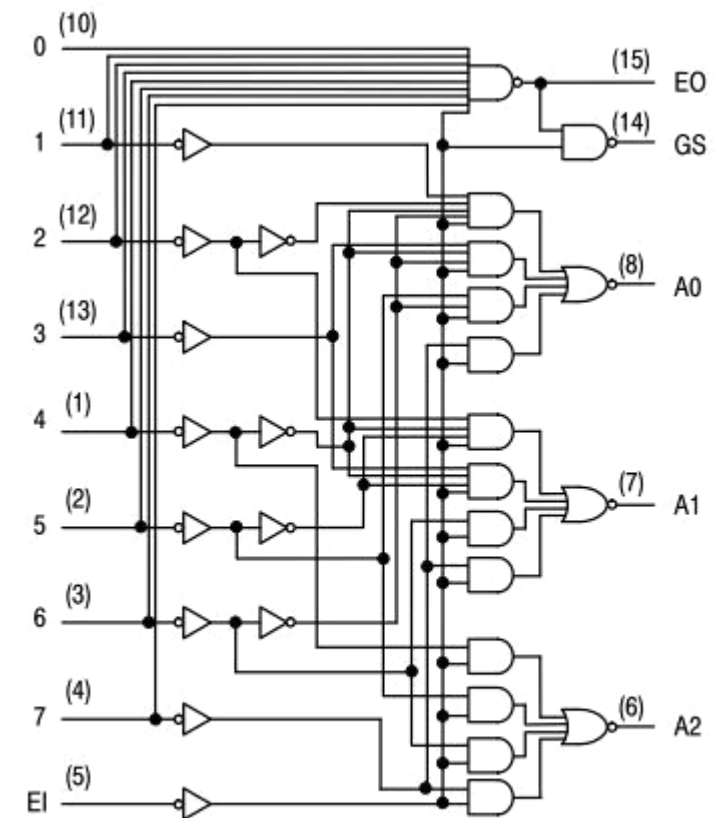


# Exercício 1

Implementar e testar no módulo DE2 um decodificador funcionalmente equivalente ao circuito integrado TTL 74LS148.

SN54/74LS148  
SN54/74LS748  
FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H



SN54/74LS148

Notar que as entradas e saídas são ativas em nível baixo

# Multiplexador

(*multiplex* ou *mux*)

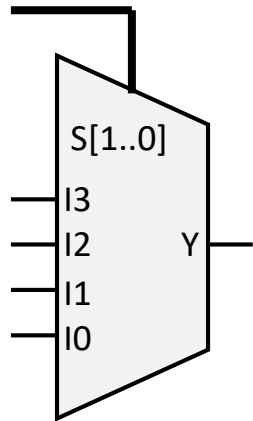
---

- Muitas entradas de dados e uma única saída
- A entrada de seleção (N bits) seleciona qual entrada de dados será enviada à saída
- Pode ter até  $2^N$  entradas de dados
- Pode ter uma ou mais entradas de habilitação (*enable*), que pode(m) ser ativa(s) em nível alto ou baixo.
- Também chamado **seletor de dados**

# Exemplo mux\_4x1

**Exemplo mux\_4x1:** multiplexador com quatro entradas de dados para uma saída

$S[1..0]$  = entrada de seleção (seleciona qual entrada vai para a saída)



Entradas						Saída
S1	S0	I3	I2	I1	I0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

$S[1..0] = "00"$   
 $Y = I0$

$S[1..0] = "01"$   
 $Y = I1$

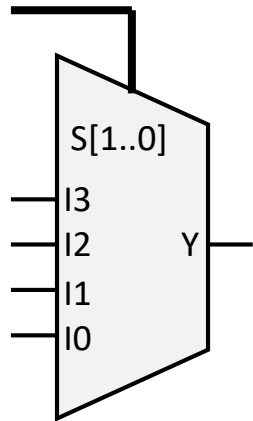
$S[1..0] = "10"$   
 $Y = I2$

$S[1..0] = "11"$   
 $Y = I3$

# Exemplo mux\_4x1

**Exemplo mux\_4x1:** multiplexador com quatro entradas de dados para uma saída

S[1..0] = entrada de seleção (seleciona qual entrada vai para a saída)



Entradas						Saída
S1	S0	I3	I2	I1	I0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux_4x1 IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        s: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        y: OUT STD_LOGIC
    );
END ENTITY;

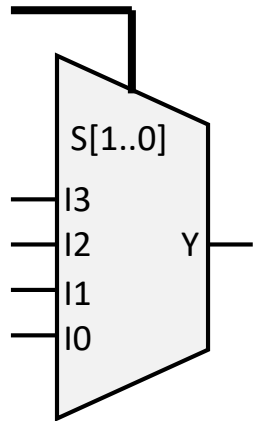
.....
```

Continua na próxima página

# Exemplo mux\_4x1

**Exemplo mux\_4x1:** multiplexador com quatro entradas de dados para uma saída

S[1..0] = entrada de seleção (seleciona qual entrada vai para a saída)



Entradas						Saída
S1	S0	I3	I2	I1	I0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

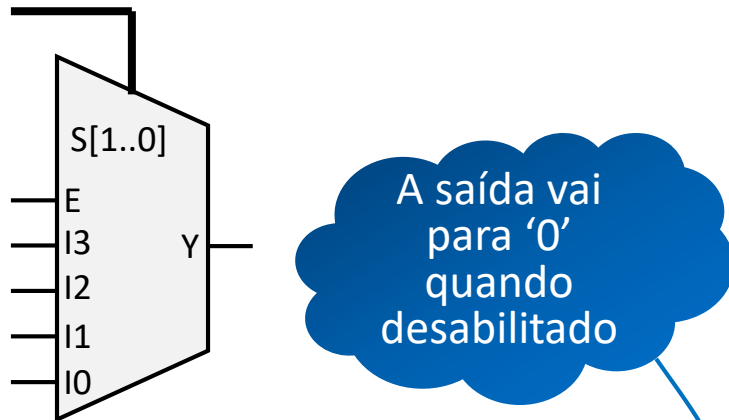
```
ARCHITECTURE arch OF mux_4x1 IS
BEGIN
```

```
    WITH s SELECT
        y <= i(0) WHEN "00",
              i(1) WHEN "01",
              i(2) WHEN "10",
              i(3) WHEN "11";
```

```
END arch;
```

# Exemplo mux\_4x1e

**Exemplo mux\_4x1e:** multiplexador com quatro entradas de dados para uma saída, com *enable* ativo alto



Entradas							Saída
E	S1	S0	I3	I2	I1	I0	Y
0	X	X	X	X	X	X	0
1	0	0	X	X	X	0	0
1	0	0	X	X	X	1	1
1	0	1	X	X	0	X	0
1	0	1	X	X	1	X	1
1	1	0	X	0	X	X	0
1	1	0	X	1	X	X	1
1	1	1	0	X	X	X	0
1	1	1	1	X	X	X	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

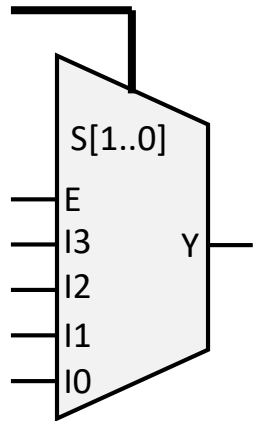
ENTITY mux_4x1e IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        s: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        e: IN STD_LOGIC;
        y: OUT STD_LOGIC
    );
END ENTITY;

.....
```

Continua na próxima página

# Exemplo mux\_4x1e

**Exemplo mux\_4x1e:** multiplexador com quatro entradas de dados para uma saída, com *enable* ativo alto



Entradas							Saída
E	S1	S0	I3	I2	I1	I0	Y
0	X	X	X	X	X	X	0
1	0	0	X	X	X	0	0
1	0	0	X	X	X	1	1
1	0	1	X	X	0	X	0
1	0	1	X	X	1	X	1
1	1	0	X	0	X	X	0
1	1	0	X	1	X	X	1
1	1	1	0	X	X	X	0
1	1	1	1	X	X	X	1

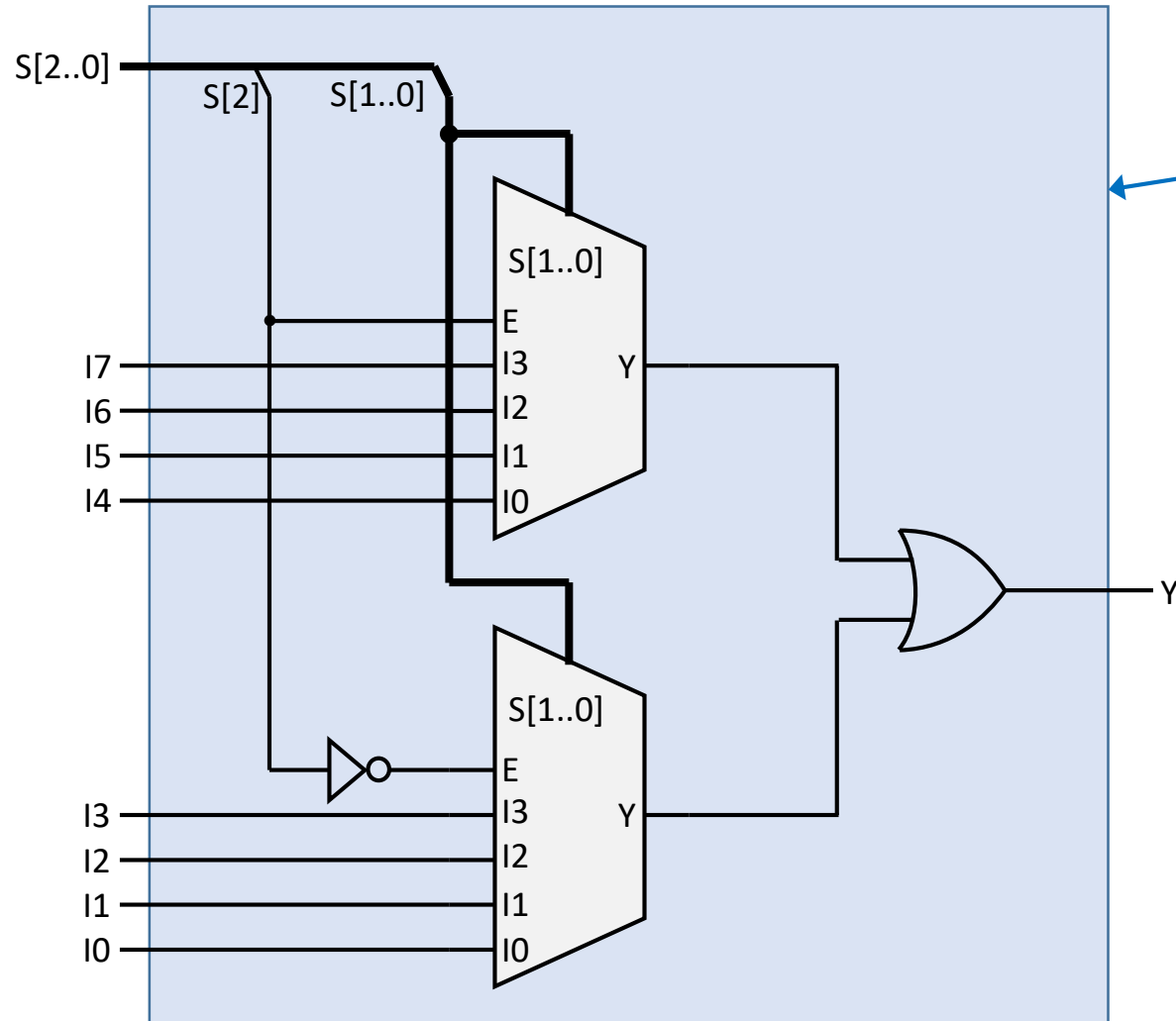
```
ARCHITECTURE arch OF mux_4x1e IS
    SIGNAL aux : STD_LOGIC;
BEGIN
```

```
    WITH s SELECT
        aux <= i(0) WHEN "00",
               i(1) WHEN "01",
               i(2) WHEN "10",
               i(3) WHEN "11";
```

```
    y <= aux WHEN e = '1' ELSE -- when enabled
        '0';                  -- when disabled
```

```
END arch;
```

# Exemplo mux\_4x1e



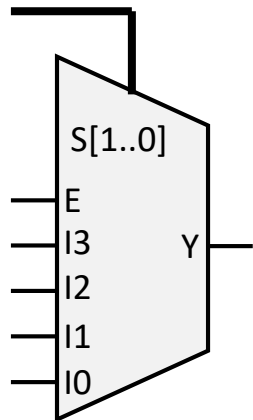
Esse circuito funciona como um multiplexador de 8 entradas para 1 saída

Seleção			Saída
S2	S1	S0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7



# Exemplo mux\_4x1z

**Exemplo mux\_4x1z:** multiplexador com quatro entradas de dados para uma **saída tri-state**, com *enable* ativo alto



A saída vai para alta impedância ('Z') quando desabilitado

Entradas							Saída
E	S1	S0	I3	I2	I1	I0	Y
0	X	X	X	X	X	X	Z
1	0	0	X	X	X	0	0
1	0	0	X	X	X	1	1
1	0	1	X	X	0	X	0
1	0	1	X	X	1	X	1
1	1	0	X	0	X	X	0
1	1	0	X	1	X	X	1
1	1	1	0	X	X	X	0
1	1	1	1	X	X	X	1

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

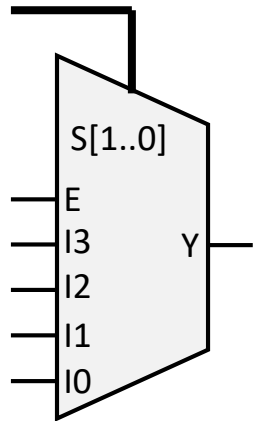
ENTITY mux_4x1z IS
    PORT (
        i: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        s: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        e: IN STD_LOGIC;
        y: OUT STD_LOGIC
    );
END ENTITY;

.....
```

Continua na próxima página

# Exemplo mux\_4x1z

Exemplo mux\_4x1z: multiplexador com quatro entradas de dados para uma **saída tri-state**, com *enable* ativo alto



Entradas							Saída
E	S1	S0	I3	I2	I1	I0	Y
0	X	X	X	X	X	X	Z
1	0	0	X	X	X	0	0
1	0	0	X	X	X	1	1
1	0	1	X	X	0	X	0
1	0	1	X	X	1	X	1
1	1	0	X	0	X	X	0
1	1	0	X	1	X	X	1
1	1	1	0	X	X	X	0
1	1	1	1	X	X	X	1

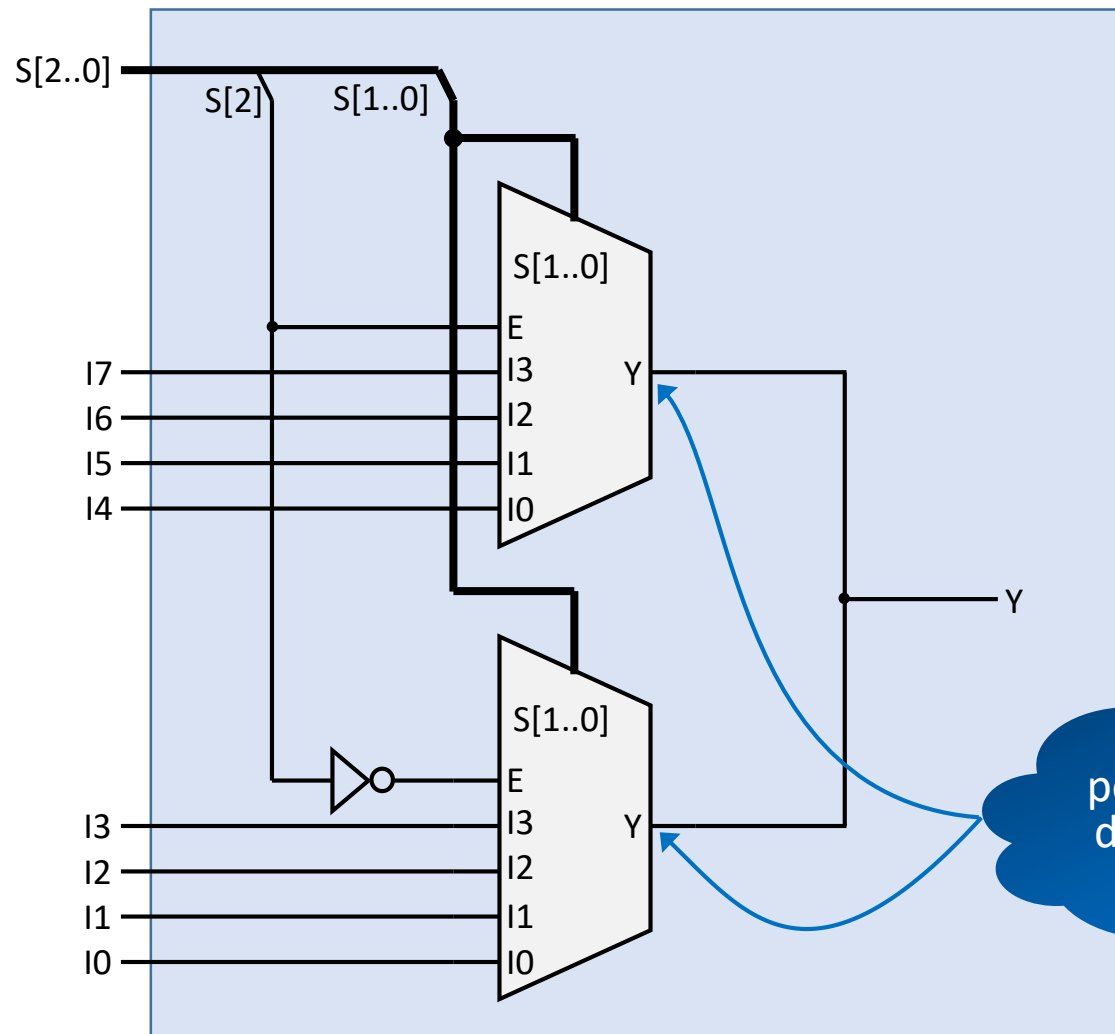
```
ARCHITECTURE arch OF mux_4x1z IS
    SIGNAL aux : STD_LOGIC;
BEGIN
```

```
    WITH s SELECT
        aux <= i(0) WHEN "00",
               i(1) WHEN "01",
               i(2) WHEN "10",
               i(3) WHEN "11";
```

```
    y <= aux WHEN e = '1' ELSE -- when enabled
        'Z';                  -- when disabled
```

```
END arch;
```

# Exemplo mux\_4x1z



Esse circuito funciona como um multiplexador de 8 entradas para 1 saída

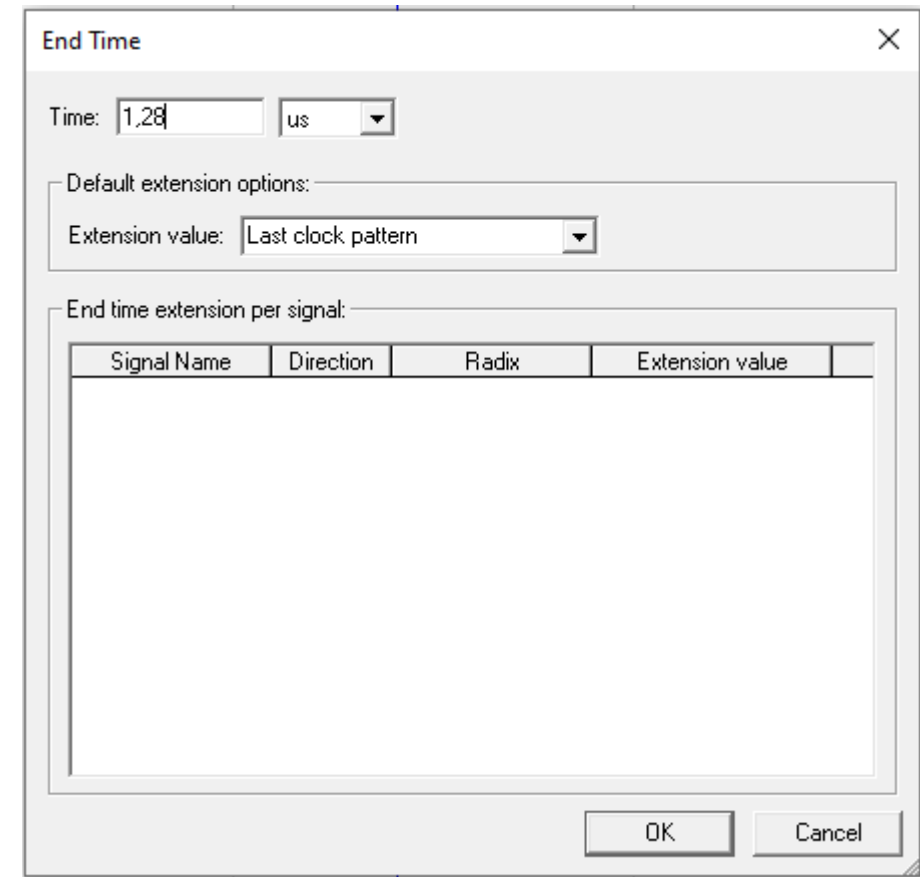
Saídas *tri-state* podem ser ligadas diretamente uma com a outra

Seleção			Saída
S2	S1	S0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

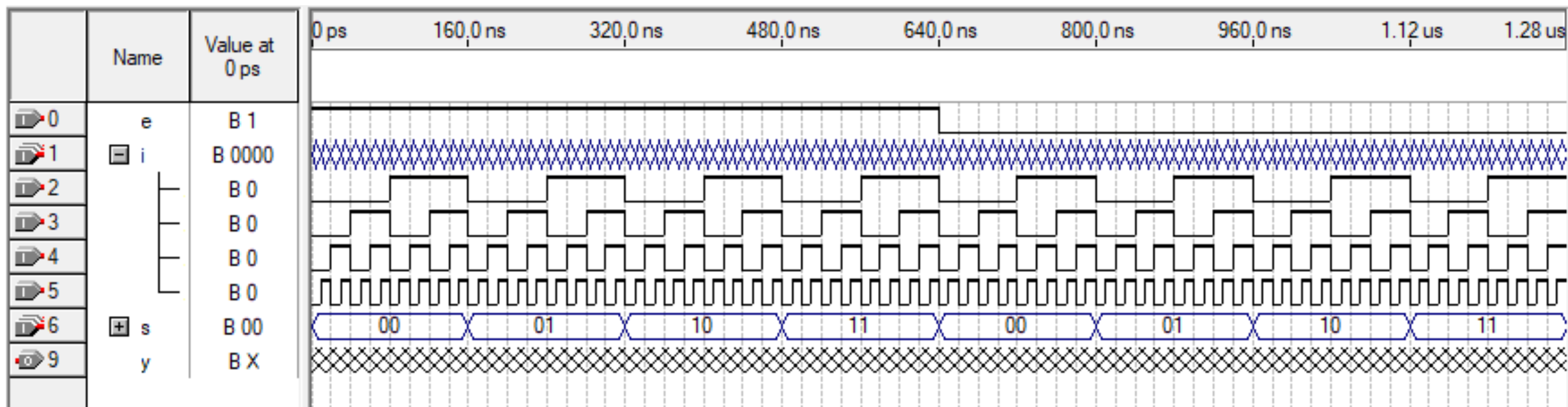
# Exercício 2

Testar, através de simulação funcional, o exemplo **mux\_4x1z**



- Após criar o arquivo para simulação, no menu “**Edit**”, na opção “**End time...**”, ajustar o parâmetro “**Time**” para o valor **1.28us**;
- inserir todos os nós para a simulação;
- de 0 até 640 ns, fazer **e** = ‘1’ (habilitado) e, após esse tempo, **e** = ‘0’;
- programar a entrada i com uma contagem, contando a cada 10ns;
- programar a entrada s com uma contagem, contando a cada 160ns.



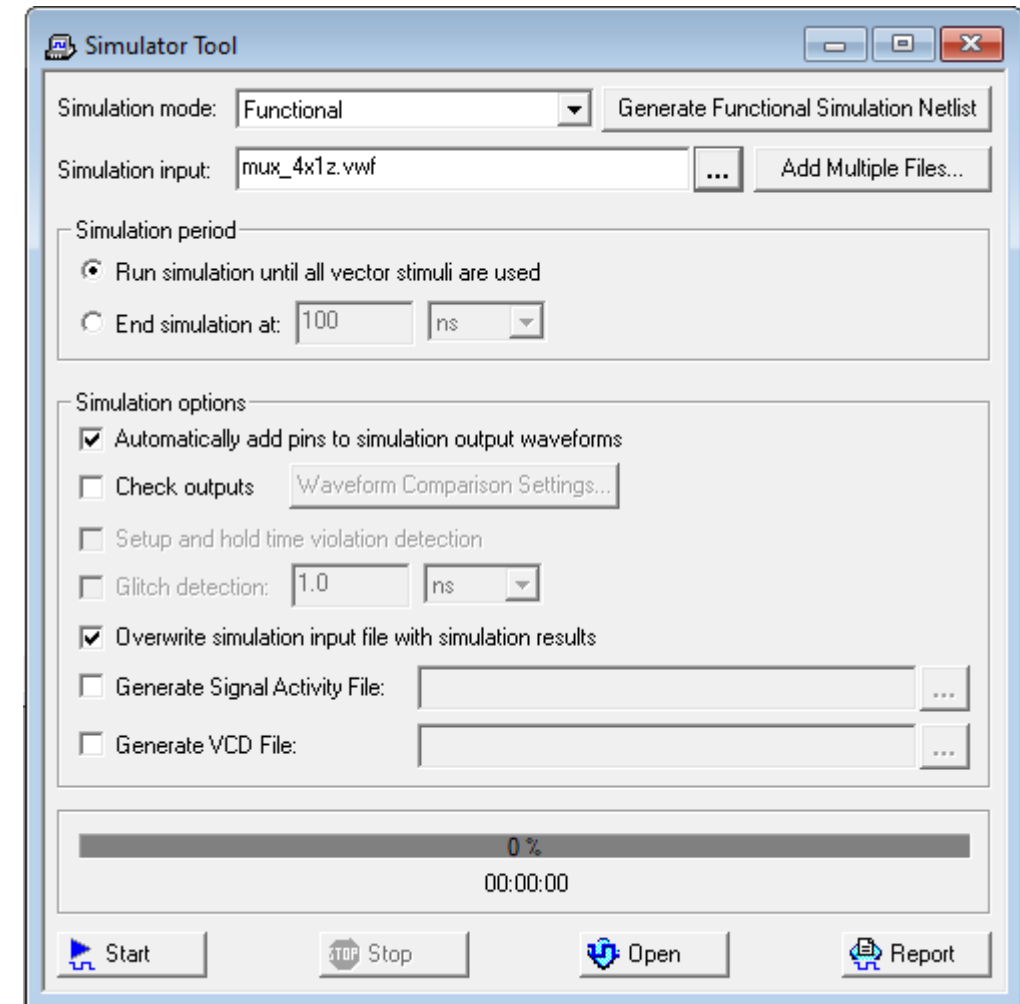
- Salvar o arquivo.



# Exercício 2

- No menu “Processing”, escolher a opção “Simulator Tool”;
- na janela aberta, no campo “Simulation mode”, escolher a opção “Functional”
- clicar no botão “Generate Functional Simulation Netlist”;
- ativar a opção “Overwrite simulation input file with simulation results”;
- clicar no botão  Start para iniciar a simulação;
- se o arquivo de simulação (.vwf) não estiver aberto, clicar no botão  Open

O resultado obtivo foi o esperado?



# Exercício 3

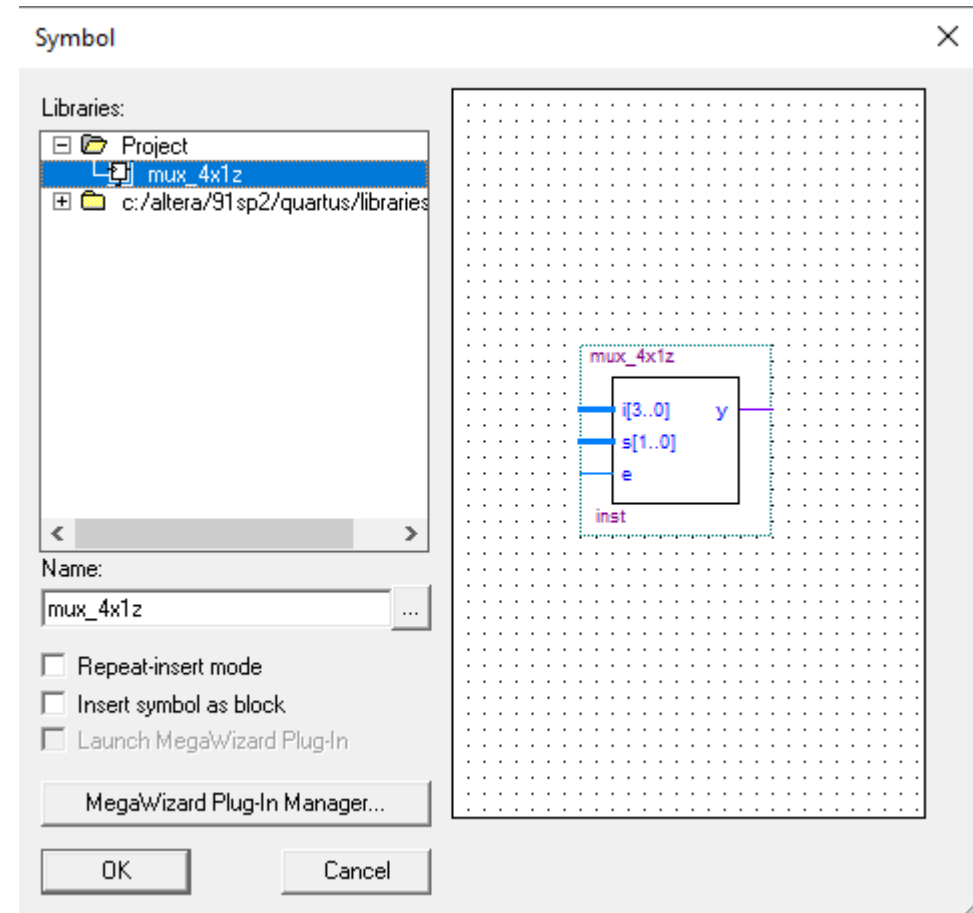


Implementar e testar, no módulo DE2, um multiplexador de 8 entradas para uma saída usando o mux\_4x1z criado e testado anteriormente.

- Abrir o arquivo .vhd do projeto **mux\_4x1z**
- No menu “File”, escolher a opção “Create/Update” e, na janela aberta, selecionar “Create Symbol File for Current File”

# Exercício 3

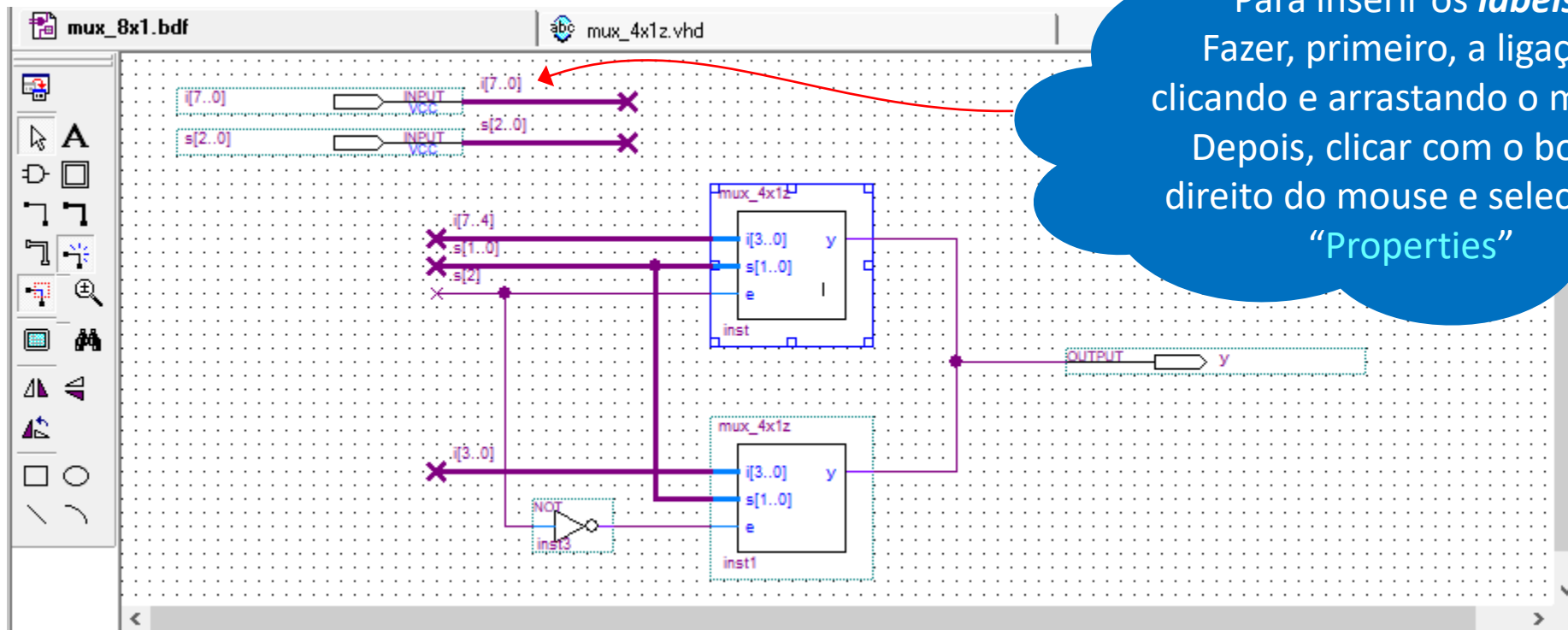
- Criar um arquivo novo, do tipo “**Block Diagram/Schematic File**”
- Com o apontador do *mouse* em qualquer lugar vazio dentro da área de trabalho, fazer um duplo clique (botão esquerdo do *mouse*) para selecionar o símbolo do bloco **mux\_4x1z** criando anteriormente;
- Inserir esse símbolo duas vezes no diagrama.





# Exercício 3

- Inserir os outros símbolos e fazer as ligações conforme mostrado abaixo;
- salvar o arquivo.



Para inserir os **labels**:  
Fazer, primeiro, a ligação,  
clikando e arrastando o mouse.  
Depois, clicar com o botão  
direito do mouse e selecionar  
“Properties”

# Exercício 3

- No menu “**Project**” selecionar a opção “**Set as Top-Level Entity**”, ou usar o atalho “**Cntrl+Shift+J**”.
- Fazer o processo de **análise e síntese** e verificar as mensagens de erros e avisos
- Vincular as entradas a chaves e a saída a um LED do módulo, conforme tabela de pinos na Aula3
- Compilar e programar o modulo DE2 com o projeto criado;
- Testar e verificar se o funcionamento corresponde ao esperado.

Não esquecer de ajustar a opção “**As input tri-stated**” para os “**Unused Pins**”

Para análise e síntese:



ou “**Ctrl+K**”

Para compilar:



ou “**Ctrl+L**”

Para programar:



# Multiplexador

*(multiplex ou mux)*

---

## Observação:

Seria perfeitamente possível descrever um multiplexador com oito entradas e uma saída, da mesma maneira que foi feito no exemplo mux\_4x1, utilizando código concorrente de VHDL.

O uso de dois multiplexadores mux\_4x1z para construir um multiplexador 8x1 foi feito como exemplo de um **projeto hierárquico**, para ilustrar como instanciar blocos previamente construídos em um projeto de maior complexidade.

Nesse exemplo, o topo da hierarquia foi feito através de diagrama esquemático. Entretanto, poderia ser feito usando a linguagem VHDL.



*Fim*

*Até a próxima aula!*