



**Universidade Federal de Santa Catarina**  
**Centro Tecnológico**  
Departamento de Informática e Estatística  
Curso de Graduação em Ciências da Computação



# **Sistemas Digitais**

**INE 5406**

## **Aula 3-P**

**Descrição de somadores em VHDL, síntese com o Quartus II e simulação com o ModelSim. Uso dos tipos `std_logic` e `std_logic_vector`. Comandos de atribuição em VHDL: atribuição simples, com sinal selecionado e com sinal condicional.**

**Prof. José Luís Güntzel**  
**[guntzel@inf.ufsc.br](mailto:guntzel@inf.ufsc.br)**

**[www.inf.ufsc.br/~guntzel/ine5406/ine5406.html](http://www.inf.ufsc.br/~guntzel/ine5406/ine5406.html)**

# Introdução à Linguagem VHDL

---

## ▶ Entidade e Arquitetura

**Uma descrição VHDL é dividida em duas partes fundamentais:**

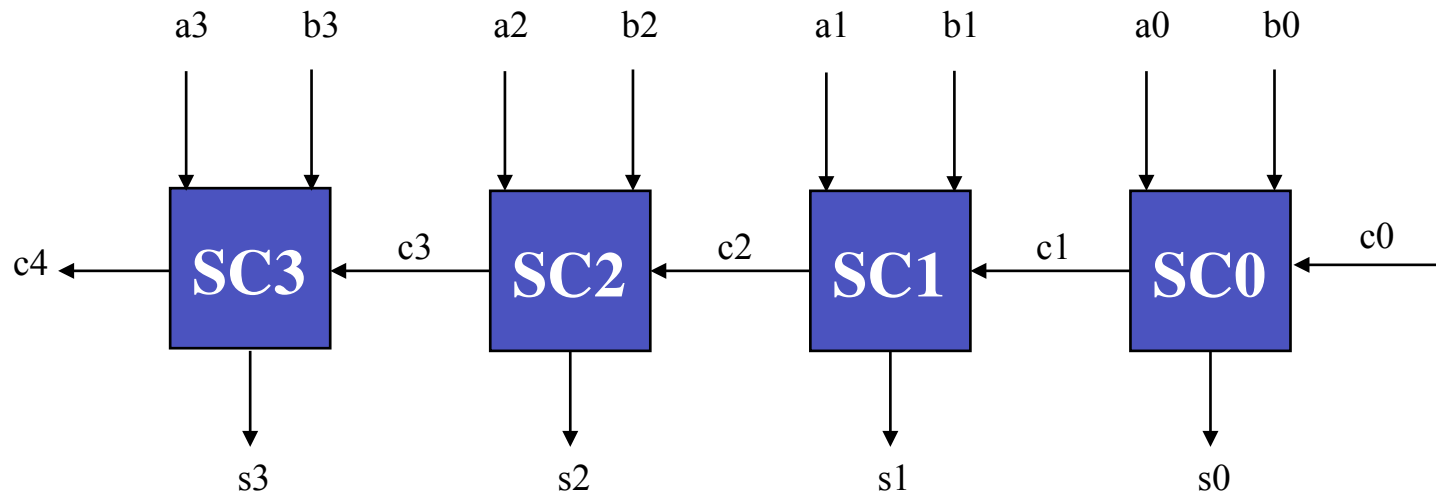
- 1) Entidade (Entity) – Descreve a interface do sistema digital descrito com o mundo externo. Apresenta a definição dos pinos de entrada e saída.**
- 2) Arquitetura (Architecture) – Descreve o comportamento ou a estrutura do sistema digital. Define como a função do sistema é realizada.**

# Introdução à Linguagem VHDL

## ► Descrição de um Operador Aritmético

Somador de 4 bits:

- Com hierarquia
- Usando 4 somadores completos de 1 bit

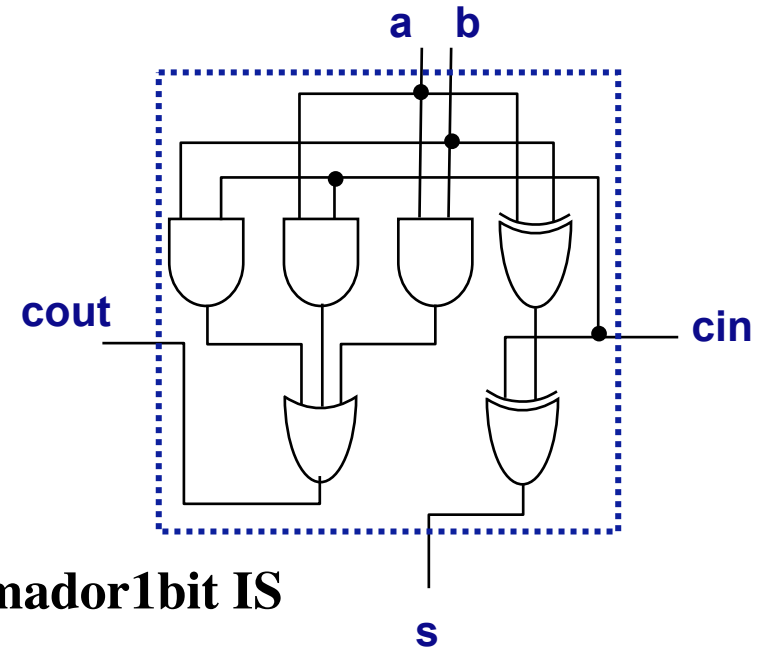


# Introdução à Linguagem VHDL

## ► Somador Completo (*Full Adder*)

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY somador1bit IS  
    PORT (cin, a, b : IN STD_LOGIC;  
          s, cout : OUT STD_LOGIC);  
END somador1bit ;
```

```
ARCHITECTURE comportamento OF somador1bit IS  
BEGIN  
    s <= a XOR b XOR cin;  
    cout <= (a AND b) OR (a AND cin) OR (b AND cin);  
END comportamento;
```



# Introdução à Linguagem VHDL

## ▶ Somador Paralelo de 4 Bits (com Hierarquia)

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

```
ENTITY somador4bits IS
```

```
    PORT (cin, a3, a2, a1, a0, b3, b2, b1, b0 : IN STD_LOGIC;  
          s3, s2, s1, s0, cout : OUT STD_LOGIC);
```

```
END somador4bits;
```

```
ARCHITECTURE estrutura OF somador4bits IS
```

```
    SIGNAL c1, c2, c3: STD_LOGIC;
```

```
    COMPONENT somador1bit
```

```
        PORT (cin, a, b : IN STD_LOGIC;  
              s, cout : OUT STD_LOGIC);
```

```
END COMPONENT;
```

```
BEGIN
```

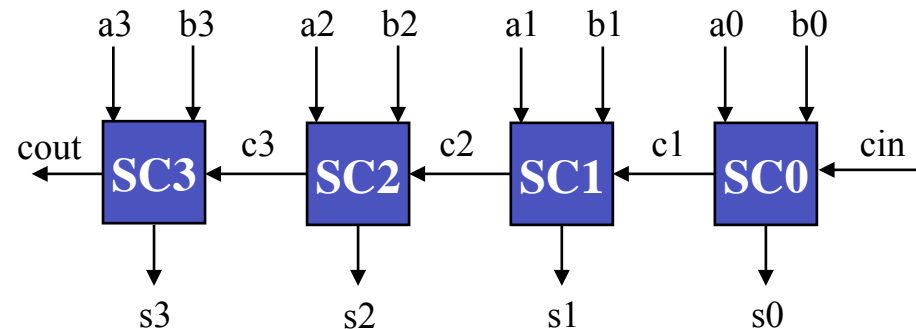
```
    SC0: somador1bit PORT MAP (cin, a0, b0, s0, c1);
```

```
    SC1: somador1bit PORT MAP (c1, a1, b1, s1, c2);
```

```
    SC2: somador1bit PORT MAP (c2, a2, b2, s2, c3);
```

```
    SC3: somador1bit PORT MAP (c3, a3, b3, s3, cout);
```

```
END estrutura;
```



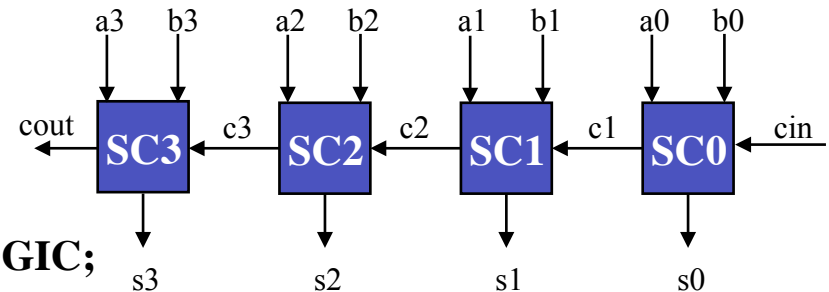
# Introdução à Linguagem VHDL

## ► Somador Paralelo de 4 Bits (s/ Hierarquia, v.1)

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

```
ENTITY somador4bitsv1 IS  
    PORT (cin, a3, a2, a1, a0, b3, b2, b1, b0 : IN STD_LOGIC;  
          s3, s2, s1, s0, cout : OUT STD_LOGIC);  
END somador4bitsv1;
```

```
ARCHITECTURE comportamento OF somador4bitsv1 IS  
    SIGNAL c1, c2, c3: STD_LOGIC;  
BEGIN  
    s0 <= a0 XOR b0 XOR cin;  
    c1 <= (a0 AND b0) OR (a0 AND cin) OR (b0 AND cin);  
    s1 <= a1 XOR b1 XOR c1;  
    c2 <= (a1 AND b1) OR (a1 AND c1) OR (b1 AND c1);  
    s2 <= a2 XOR b2 XOR c2;  
    c3 <= (a2 AND b2) OR (a2 AND c2) OR (b2 AND c2);  
    s3 <= a3 XOR b3 XOR c3;  
    cout <= (a3 AND b3) OR (a3 AND c3) OR (b3 AND c3);  
END comportamento;
```



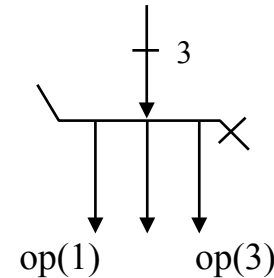
# Introdução à Linguagem VHDL

## ► Representação de Números em VHDL

### Exemplos de declarações de sinais com vários bits

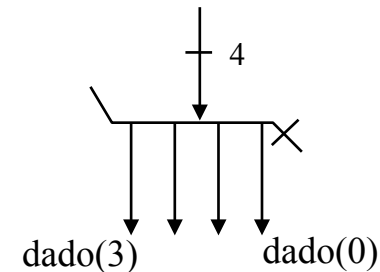
#### Exemplo 1:

**SIGNAL op: STD\_LOGIC\_VECTOR ( 1 TO 3);**



#### Exemplo 2:

**SIGNAL dado: STD\_LOGIC\_VECTOR ( 3 DOWNTO 0);**



# Introdução à Linguagem VHDL

---

## ► Representação de Números em VHDL

Se o sinal é declarado como `STD_LOGIC_VECTOR` pode-se atribuir-lhe uma string binária de várias maneiras...

```
dado<= "1101";
```

**OU**

```
dado(3) <= '1'; dado(2) <= '1'; dado(1) <= '0'; dado(0) <= '1';
```

**OU**

```
dado(3) <= '1'; dado(2 DOWNTO 0) <= "101";
```



# Introdução à Linguagem VHDL

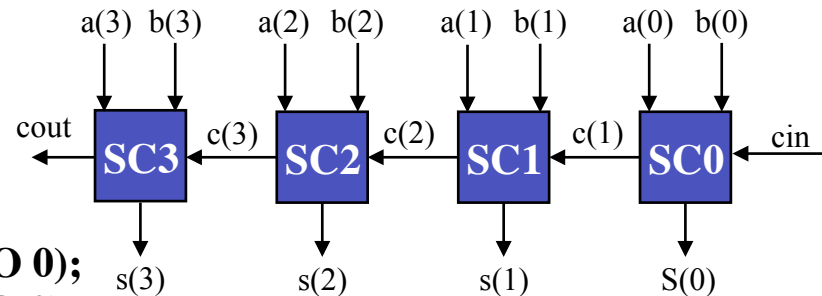
## ▶ Somador Paralelo de 4 Bits (com Hierarquia)

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

```
ENTITY somador4bits IS  
  PORT (cin : IN STD_LOGIC;  
        a, b : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
        s : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);  
        cout : OUT STD_LOGIC);  
END somador4bits;
```

```
ARCHITECTURE comportamento OF somador4bits IS  
  SIGNAL c : STD_LOGIC_VECTOR (3 DOWNTO 1);  
  COMPONENT somador1bit  
    PORT (cin, a, b : IN STD_LOGIC;  
          s, cout : OUT STD_LOGIC);  
  END COMPONENT;
```

```
BEGIN  
  SC0: somador1bit PORT MAP (cin, a(0), b(0), s(0), c(1));  
  SC1: somador1bit PORT MAP (c(1), a(1), b(1), s(1), c(2));  
  SC2: somador1bit PORT MAP (c(2), a(2), b(2), s(2), c(3));  
  SC3: somador1bit PORT MAP (c(3), a(3), b(3), s(3), cout);  
END comportamento;
```



# Introdução à Linguagem VHDL

## ▶ Somador Paralelo de 4 Bits (s/ Hierarquia, v.1)

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

```
ENTITY somador4bits IS
```

```
  PORT (cin : IN STD_LOGIC;  
        a, b : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
        s : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);  
        cout : OUT STD_LOGIC);
```

```
END somador4bits;
```

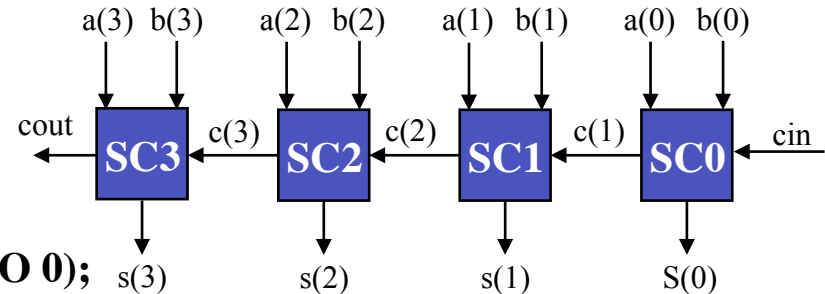
```
ARCHITECTURE comportamento OF somador4bits IS
```

```
  SIGNAL c : STD_LOGIC_VECTOR (3 DOWNTO 1);
```

```
BEGIN
```

```
  s(0) <= a(0) XOR b(0) XOR cin;  
  c(1) <= (a(0) AND b(0)) OR (a(0) AND cin) OR (b(0) AND cin);  
  s(1) <= a(1) XOR b(1) XOR c(1);  
  c(2) <= (a(1) AND b(1)) OR (a(1) AND c(1)) OR (b(1) AND c(1));  
  s(2) <= a(2) XOR b(2) XOR c(2);  
  c(3) <= (a(2) AND b(2)) OR (a(2) AND c(2)) OR (b(2) AND c(2));  
  s(3) <= a(3) XOR b(3) XOR c(3);  
  cout <= (a(3) AND b(3)) OR (a(3) AND c(3)) OR (b(3) AND c(3));
```

```
END comportamento;
```



# Introdução à Linguagem VHDL

---

## ▶ Atribuição de Operações Aritméticas

**Se:**

```
SIGNAL A, B, S : STD_LOGIC_VECTOR ( 3 DOWNTO 0);
```

**Então:**

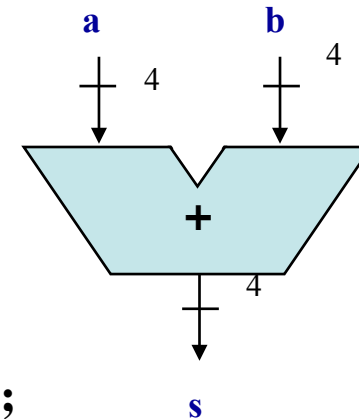
```
S <= A + B;
```

representa um somador de 4 bits...

# Introdução à Linguagem VHDL

## ▶ Somador Paralelo de 4 Bits (s/ Hierarquia, v.2)

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
USE ieee.std_logic_signed.all;  
  
ENTITY somador4bitsv2 IS  
PORT ( a, b : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
      s : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));  
END somador4bitsv2;  
ARCHITECTURE comportamento OF somador4bitsv2 IS  
  
BEGIN  
    s <= a + b;  
END comportamento;
```

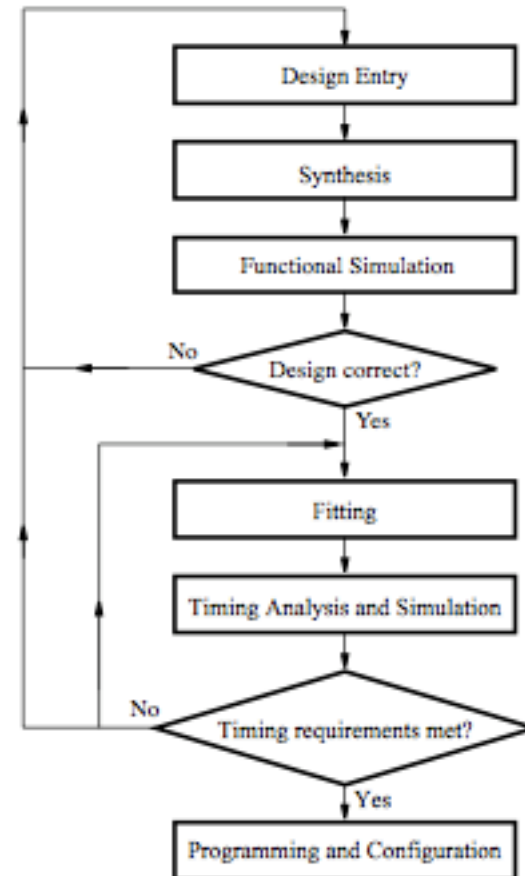


**A maior parte dos dispositivos FPGAs possuem estruturas otimizadas que facilitam a implementação da adição. (Implica em uso de *macro-function*)**

# Projeto de Sistemas Digitais com Ferramentas EDA

## ► Fluxo de Projeto para FPGAs

Documentação Oficial do Quartus II  
Disponível em  
<http://www.altera.com/literature/lit-qts.jsp>



# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Passos do projeto “somador4bits”

### Organizando o Ambiente de Trabalho no Computador

1. Na pasta “Meus Documentos”, criar uma pasta com nome “**somador4bits**”.
2. Acessar o sítio “[www.inf.ufsc.br/~guntzel/ine5406/aula3P](http://www.inf.ufsc.br/~guntzel/ine5406/aula3P)” e baixar os arquivos ali disponíveis para a pasta recém-criada. Os arquivos são:
  - > somador1bit.vhd
  - > somador4bits.vhd

**Obs: jamais crie seus projetos na mesma pasta onde o Quartus II ou o ModelSim estão instalados!**

# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Passos do projeto “somador4bits” Invocando o Quartus II e Criando um Projeto

3. Invocar o Quartus II (a partir do ícone na área de trabalho, ou a partir do “Iniciar->Programas” do windows, sub-menu “Altera”).
4. Na opção “New” (canto superior da janela), selecionar “New Project Wizard”.
5. Clicar em “Next”.
6. Selecionar o caminho para a pasta criada no passo 1 (clicando no botão identificado com “...”).
7. Na caixa de diálogo identificada por “What is the name of this project”, escrever “somador4bits”. Clicar em “Next”.
8. Na caixa de diálogo identificada por “File Name:”, clicar na caixa com “...” e selecionar os dois arquivos VHDL deste projeto (somador1bit.vhd e somador4bits.vhd). Clicar em “Add All” e depois, clicar em “Next”.

# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Passos do projeto “somador4bits” Invocando o Quartus II e Criando um Projeto (cont.)

9. Na caixa de diálogo “Device Family”, selecionar “Cyclone II”. Na lista identificada por “Available Devices”, selecionar EP2C35F672C6. Clicar em “Next”. (Ver próximo slide.)
10. Na caixa de diálogo “Simulation”, selecionar “ModelSim-Altera”. Clicar em Next.
11. Clicar em “Finish”. (Ver próximo slide.)



# Projeto de Sistemas Digitais com Ferramentas EDA

## Passos do projeto “somador4bits”

Selecionar “Cyclone II”



Selecionar “EP2C35F672C6”



Após, clicar em “Next”



New Project Wizard: Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family:  
Family: Cyclone II  
Devices: All

Target device:  
 Auto device selected by the Fitter  
 Specific device selected in 'Available devices' list

Show in 'Available device' list:  
Package: Any  
Pin count: Any  
Speed grade: Any  
 Show advanced devices  
 HardCopy compatible only

Available devices:

Name	Core v...	LEs	User I/...	Memor...	Embed...	PLL
EP2C20F484C8	1.2V	18752	315	239616	52	4
EP2C20F484I8	1.2V	18752	315	239616	52	4
EP2C20Q240C8	1.2V	18752	142	239616	52	4
EP2C35F484C6	1.2V	33216	322	483840	70	4
EP2C35F484C7	1.2V	33216	322	483840	70	4
EP2C35F484C8	1.2V	33216	322	483840	70	4
EP2C35F484I8	1.2V	33216	322	483840	70	4
EP2C35F672C6	1.2V	33216	475	483840	70	4
EP2C35F672C7	1.2V	33216	475	483840	70	4

Companion device:  
HardCopy:   
 Limit DSP & RAM to HardCopy device resources

< Back Next > Finish Cancel

# Projeto de Sistemas Digitais com Ferramentas EDA

## ▶ Passos do projeto “somador4bits”

Selecionar “ModelSim-Altera”



Após, clicar em “Next”



Specify the other EDA tools -- in addition to the Quartus II software -- used with the project.

**Design Entry/Synthesis**

Tool name: <None>

Format:

Run this tool automatically to synthesize the current design

**Simulation**

Tool name: ModelSim-Altera

Format: VHDL

Run gate-level simulation automatically after compilation

**Timing Analysis**

Tool name: <None>

Format:

Run this tool automatically after compilation

< Back   Next >   Finish   Cancel

# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Passos do projeto “somador4bits”

### Compilando um Projeto

12. No menu “Processing” (aba superior da janela do Quartus II), selecionar “Start Compilation”. (Ou clicar no triângulo roxo, na aba superior).
13. Aguardar a mensagem “Full Compilation was Succesfull” (*warnings* são normais) ou a mensagem de erros (quando houver erros no VHDL).
14. Anotar os seguintes dados mostrados na janela “Compilation Report – Flow Summary”:
  - Total combinational functions:
  - Dedicated logic registers:
15. Anotar os seguintes dados mostrados na janela “Message” (procurar pela linha que inicia por “Longest tpd from ...”):
  - tpd:
  - Source pin
  - Destination pin:

# Projeto de Sistemas Digitais com Ferramentas EDA

## ▶ Simulação “Gate-level” com Modelsim-Altera Planejando a Simulação

Iremos simular apenas 8 possíveis combinações de entradas (dentre as  $16 \times 16 \times 2 = 512$  possíveis).

cin	A	B	cout	S
0	0101	0001	0	0110
0	0101	0101	0	1010
0	0101	0111	0	1100
0	0101	1011	1	0000
1	0101	0001	0	0111
1	0101	0101	0	1011
1	0101	0111	0	1101
1	0101	1011	1	0001

# Projeto de Sistemas Digitais com Ferramentas EDA

## ▶ Simulação “Gate-level” com Modelsim-Altera

### Planejando a Simulação

Criaremos um arquivo de estímulos  
(chamando-o de `estimulos.do`)

Esboço do arquivo de estímulos:

cin	A	B	cout	S
0	0101	0001	0	0110
0	0101	0101	0	1010
0	0101	0111	0	1100
0	0101	1011	1	0000
1	0101	0001	0	0111
1	0101	0101	0	1011
1	0101	0111	0	1101
1	0101	1011	1	0001

```
force /cin 0 0 ns, 1 80 ns -r 160 ns
force /A 0101 0 ns
force /B 0001 0 ns, 0101 20 ns, 0111 40 ns, 1011 60 ns -r 80 ns
```

# Projeto de Sistemas Digitais com Ferramentas EDA

---

## ▶ Simulação “Gate-level” com Modelsim-Altera

**Seguir os passos da simulação Gate-level com o Modelsim -Altera, conforme descrito nos slides da aula 2P.**

**Atenção:**

**será necessário definir novamente o caminho do Modelsim -Altera antes de chamá-lo.**

# Introdução à Linguagem VHDL

## ► Comandos de Atribuição em VHDL

VHDL provê os seguintes comandos de atribuição:

- **Simple**
- **Sinal selecionado** (*selected signal assignment*)
- **Sinal condicional** (*conditional signal assignment*)
- **Geração** (*for generate statement*)
- **If-then-else** (*if-then-else statement*)
- **Case** (*case statement*)

Usados  
somente dentro  
de processos

# Introdução à Linguagem VHDL

## ▶ Usando Comandos de Atribuição Simples

### Multiplexador 2:1

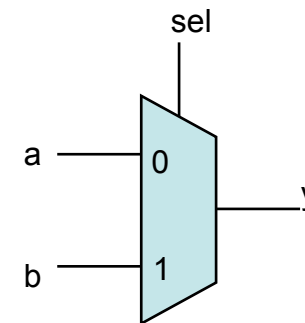
símbolo (ou esquemático)  
no nível lógico

#### Tabela-verdade

sel	A	B	y
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

#### Tabela-verdade reduzida

sel	y
0	A
1	B



#### Equação

$$y = \overline{\text{sel}} \cdot A + \text{sel} \cdot B$$



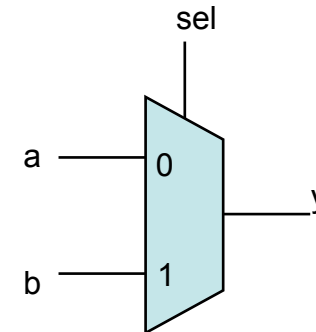
# Introdução à Linguagem VHDL

## ▶ Usando Comandos de Atribuição Simples

### Multiplexador 2:1

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY mux2para1 IS  
    PORT ( sel, a, b : IN STD_LOGIC;  
          y : OUT STD_LOGIC);  
END mux2para1;
```

sel	y
0	A
1	B



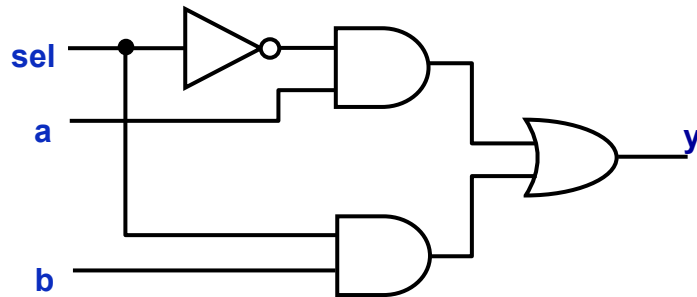
```
ARCHITECTURE comportamento OF mux2para1 IS  
BEGIN  
    y <= (a AND (NOT(sel))) OR (b AND sel);  
END comportamento;
```

# Estrutura de um FPGA Altera

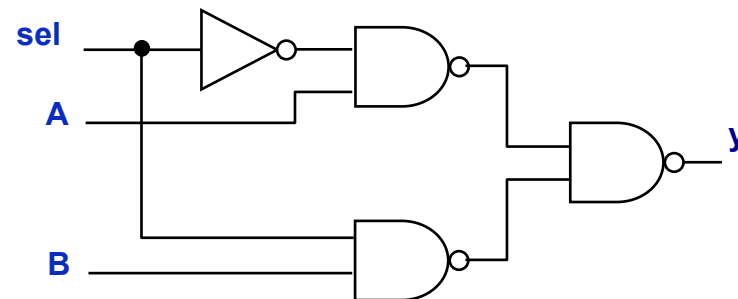
## ► Mapeamento para o FPGA

Como o mux2:1 será mapeado para o FPGA escolhido?

Será assim?



Ou assim?

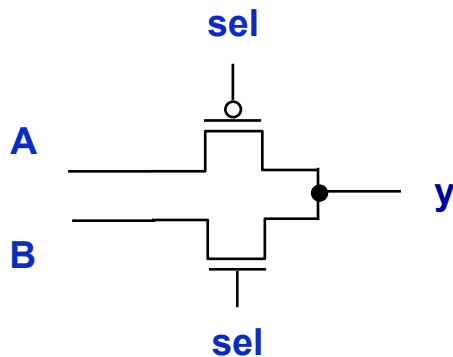


# Estrutura de um FPGA Altera

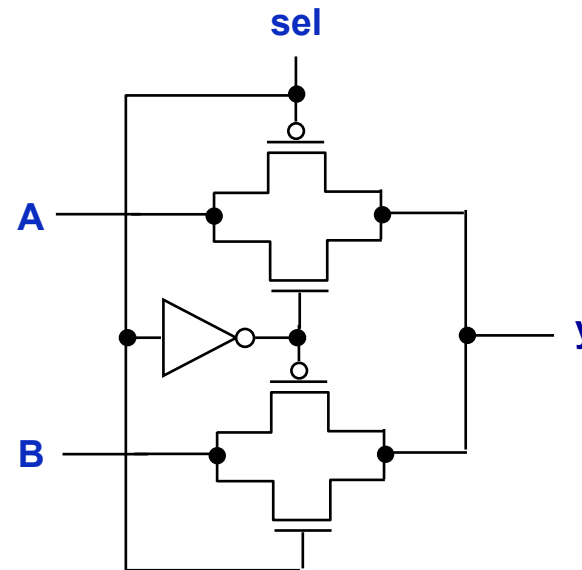
## ► Mapeamento para o FPGA

Como o mux2:1 será mapeado para o FPGA escolhido?

Ou assim?



Ou assim?



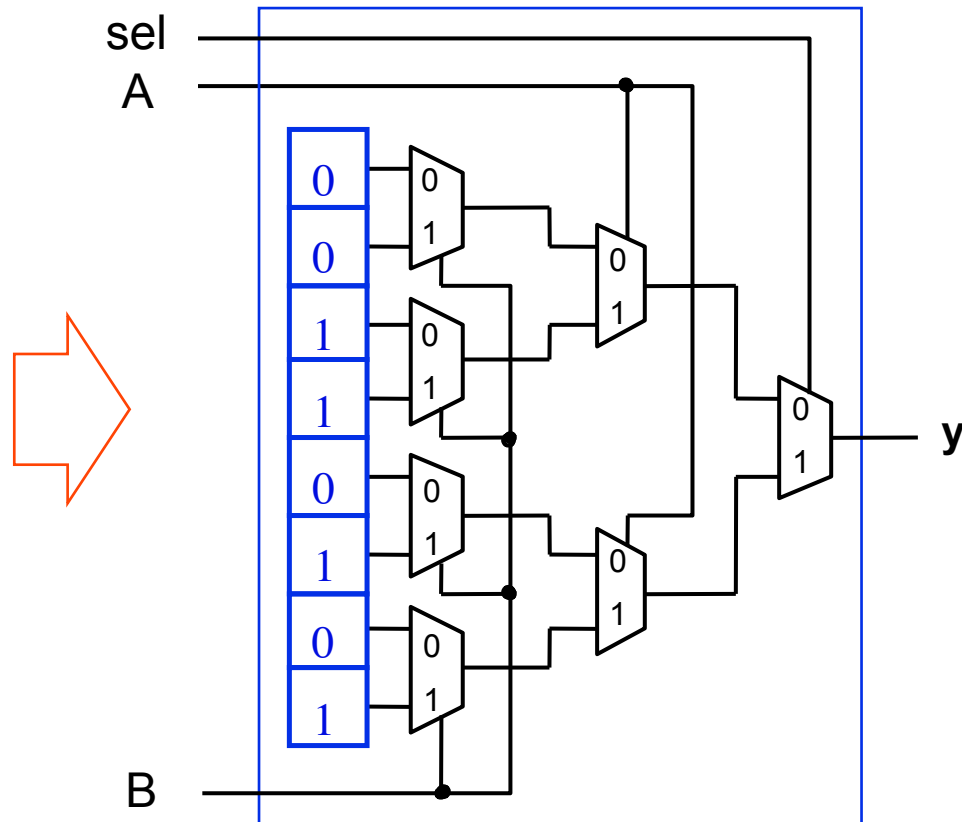
# Estrutura de um FPGA Altera

## ► Mapeamento para o FPGA

Como o mux2:1 será mapeado para o FPGA escolhido?

Na verdade, é assim...

sel	A	B	y
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

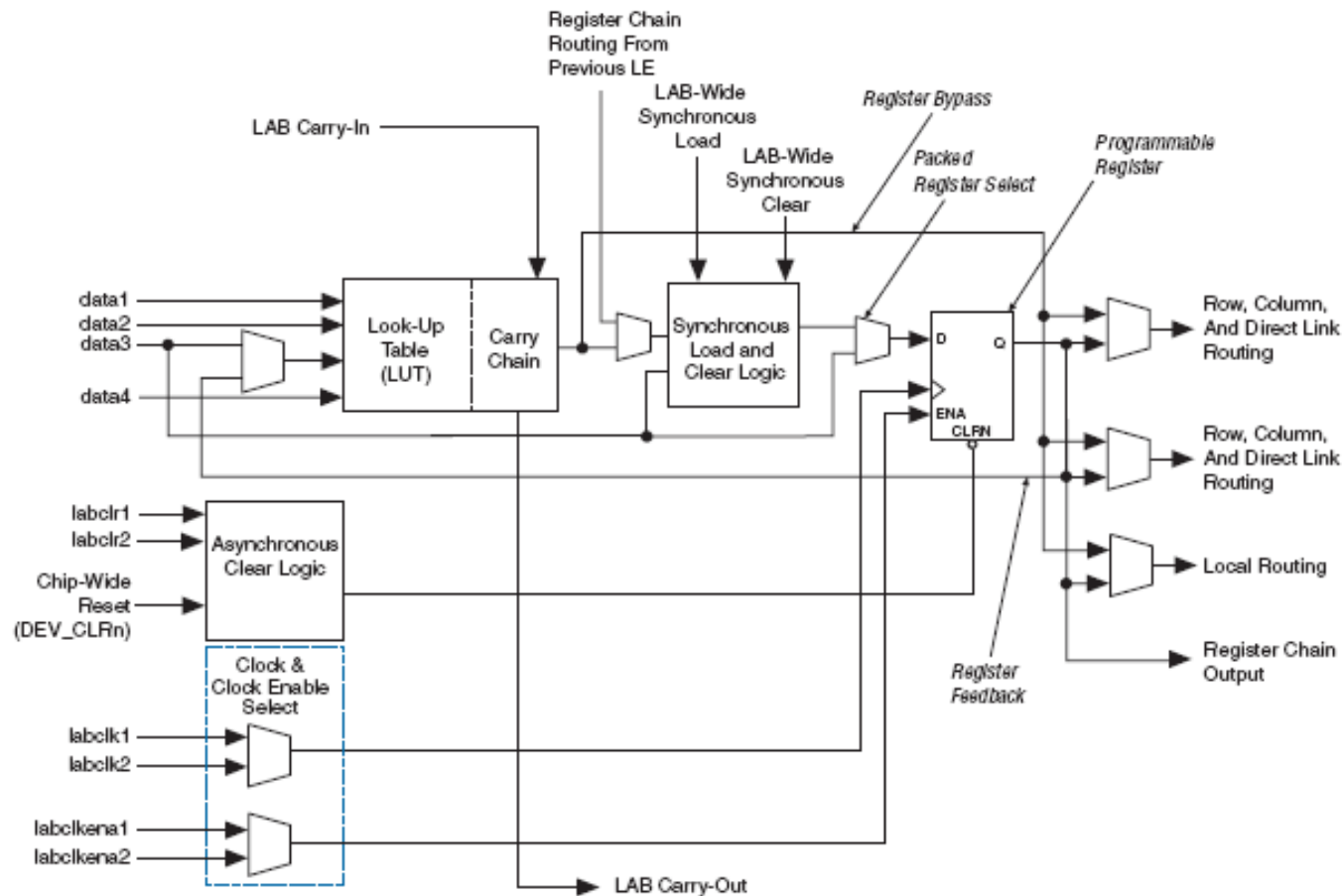




# Estrutura de um FPGA Altera

## ▶ Estrutura de um FPGA (Cyclone II)

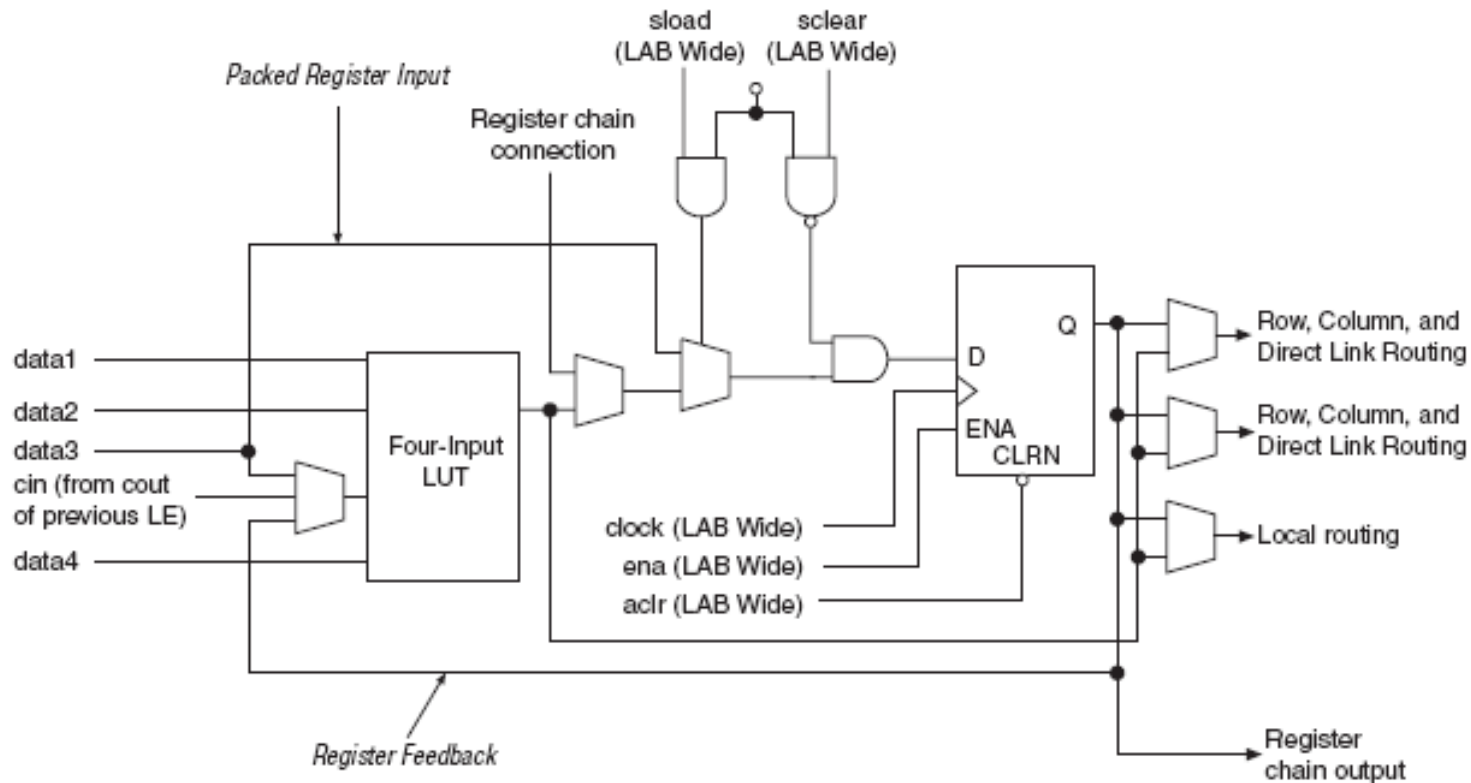
Figure 2-2. Cyclone II LE



# Estrutura de um FPGA Altera

## ► Estrutura de um FPGA (Cyclone II)

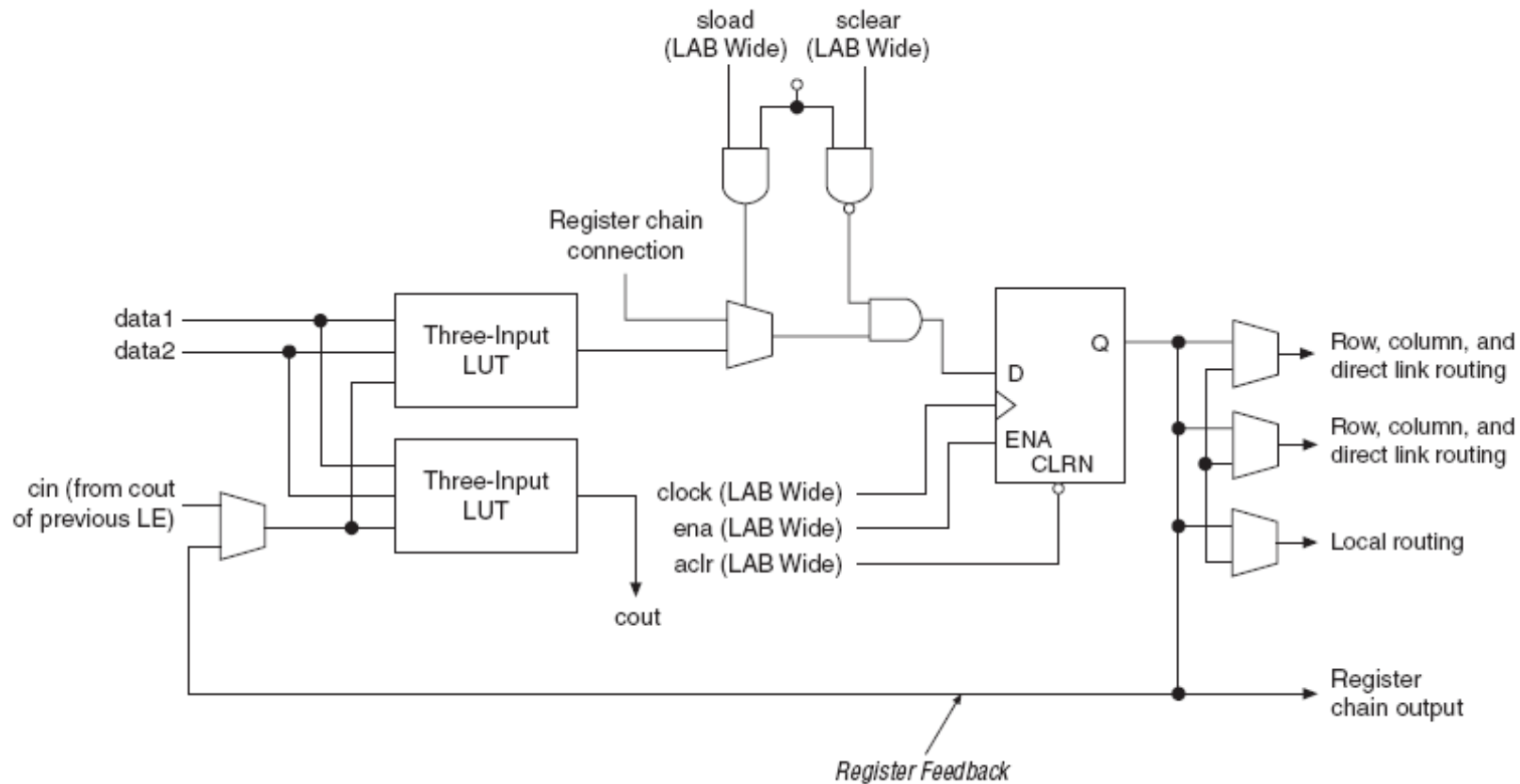
Figure 2-3. LE in Normal Mode



# Estrutura de um FPGA Altera

## ► Estrutura de um FPGA (Cyclone II)

Figure 2-4. LE in Arithmetic Mode





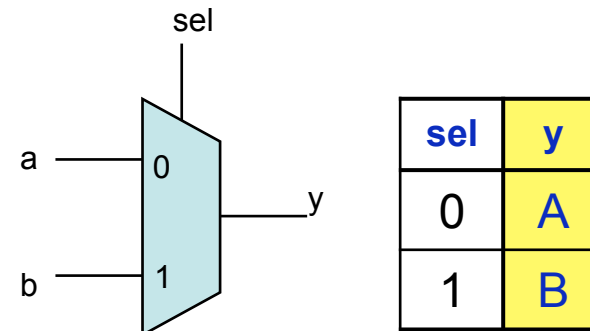
# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Selecionado

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY mux2para1 IS  
PORT ( sel, a, b : IN STD_LOGIC;  
      y : OUT STD_LOGIC);  
END mux2para1 ;
```

```
ARCHITECTURE comportamento OF mux2para1 IS  
BEGIN  
  WITH sel SELECT  
    y <= a WHEN '0',  
      b WHEN OTHERS;  
END comportamento;
```

Multiplexador 2:1



- Deve haver um “WHEN” para cada valor possível de sel
- y foi declarado como STD\_LOGIC, logo, pode valer 0, 1, Z, -

# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Selecionado

### Multiplexador 2:1 – Nível RT

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

```
ENTITY mux2para1 IS
```

```
PORT ( sel: IN STD_LOGIC;
```

```
      a, b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
```

```
      y : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
```

```
END mux2para1 ;
```

```
ARCHITECTURE comportamento OF mux2para1 IS
```

```
BEGIN
```

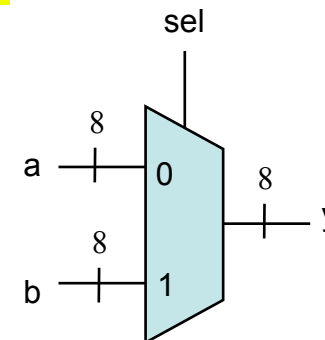
```
  WITH sel SELECT
```

```
    y <= a WHEN '0',
```

```
    b WHEN OTHERS;
```

```
END comportamento;
```

sel	y
0	A
1	B



# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Condicional

### Multiplexador 2:1 – Nível RT

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

```
ENTITY mux2para1 IS  
PORT (sel: IN STD_LOGIC;
```

```
    a, b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);  
    y : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
```

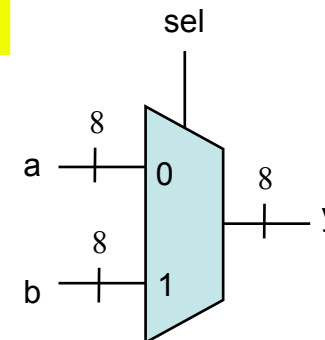
```
END mux2para1;
```

```
ARCHITECTURE comportamento OF mux2para1 IS  
BEGIN
```

```
    y <= a WHEN sel = '0' ELSE b;
```

```
END comportamento;
```

sel	Y
0	A
1	B



# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Selecionado

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

```
ENTITY mux4para1 IS  
PORT ( sel: IN STD_LOGIC_VECTOR(1 DOWNTO 0);  
      a, b, c, d : IN STD_LOGIC_VECTOR(7 DOWNTO 0);  
      y :: OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );  
END mux4para1;
```

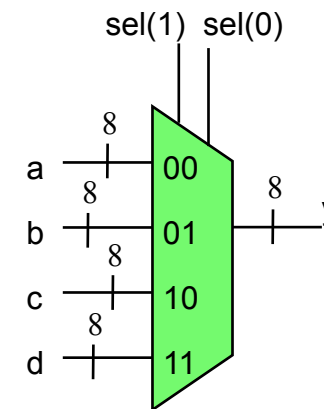
```
ARCHITECTURE comportamento OF mux4para1 IS  
BEGIN
```

```
  WITH sel SELECT  
    y <= a WHEN "00",  
      b WHEN "01",  
      c WHEN "10",  
      d WHEN OTHERS;
```

```
END comportamento;
```

INE/CTC/UFSC  
Sistemas Digitais - semestre 2011/1

### Multiplexador 4:1



sel(1)	sel(0)	y
0	0	a
0	1	b
1	0	c
1	1	d

# Introdução à Linguagem VHDL

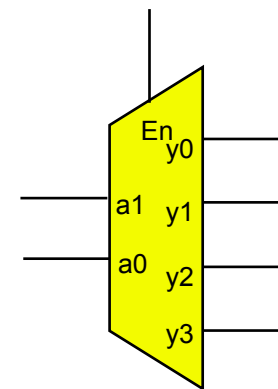
## ▶ Usando Atribuição com Sinal Seleccionado

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

ENTITY dec2para4 IS
PORT ( a : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
      En : IN STD_LOGIC;
      y : OUT STD_LOGIC_VECTOR(0 TO 3) );
END dec2para4;

ARCHITECTURE comportamento OF dec2para4 IS
  SIGNAL Ena : STD_LOGIC_VECTOR (2 DOWNTO 0);
BEGIN
  Ena <= En & a; -- concatenação dos sinais a e enable
  WITH Ena SELECT
    Y <= "1000" WHEN "100",
        "0100" WHEN "101",
        "0010" WHEN "110",
        "0001" WHEN "111",
        "0000" WHEN OTHERS;
END comportamento;
```

Decodificador 2:4



En	a1	a0	y0	y1	y2	y3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

slide

Prof. José Luís Güntzel

# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Condicional

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

ENTITY prioridade IS
PORT ( w : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      z : OUT STD_LOGIC);
END prioridade;

ARCHITECTURE comportamento OF prioridade IS
BEGIN
    y <= "11" WHEN w(3) = '1' ELSE
        "10" WHEN w(2) = '1' ELSE
        "01" WHEN w(1) = '1' ELSE
        "00";
    z <= '0' WHEN w = "0000" ELSE '1';
END comportamento;
```

Codificador de prioridade 4:2

w3	w2	w1	w0	y1	y0	z
0	0	0	0	X	X	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

Note que há uma prioridade na avaliação dos "WHENs"

**Pergunta: qual das duas atribuições acima ocorre primeiro?**

# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Condicional

### Comparador de 4 bits

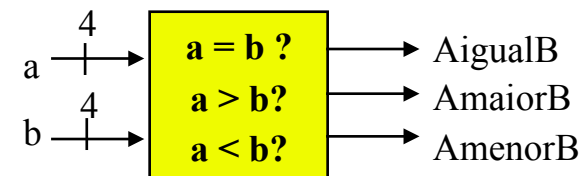
(1ª versão, para números sem sinal)

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

← Permite que sinais STD\_LOGIC\_VECTOR sejam tratados como números binários sem sinal com os operadores relacionais de VHDL

```
ENTITY comp IS  
PORT ( A, B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);  
       AigualB, AmaiorB, AmenorB : OUT STD_LOGIC);  
END comp;
```

```
ARCHITECTURE comportamento OF comp IS  
BEGIN  
    AigualB <= '1' WHEN A=B ELSE '0';  
    AmaiorB <= '1' WHEN A>B ELSE '0';  
    AmenorB <= '1' WHEN A<B ELSE '0';  
END comportamento;
```



# Introdução à Linguagem VHDL

## ▶ Usando Atribuição com Sinal Condicional

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

```
ENTITY comp IS  
  PORT ( A, B : IN SIGNED (3 DOWNTO 0);  
        AigualB, AmaiorB, AmenorB : OUT STD_LOGIC);  
END comp;
```

```
ARCHITECTURE comportamento OF comp IS  
BEGIN  
  AigualB <= '1' WHEN A=B ELSE '0';  
  AmaiorB <= '1' WHEN A>B ELSE '0';  
  AmenorB <= '1' WHEN A<B ELSE '0';  
END comportamento;
```

**Comparador de 4 bits**  
(2ª versão, para números com sinal)

