GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Introdução à Linguagem VHDL

Período 2021/1

Roberto G. Pacheco
pacheco@gta.ufrj.br

Grupo de Teleinformática e Automação – GTA/UFRJ
Programa de Engenharia Elétrica - PEE/COPPE/UFRJ
Universidade Federal do Rio de Janeiro

COPPE UFRJ

Bibliografia

- Volnei A. Pedroni, "Circuit Design and Simulation with VHDL", The MIT Press, 2nd Ed.

Sobre VHDL

- VHDL significa VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (Linguagem de Descrição de Hardware)
  - Linguagem de Descrição de Hardware vs Linguagem de Programação
  - Desenvolvida pelo Departamento de Defesa dos EUA na década de 80 e a primeira a ser padronizada pelo IEEE
  - FPGA (Field Programmable Field Array)
    - Uma placa composta de portas lógicas e elementos de memória (FFs) organizadas em formato matricial

Objetivos

- O código em VHDL permite descrever em software o comportamento e a estrutura do hardware de qualquer sistema digital

- Síntese
  - tradução do código em uma estrutura de hardware (FPGA) para implementar as funcionalidades descritas no código
    - Processo de transferir o código à FPGA

- Simulação
  - Processo de testar para garantir que o circuito sintetizado alcança as funcionalidades programadas no código
  - O simulador utilizado será o Quartus II Lite

Projeto em VHDL

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Análise teórica
- Desenvolvimento do código em VHDL
- Simulação
- Síntese

COPPE UFRJ

Observações Iniciais

- O código em VHDL não é case-sensitive
  - VHDL não diferencia letras maiúsculas e minúsculas
    - SignalIN é o mesmo que signalin
  - Geralmente, letras maiúsculas são reservadas para comandos reservados
    - AND, OR, XOR, LIBRARY, USE

- Comentários
  - Iniciados por "--"
  - Terminados pelo fim da linha, consequentemente, fim do comentário
  - Os relatórios DEVEM ser todos comentados
  - Boa prática disponibilizar os códigos desenvolvidos em um repositório, como github

Observações Iniciais

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Os códigos em VHDL possuem comandos:
  - Concorrentes:
    - Executados todos de forma paralela
  - Sequenciais:
    - Executados em sequência, assemelhando-se mais a execução de linguagem de programação

COPPE UFRJ

Comandos básicos

- Atribuição de sinais
  - A <= B
    - Cuidado com o pensamento de programação
    - A <= B; A<=C; => implica curto-circuito
    - São Fios e entradas físicas
- Comparação
  - ">", "=", "<"
- Operações Booleanas
  - AND, OR, NOT, XOR;

Comandos básicos

- Declarações Concorrentes
  - a
- Declarações Sequenciais
  - a

Estrutura Básica do Código em VHDL

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Declaração de Bibliotecas
- Entity
- Architecture

COPPE UFRJ

Declaração de Bibliotecas

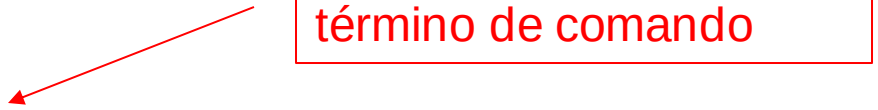
- Declaração de Bibliotecas
  - A declaração de biblioteca é uma lista de bibliotecas e pacotes que o compilador precisa para processar o projeto, ou seja, o circuito digital descrito no código

- Exemplo:
  - LIBRARY ieee;
  - USE ieee.std_logic_1164.all;

Declaração de Bibliotecas

- Declaração de Bibliotecas
  - A declaração de biblioteca é uma lista de bibliotecas e pacotes que o compilador precisa para processar o projeto, ou seja, o circuito digital descrito no código

- Exemplo:
  - LIBRARY ieee;
  - USE ieee.std_logic_1164.all;



Declaração de Bibliotecas

- Declaração de Bibliotecas
  - A declaração de biblioteca é uma lista de bibliotecas e pacotes que o compilador precisa para processar o projeto, ou seja, o circuito digital descrito no código

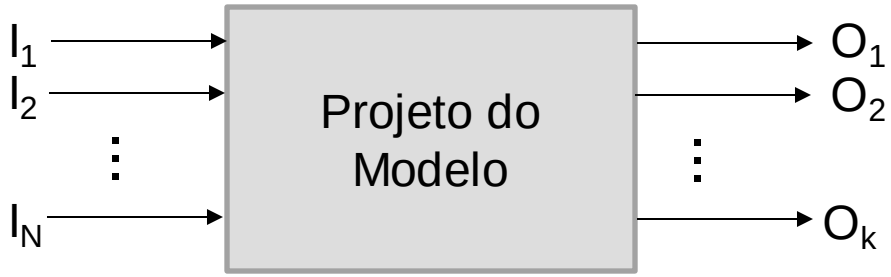
- Exemplo:
  - LIBRARY ieee;
  - USE ieee.std_logic_1164.all;
    - Geralmente essa biblioteca é importada para permitir utilizar tipos lógicos, como STD_LOGIC

Entity

- Entity é uma lista de entradas e saídas do modelo do sistema digital projetado.
  - Descreve a interface de entradas e saídas de modelo

Entity

- Entity é uma lista de entradas e saídas do modelo do sistema digital projetado.
  - Descreve a interface de entradas e saídas de modelo



Entity

- Sintaxe

```
ENTITY entity_name IS
    GENERIC (constant_name: constant_type:= constant_value;
                    ...);
     PORT(port_name : signal_mode signal_type;
      ...)
```

Entity

- Sintaxe

```
ENTITY entity_name IS
    GENERIC(constant_name: constant_type:= constant_value;
                ...);
     PORT(port_name : signal_mode signal_type;
              ...)
```

- entity_name: nome de identificação da entity/modelo
- GENERIC: define uma constante genérica que pode ser usada e alterada em qualquer momento, inclusive dentro de PORT
  - GENERIC (number_of_bits: INTEGER := 16)
- PORT: declara a interface I/O

Entity

- Sintaxe de PORT

  PORT(port_name : signal_mode signal_type;

  ...)

- Cada sinal de entrada ou saída possui um modo (IN, OUT, BUFFER, INOUT) e um tipo de sinal (BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR)

Modos do Sinal PORT

- IN: dados de entrada que fluem para dentro da ENTIDADE.
  - Não pode escrever nesses sinais
  - Ex.: clock, dados de entrada, dados de controle, entrada unidirecionais de dados

- OUT: dados de saída que fluem para fora da ENTIDADE
  - Os dados em modo OUT são usados quando não serão lidos pela ENTIDADE

- BUFFER: dados de saída que fluem para fora da ENTIDADE, mas também podem ser lidas. Na prática, esses dados são armazenadas em elemento de memória (FF)
  - Buffer pode ser utilizado para realimentação interna
  - Contudo, o BUFFER não pode ser usado como dados de entrada

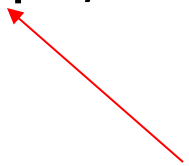
Modos do Sinal PORT

- INOUT: dados que podem fluir para dentro e fora da ENTIDADE
  - Evitem utilizar esse modo
    - Aumenta a complexidade do projeto
    - Reduz a interpretabilidade do código
    - Dificulta a detecção de erros
  - Devem ser utilizados apenas para barramento bidirecionais

Entity

- Sintaxe

```
ENTITY entity_name IS
    GENERIC (constant_name: constant_type:= constant_value;
                    ...);
     PORT(port_name : signal_mode signal_type;
      ...)
```



Tipos de sinais em VHDL

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- BIT, BIT_VECTOR:
  - Representa nível lógico de sinais de entrada ou saída
  - Valores: '0' ou '1'
  - Tipos nativos do VHDL, logo não necessita de declaração de bibliotecas

- STD_LOGIC, STD_LOGIC_VECTOR:
  - Representa nível lógico de sinais de entrada ou saída
  - Valores: '0', '1', '-' (don't case), 'Z' (alta impedância)
  - Esse tipo de sinais necessita das seguintes bibliotecas:
    - LIBRARY ieee;
    - USE ieee.std_logic_1164.all;

COPPE UFRJ

Tipos de sinais em VHDL

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- BIT, BIT_VECTOR:
  - Representa nível lógico de sinais de entrada ou saída
  - Valores: '0' ou '1'
  - Tipos nativos do VHDL, logo não necessita de declaração de bibliotecas

- STD_LOGIC, STD_LOGIC_VECTOR:
  - Representa nível lógico de sinais de entrada ou saída
  - Valores: '0', '1', '-' (don't case), 'Z' (alta impedância)
  - Esse tipo de sinais necessita das seguintes bibliotecas:
    - LIBRARY ieee;
    - USE ieee.std_logic_1164.all;

COPPE UFRJ

Tipos de sinais em VHDL

- CONSTANT:
  - Atribuição de valores constantes
  - Sintaxe: constant_name: constant_type := constant_value
    - Ex.: n_bits: INTEGER := 16
- SIGNAL:
  - Tipo de sinais que transmitem dados de dentro para fora da entidade, ou entre componentes internos do circuito, atuando como fios.
  - Funciona para "armazenar variáveis temporárias"
- VARIABLE:
  - Armazena valores na parte sequencial do VHDL

Constant vs Generic

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Constant

- Sintaxe
  - CONSTANT n: INTEGER := 8;
- Atribuição de um valor constante
  - Define-se até na Architecture (corpo do projeto)

Generic

- Sintaxe
  - GENERIC(n: INTEGER :=8);
- Atribuição de um valor constante semelhante a Constant
  - Define-se na entity
  - Pode ser alterado em qualquer momento
  - Pode ser alterado e mapeado como componente

COPPE UFRJ

Constant vs Generic

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Constant

- Sintaxe
  - CONSTANT n: INTEGER := 8;
- Atribuição de um valor constante
  - Define-se até na Architecture (corpo do projeto)

Generic

- Sintaxe
  - GENERIC(n: INTEGER :=8);
- Atribuição de um valor constante semelhante a Constant
  - Define-se na entity
  - Pode ser alterado em qualquer momento
  - Pode ser alterado e mapeado como componente

COPPE UFRJ

Tipos de sinais em VHDL

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Predefined data types	Library / Package of origin	Synthesizable values without restrictions
BIT, BIT_VECTOR	std / standard	'0', '1'
BOOLEAN	std / standard	TRUE, FALSE
INTEGER	std / standard	$-(2^{31}-1)$ to $+(2^{31}-1)$
NATURAL	std / standard	0 to $+(2^{31}-1)$
POSITIVE	std / standard	1 to $+(2^{31}-1)$
CHARACTER	std / standard	256-symbol alphabet (1 byte/symbol)
STRING	std / standard	Set of characters
REAL	std / standard	Little or no synthesis support
STD_(U)LOGIC, STD_(U)LOGIC_VECTOR	ieee / std_logic_1164	Input: '0' or 'L', '1' or 'H' Output: '0' or 'L', '1' or 'H', '-' or 'X', and 'Z'
UNSIGNED, SIGNED	ieee / numeric_std, ( ) / std_logic_arith	Same as STD_LOGIC

COPPE UFRJ

Exemplos – Contador de 4 bits

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```

Exemplos – Contador de 4 bits

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```

A última não precisa do ";"

COPPE UFRJ

Exemplos – Contador de 4 bits

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```

END entity_name;

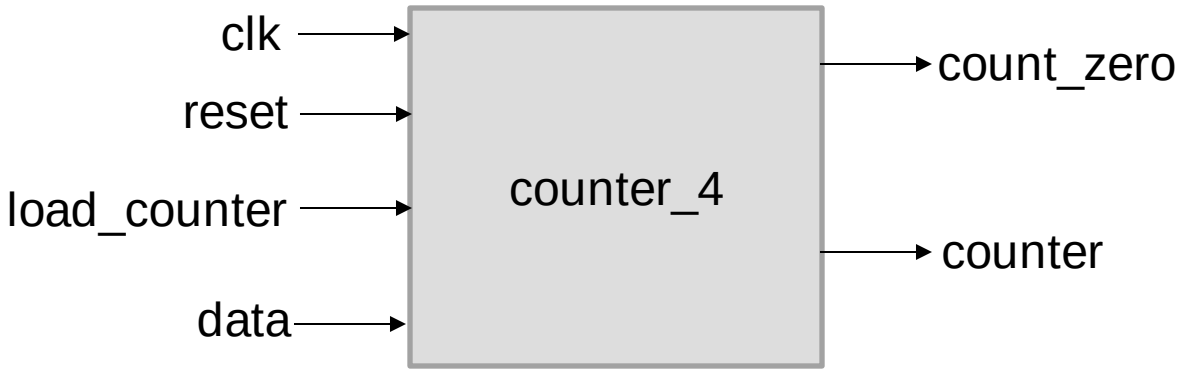
Exemplos – Contador de 4 bits

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```

CUIDADO: não misture BIT com STD_LOGIC ou BIT_VECTOR com STD_LOGIC_VECTOR

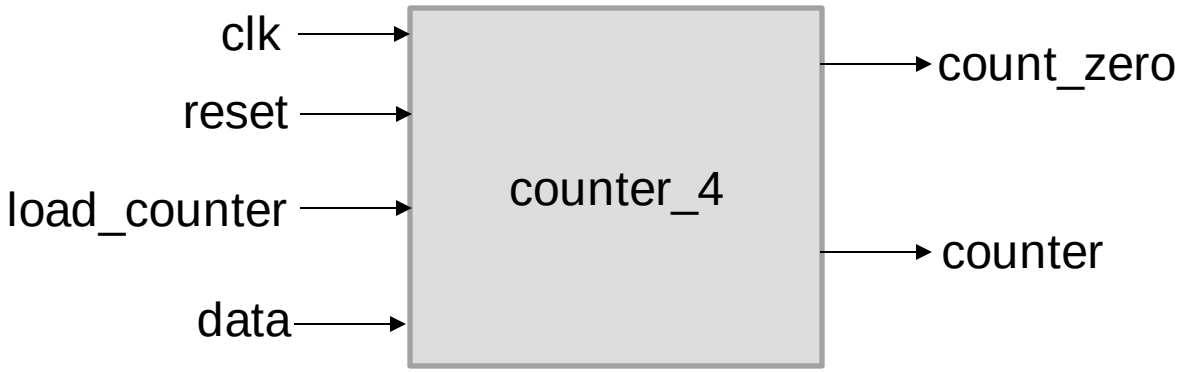
Exemplos – Contador de 4 bits

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```

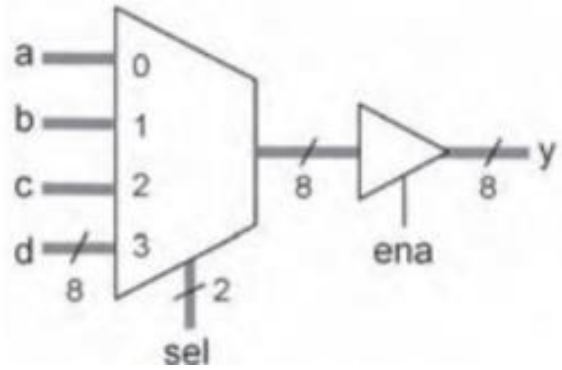


Exemplos – Contador de 4 bits

```
ENTITY counter_4 IS PORT (
    clk, reset, load_counter:   IN BIT;
    data:                       IN BIT_VECTOR(3 DOWNTO 0);
    count_zero:                 OUT BIT;
    counter:                    BUFFER BIT_VECTOR(3 DOWNTO 0)
    );
END counter_4;
```



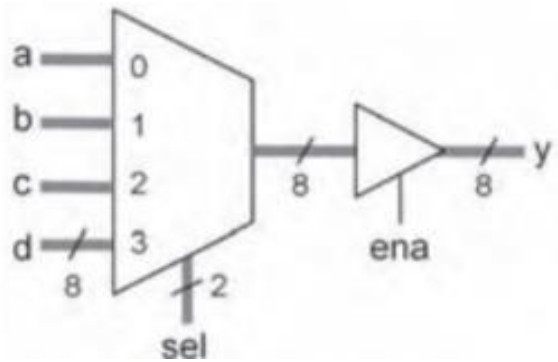
Exemplos – Buffered MUX 8 bits



ena	sel	y
0	X	"ZZZZZZZZ"
1	0	a
	1	b
	2	c
	3	d

Exemplos – Buffered MUX 8 bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY buffered_mux IS PORT (
    a, b, c, d:          IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    sel:                 IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    ena:                 INT BIT;
    y:                   OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END buffered_mux;
```

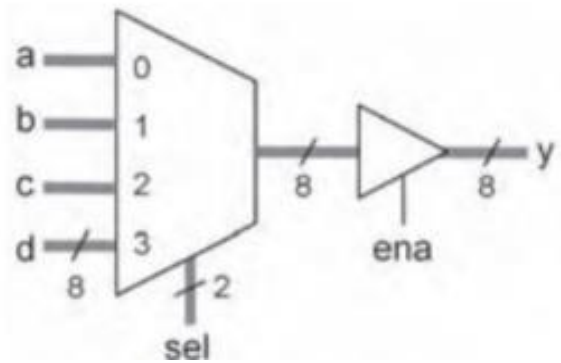


ena	sel	y
0	X	"ZZZZZZZZ"
1	0	a
	1	b
	2	c
	3	d

Exemplos – Buffered MUX 8 bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY buffered_mux IS PORT (
    a, b, c, d:          IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    sel:                 IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    ena:                 INT BIT;
    y:                   OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END buffered_mux;
```

Por que não usar BIT_VECTOR?

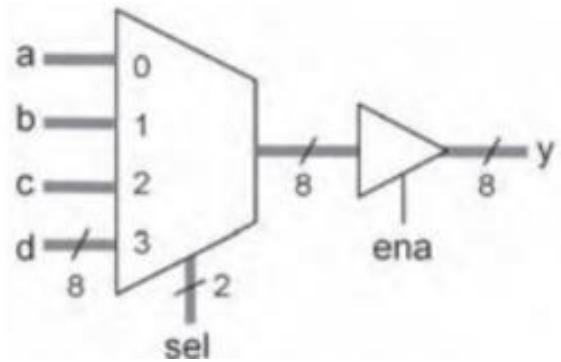


ena	sel	y
0	X	"ZZZZZZZZ"
1	0	a
	1	b
	2	c
	3	d

Exemplos – Buffered MUX 8 bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY buffered_mux IS PORT (
    a, b, c, d:          IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    sel:                 IN STD_LOGIC_VECTOR(1 DOWNTO 0);
    ena:                 INT BIT;
    y:                   OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END buffered_mux;
```

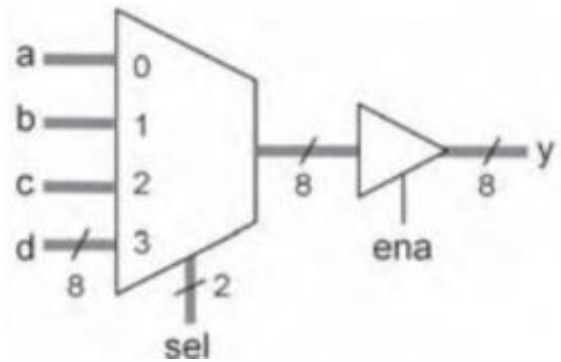
Note a necessidade da alta impedância na tabela-verdade



ena	sel	y
0	X	"ZZZZZZZZ"
1	0	a
	1	b
	2	c
	3	d

Exemplos – Buffered MUX 8 bits

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY buffered_mux IS PORT (
    a, b, c, d:          IN STD_LOGIC_VECTOR (7 DOWNTO 0);
    sel:                 IN INTEGER RANGE 0 TO 3;
    ena:                 INT BIT;
    y:                   OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END buffered_mux;
```



ena	sel	y
0	X	"ZZZZZZZZ"
1	0	a
	1	b
	2	c
	3	d

Vetores

- BIT_VECTOR:
  - Declaração:
    - bit_signal_vector: BIT_VECTOR (max_index DOWNTO 0);
    - bit_vector1, bit_vector2 : BIT_VECTOR (3 DOWNTO 0);
    - bit_standalone: BIT;

- Formas de atribuição
  - bit_vector1(0) <= '1';
  - bit_vector1(0) <= bit_standalone;
  - bit_vector1(0) <= bit_vector2(3);
  - bit_vector1 <= "0101"

Vetores

- BIT_VECTOR:
  - Declaração:
    - bit_signal_vector: BIT_VECTOR (max_index DOWNTO 0);
    - bit_vector1, bit_vector2: BIT_VECTOR (3 DOWNTO 0);
    - bit_standalone: BIT;

- Formas de atribuição
  - bit_vector1(0) <= '1';
  - bit_vector1(0) <= bit_standalone;
  - bit_vector1(0) <= bit_vector2(3);
  - bit_vector1 <= "0101"

Aspas simples para atribuir 1 bit

Aspas dupla para atribuir conjunto de bits

Operadores

- Operadores de Atribuição ("<=", ":=")
  - "<=" usado para atribuir valores à sinais
  - ":=" usado para atribuir valores à constantes, variables

- Operadores de concatenação ("&", ",")
  - k: CONSTANT BIT_VECTOR(1 TO 4) := "1100";
  - x <= ('Z', k(2 TO 3), "11111");
    - x <= 'Z1011111'
  - x <= 'Z' & k(2 TO 3) & "11111";
    - x <= 'Z1011111'

Operadores

- Operadores Lógicos (NOT, AND, NAND, OR, NOR, XOR, XNOR)
  - Exemplos:
    - x <= a NAND b; -- x <= (a AND b)'
    - y <= NOT (a AND b); -- y <= (a AND b)'
    - z <= NOT a AND b; -- z <= a' AND b;

- Operadores Aritméticos (+, –, *, /, **, ABS, REM, MOD)
  - Exemplos:
    - x <= (a + b)**N;
    - y <= ABS(a) + ABS(b);
    - z <= a/(a + b);

Operadores

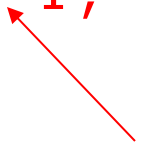
- Operadores de deslocamento (SLL, SRL, SLA, SRA, ROL, ROR)
  - SLL (shift left logical): Data are shifted to the left with '0's in the empty positions
  - SRL (shift right logical): Data are shifted to the right with '0's in the empty positions
  - SLA (shift left arithmetic): Data are shifted to the left with the rightmost bit in the empty positions
  - SRA (shift right arithmetic): Data are shifted to the right with the leftmost bit in the empty positions
  - ROL (rotate left): Circular shift to the left
  - ROR (rotate right): Circular shift to the right

Operadores

- Operadores de deslocamento (SLL, SRL, SLA, SRA, ROL, ROR)
  - Exemplos:
    - a <= "11001";
    - x <= a SLL 2; -- result: x <= "00100"
    - y <= a SLA 2; --result: y <= "00111"
    - z <= a SLL –3; --result: z <= "00011"

Operadores

- Operadores de Comparação (=, /=, >, <, >=, <=)
  - Exemplos:
    - IF a >= b THEN x <= '1';

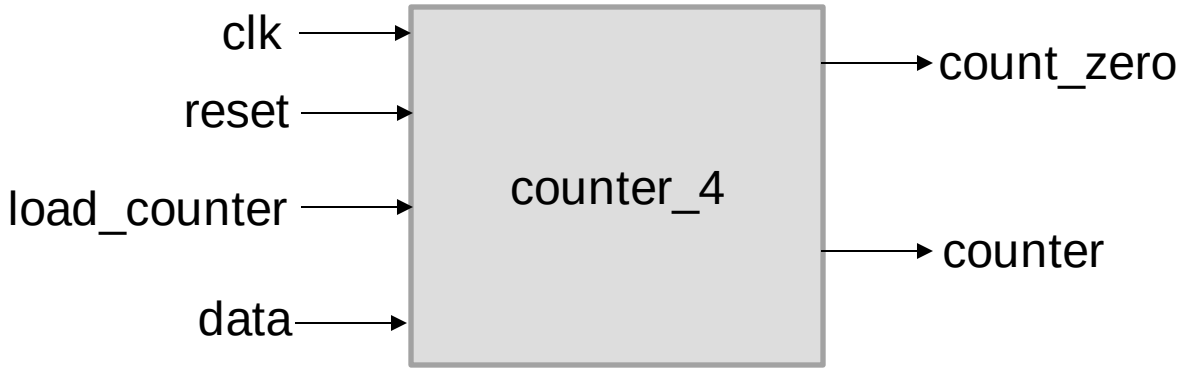


Note "<=" pode significar atribuição de sinal, mas também pode significar comparação, dentro do contexto de IF THEN

Architecture

- Descreve o comportamento e as funcionalidades do sistema

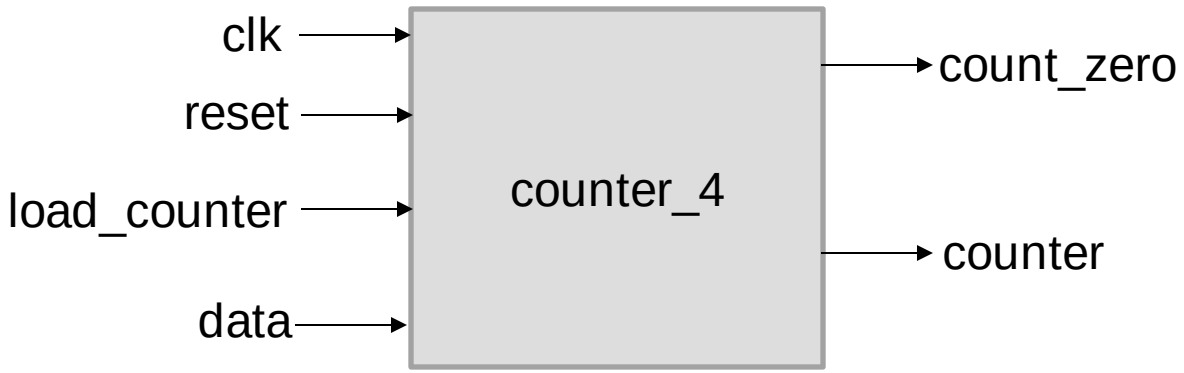
- Enquanto a Entity descreve a interface



Architecture

- Descreve o comportamento e as funcionalidades do sistema

- Enquanto a Entity descreve a interface
  - Architecture descreve o funcionamento do interior da entidade



Exemplo Completo – Maioria de 3

- Exercício:
  - Projete um circuito que apresenta saída ativa em nível alto, quando a maioria das suas três entradas estiverem em nível alto.

Exemplo Completo – Maioria de 3

- Exercício:
  - Projete um circuito que apresenta saída ativa em nível alto, quando a maioria das suas três entradas estiverem em nível alto.
    - Entradas:
      - a, b, c
    - Saída
      - y
  - Após fazer o projeto de circuito combinacional, obtém-se:
    - y <= (a AND b) OR (a AND c) OR (b AND c);

Exemplo Completo – Maioria de 3

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
Entity majority_3 IS PORT(
                        a, b, c :   IN STD_LOGIC;
                         y:          OUT STD_LOGIC
                    );
end majority_3;
ARCHITECTURE arq_maj_3 OF majority_3 IS
BEGIN
        y <= (a AND b) OR (a AND c) OR (b AND c);
END arq_maj_3;
```

Declaração de bibliotecas

Bloco de Entity

Bloco de Architecture

Exemplo Completo – Half Adder

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY half_adder IS PORT (
                        a, b:       IN  STD_LOGIC;
                        sum, carry_out:     OUT STD_LOGIC
                        );
end half_adder;
ARCHITECTURE HA of half_adder IS
BEGIN
      sum <= a XOR b;
      carry_out <= a AND b;
END HA;
```

Declaração de bibliotecas

Bloco de Entity

Bloco de Architecture

Exemplo Completo – Half Adder

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY half_adder IS PORT (
                        a, b:       IN  STD_LOGIC;
                        sum, carry_out:     OUT STD_LOGIC
                        );
end half_adder;
ARCHITECTURE HA of half_adder IS
BEGIN
      sum <= a XOR b;
      carry_out <= a AND b;
END HA;
```

Declaração de bibliotecas

Bloco de Entity

COMANDOS CONCORRENTES

Bloco de Architecture

Exemplo Completo – Full Adder

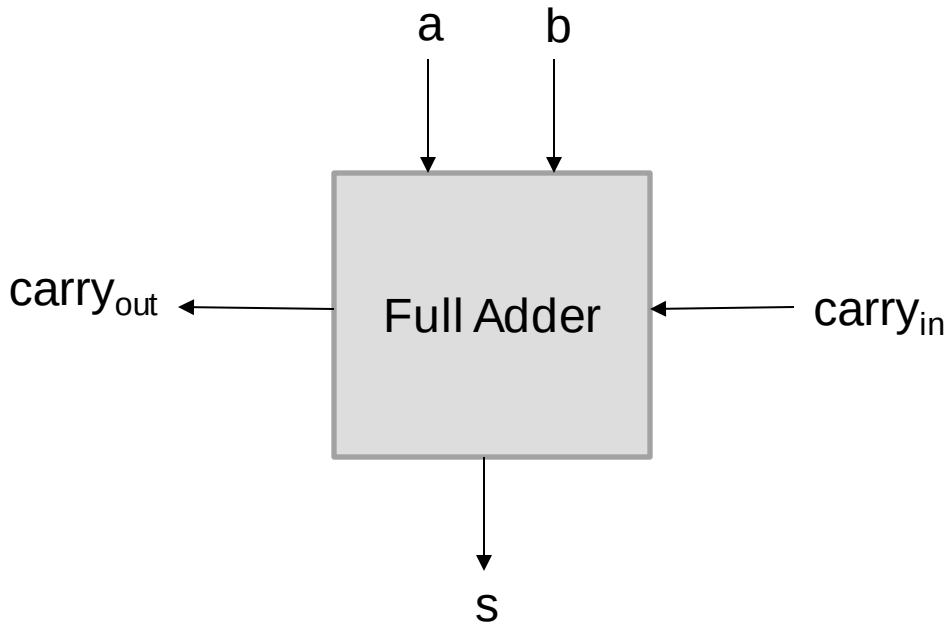
```
ENTITY full_adder IS
        PORT(a, b, carry_in:        IN    BIT;
                sum, carry_out:     OUT BIT;
END full_adder;
```

Exemplo Completo – Full Adder

Projetar um circuito somador completo

Exemplo Completo – Full Adder

Projetar um circuito somador completo (full adder) de 1 bit



Entity do Full Adder

```
ENTITY full_adder IS
        PORT(a, b, c_in:        IN BIT;
             s, c_out:          OUT BIT);
END full_adder;
```

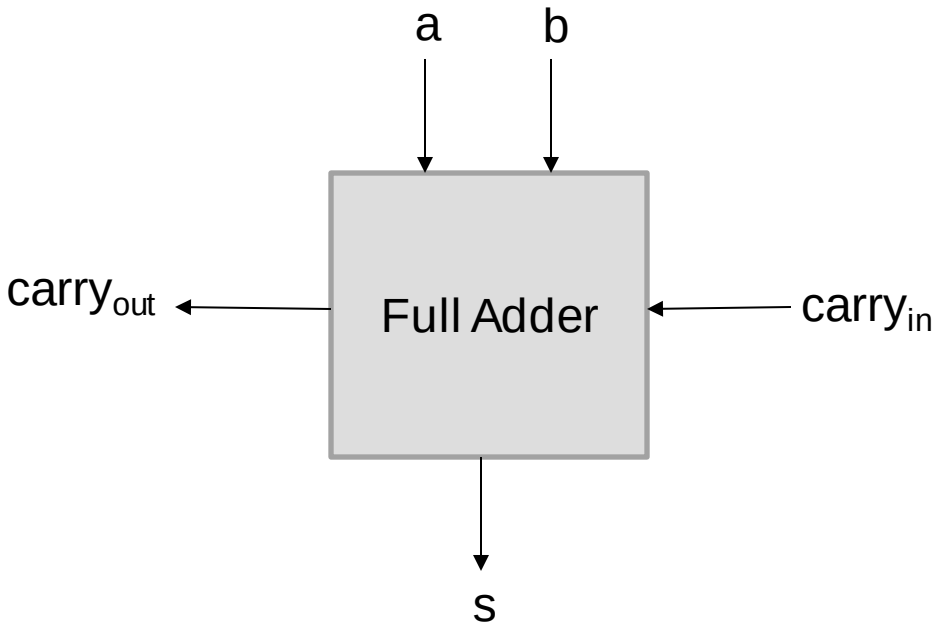


Tabela-Verdade do Full Adder

Tipos de sinais em VHDL

A implementação da Architecture pode ser realizada de diversos modos conforme o desenvolvedor. Destaca-se três tipos:

- Fluxo de dados (dataflow)
- Estrutural (structural)
- Comportamental (behavorial)

Tipos de sinais em VHDL

A implementação da Architecture pode ser realizada de diversos modos conforme o desenvolvedor. Destaca-se três tipos:

- Fluxo de dados (dataflow)
- Estrutural (structural)
- Comportamental (behavorial)

Architecture do Full Adder: Dataflow

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

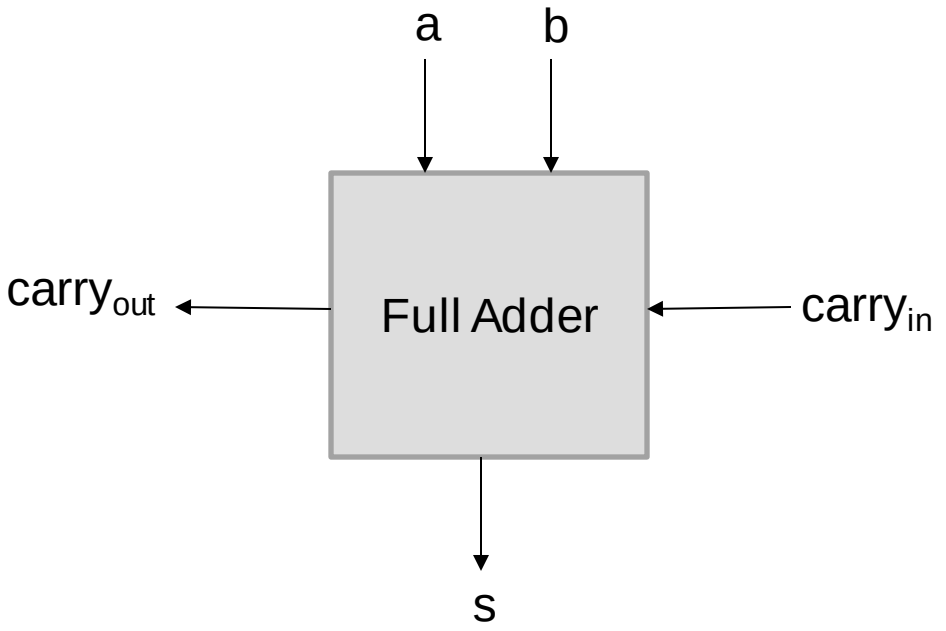
- Descreve o fluxo de dados do sistema
- As saídas são atribuídas diretamente, por meio de portas lógicas.
- Todas as expressões são concorrentes no tempo, ou seja, as atribuições ocorrem simultaneamente

COPPE UFRJ

Architecture do Full Adder: Dataflow

```
ENTITY full_adder IS
        PORT(a, b, c_in:        IN BIT;
             s, c_out:          OUT BIT);
END full_adder;

ARCHITECTURE dataflow OF full_adder IS
SIGNAL x1, x2, x3, x4: BIT;
BEGIN
x1<= b XOR c_in;
s<= a XOR x1;
x2 <= a AND b;
x3 <= a and c_in;
x4 <= b AND c_in;
c_out <= x2 OR x3 OR x4;
END dataflow;
```

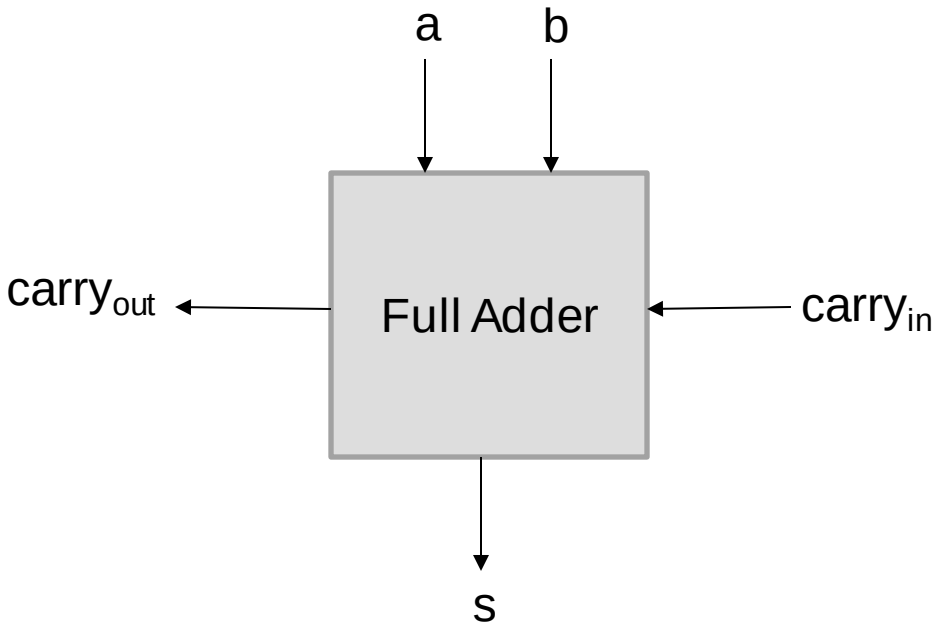


Architecture do Full Adder: Dataflow

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
ENTITY full_adder IS
        PORT(a, b, c_in:        IN BIT;
             s, c_out:          OUT BIT);
END full_adder;

ARCHITECTURE dataflow OF full_adder IS
BEGIN
s<= a XOR B XOR c_in;
c_out <= (a AND b) OR (a AND c_in) OR (b AND c_ind);
END dataflow;
```



COPPE UFRJ

Architecture do Full Adder: Dataflow

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Eliminando os sinais temporários SIGNAL

```
ENTITY full_adder IS
        PORT(a, b, c_in:        IN BIT;
                s, c_out:       OUT BIT);
END full_adder;

ARCHITECTURE dataflow OF full_adder IS
BEGIN
s<= a XOR B XOR c_in;
c_out <= (a AND b) OR (a AND c_in) OR (b AND c_ind);
END dataflow;
```


COPPE UFRJ

Architecture: Dataflow

Além disso, é possível descrever uma arquitetura dataflow usando comandos condicionais WHEN - ELSE

- Sintaxe:
  - assignment WHEN conditions ELSE

- Exemplo:

```
x <= a WHEN sel=0 ELSE
b WHEN sel=1 ELSE
c;
```

Caso default

Exemplo

Além disso, é possível descrever uma arquitetura dataflow usando comandos condicionais WHEN - ELSE

- Sintaxe:
  - assignment WHEN conditions ELSE

- Exemplo:

```
x<= "00" WHEN (a=b)
  ELSE "01" WHEN (a=c)        --  a !=b
  ELSE "10" WHEN (a=b)   -- a != b and a!=c
  ELSE "11";                  --default case
```

Exemplo

Projete um circuito que obtenha saída em nível alto quando as duas entradas forem iguais

Exemplo

Projete um circuito que obtenha saída em nível alto quando as duas entradas forem iguais

- Tabela-Verdade:

a	b	
0	0	1
0	1	0
1	0	0
1	1	1

Exemplo

Projete um circuito que obtenha saída em nível alto quando as duas entradas forem iguais
Vamos usar a estrutura de condicional WHEN-ELSE

```
ENTITY detect_equal IS
        PORT(a, b:    IN BIT;
         y: OUT BIT);
END detect_equal;
ARCHITECTURE dataflow OF detect_equal IS
BEGIN
      y<= '1' WHEN a=b ELSE
              '0';
END dataflow;
```

- Tabela-Verdade:

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

Exemplo

Projete um circuito que obtenha saída em nível alto quando as duas entradas forem iguais
Vamos usar a estrutura de condicional WHEN-ELSE

```
ENTITY detect_equal IS
        PORT(a, b:    IN BIT;
         y: OUT BIT);
END detect_equal;
ARCHITECTURE dataflow OF detect_equal IS
BEGIN
      y<= a XNOR b;
END dataflow;
```

- Tabela-Verdade:

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

Tipos de sinais em VHDL

A implementação da Architecture pode ser realizada de diversos modos conforme o desenvolvedor. Destaca-se três tipos:

- Fluxo de dados (dataflow)
- Estrutural (structural)
- Comportamental (behavorial)

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: Detector de Overflow:

O Overflow ocorre sempre que o resultado da adição está fora da faixa de números inteiros representáveis. Quando ocorre, o resultado está INCORRETO, de forma que é necessário detectar essa situação de overflow

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: Detector de Overflow:

Considere z = x+y, onde $x=x_{n-1}...x0$, $y=y_{n-1}...y_0$ que resulta em $z=z_{n-1}...z_0$

$$v = x_{n-1}'y_{n-1}'z_{n-1} + x_{n-1}y_{n-1}z_{n-1}'$$

Onde v=1 significa que ocorre overflow e v=0 indica que não ocorreu overflow.

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: Detector de Overflow:

$$v = \underbrace{x_{n-1}'y_{n-1}'z_{n-1}}_{G1} + \underbrace{x_{n-1}y_{n-1}z_{n-1}'}_{G2}$$

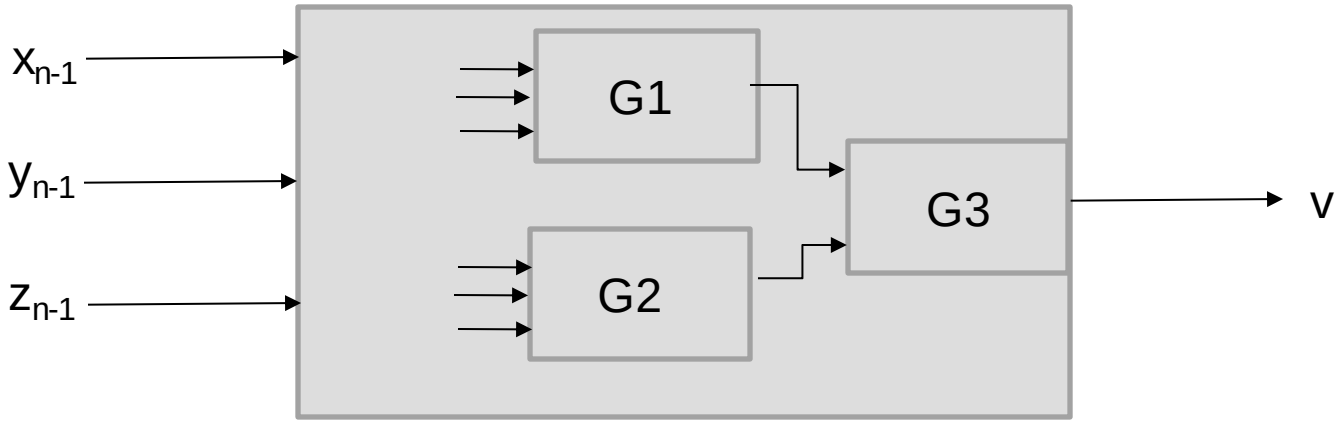
$$\underbrace{\qquad\qquad\qquad\qquad}_{G3}$$

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: Detector de Overflow:

$$v = x_{n-1}'y_{n-1}'z_{n-1} + x_{n-1}y_{n-1}z_{n-1}'$$



Arquitetura Estrutural

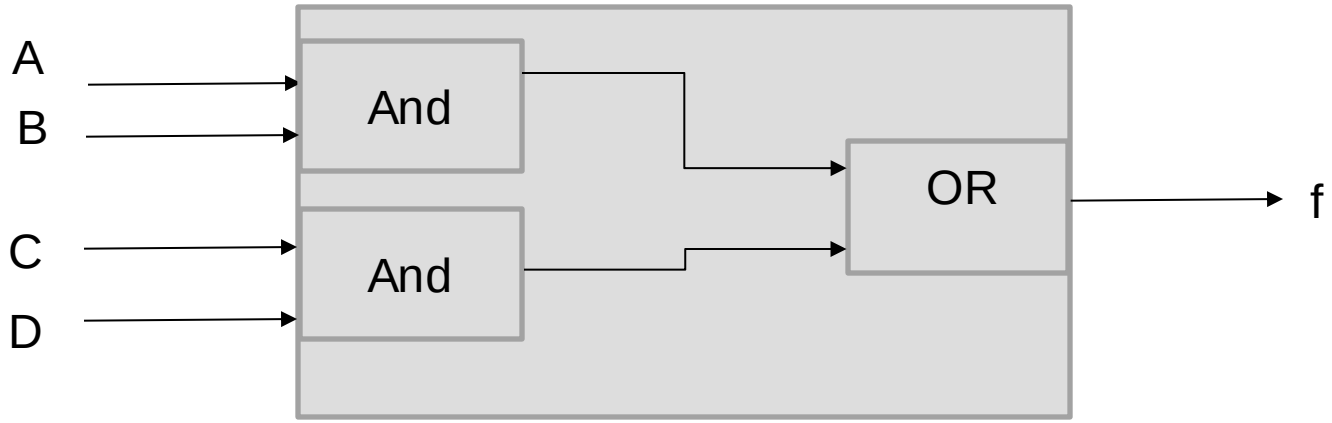
Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: $f = A.B + C.D$

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Exemplo: f = A.B + C.D



Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Sintaxe:

A sintaxe é marcada pela presença de dois compontenes:

Arquitetura Estrutural

Esse tipo de arquitetura descreve os componentes do sistema digital e suas interconexões. Nessa arquitetura, a atribuição de sinais é realizada a partir do mapeamento de entradas e saídas de cada componente

Sintaxe:

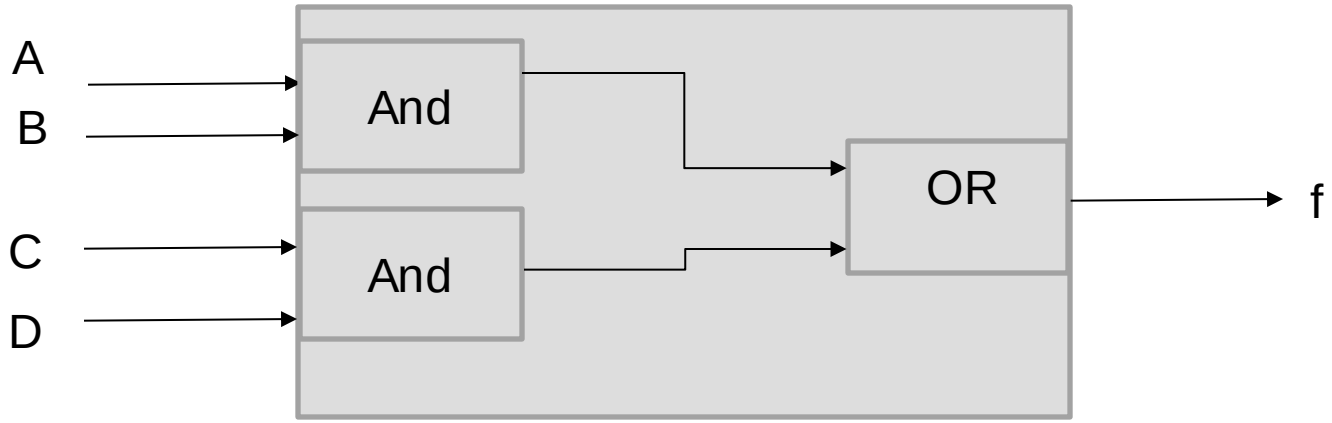
A sintaxe é marcada pela presença de dois componentes:

Component: descreve a interface do componente, ou seja, funciona como a ENTITY na construção de componente

Port map: faz o mapeamento dos sinais de entrada e saída.

Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural



Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
ENTITY and_gate IS
    Port(a, b: in bit;
         s:  out bit);
End and_gate;
Architecture dataflow of and_gate IS
Begin
    s <= a AND b;
End dataflow;
```

Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
ENTITY and_gate IS
    Port(a, b: in bit;
         s:  out bit);
End and_gate;
Architecture dataflow of and_gate IS
Begin
    s <= a AND b;
End dataflow;
```

Geralmente, esses COMPONENTES ficam em outros arquivos VHDL dentro do mesmo projeto

Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

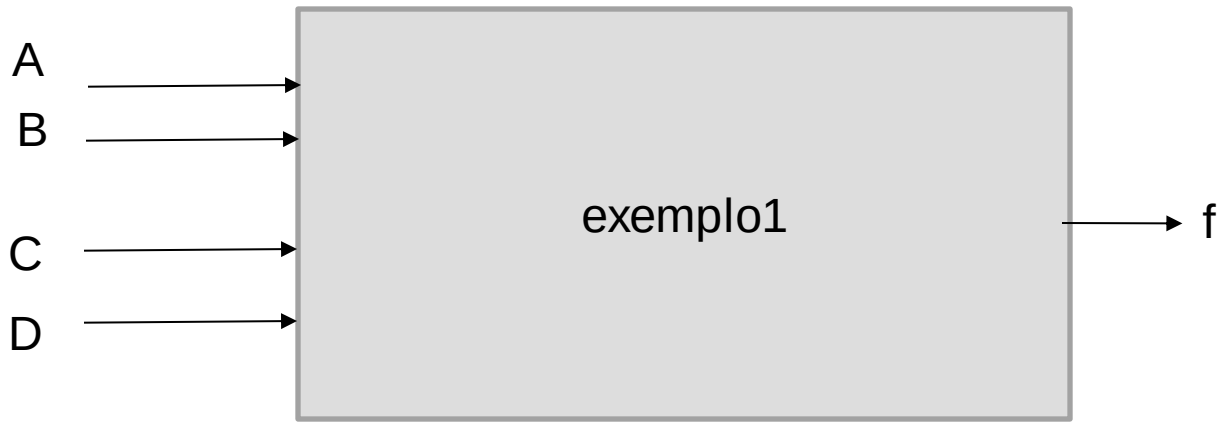
```
ENTITY or_gate IS
    Port(a, b: in bit;
         s:  out bit);
End or_gate;
Architecture dataflow of or_gate IS
Begin
    s <= a OR b;
End dataflow;
```

Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
ENTITY exemplo1 IS
   Port (a, b, c, d: in bit;
         f: out bit);
end exemplo1;
```

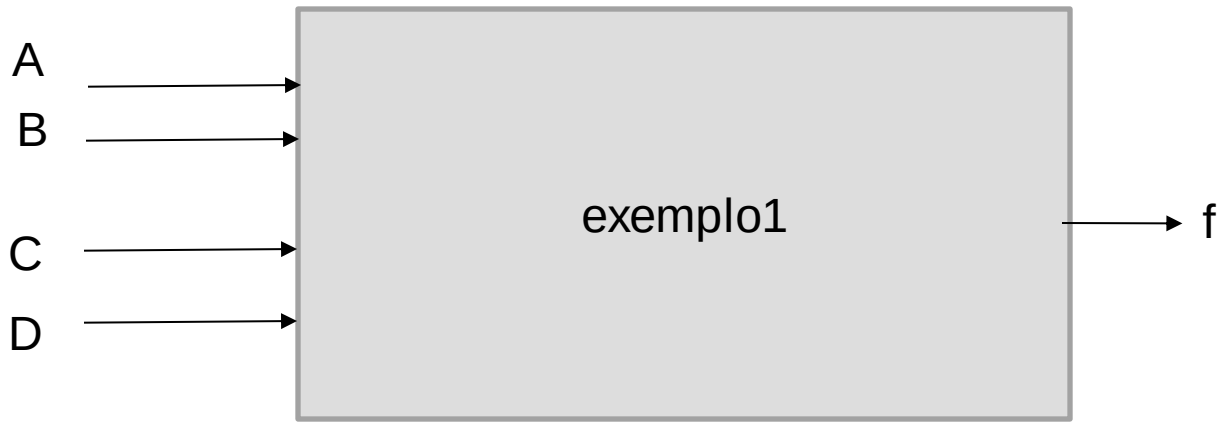


Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
ENTITY exemplo1 IS
   Port (a, b, c, d: in bit;
         f: out bit);
end exemplo1;
-- Vamos criar os compoentes
```

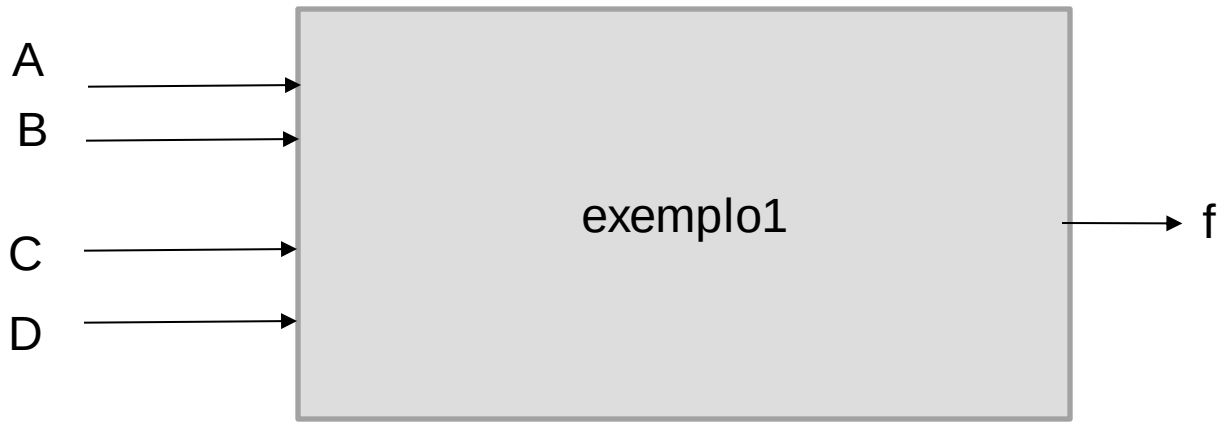


Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
ENTITY exemplo1 IS
   Port (a, b, c, d: in bit;
         f: out bit);
end exemplo1;
Component and_gate
    Port(a,b: in bit;
         s: out bit);
End and_gate;
```



Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Component or_gate
    Port(a,b: in bit;
         s: out bit);
End or_gate;
```

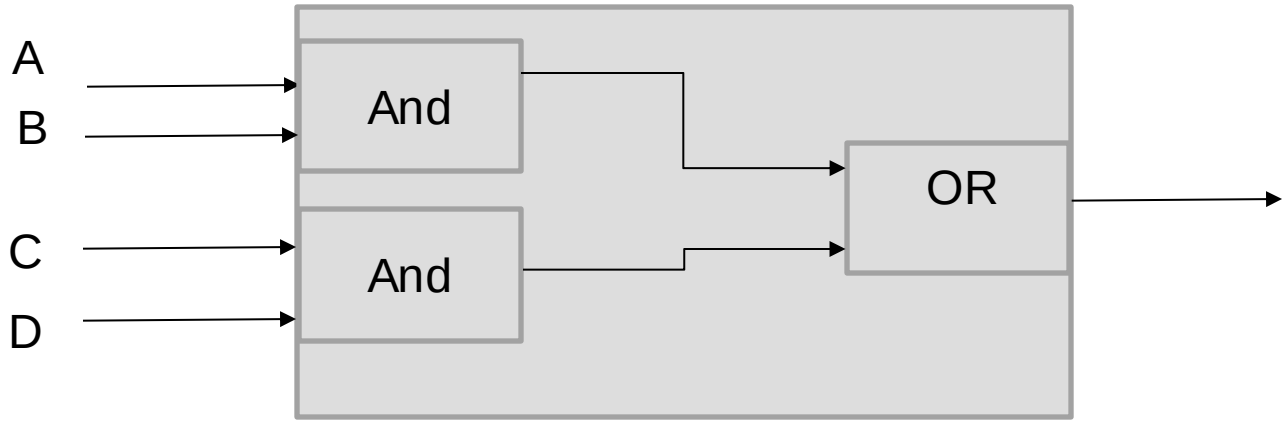
Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Architecture structural of exemplo1 is
SIGNAL x1, x2: bit;
BEGIN
    -- Mapeamento dos componentes criados

End structural;
```



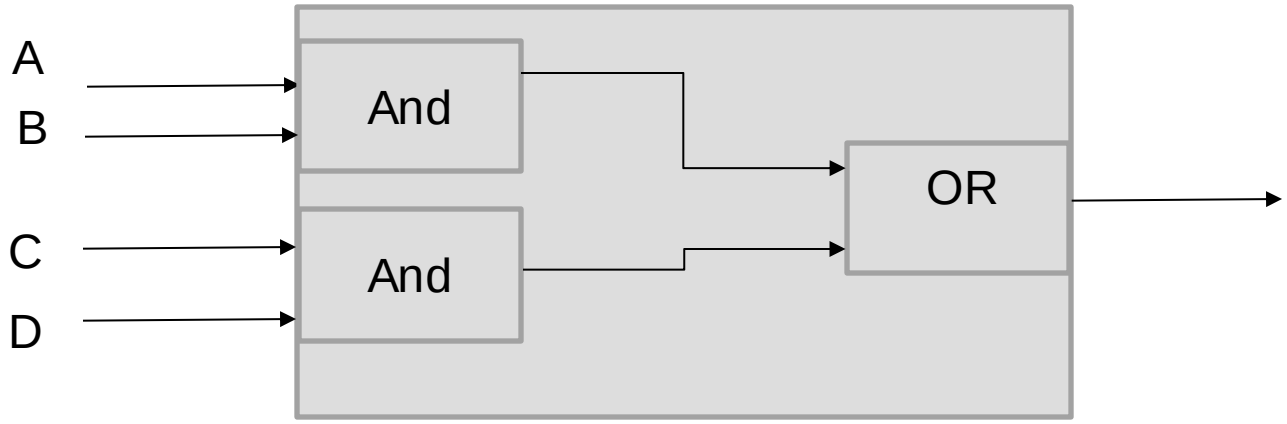
Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Architecture structural of exemplo1 is
SIGNAL x1, x2: bit;
BEGIN
    And_0: PORT MAP(a, b, x1);
    And_1: PORT MAP(c, d, x2);
    OR_0: PORT MAP(x1, x2, f);

End structural;
```



Arquitetura Estrutural

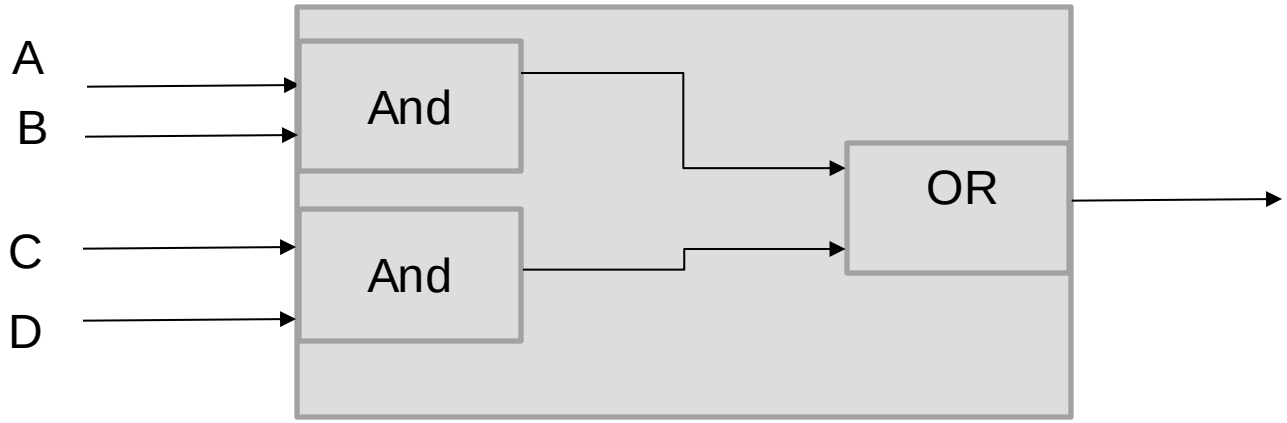
Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Architecture structural of exemplo1 is
SIGNAL x1, x2: bit;
BEGIN
    And_0: PORT MAP(a, b, x1);
    And_1: PORT MAP(c, d, x2);
    OR_0: PORT MAP(x1, x2, f);

End structural;
```

Label



Arquitetura Estrutural

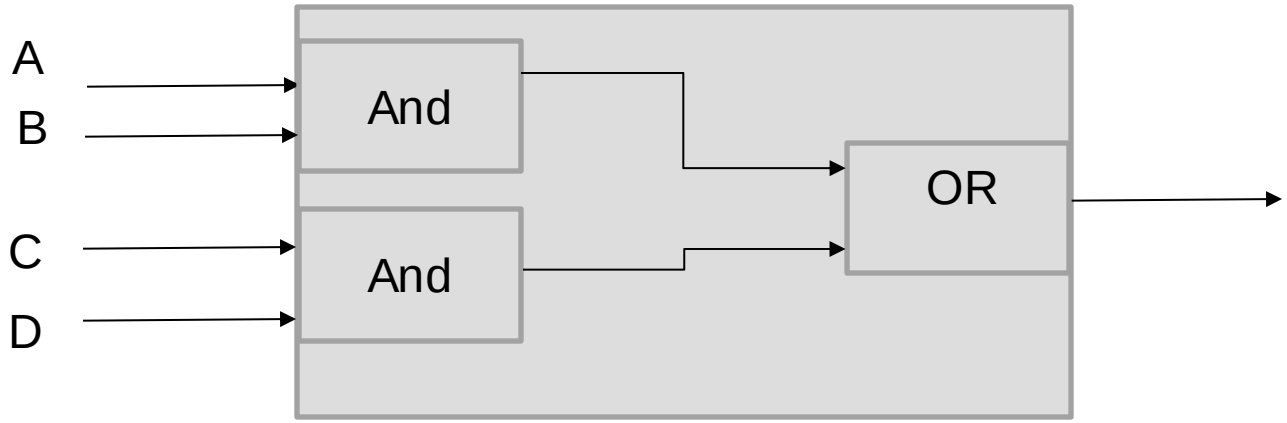
Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Architecture structural of exemplo1 is
SIGNAL x1, x2: bit;
BEGIN
    And_0: PORT MAP(a, b, x1);
    And_1: PORT MAP(c, d, x2);
    OR_0: PORT MAP(x1, x2, f);

End structural;
```

Label identifica o componentes



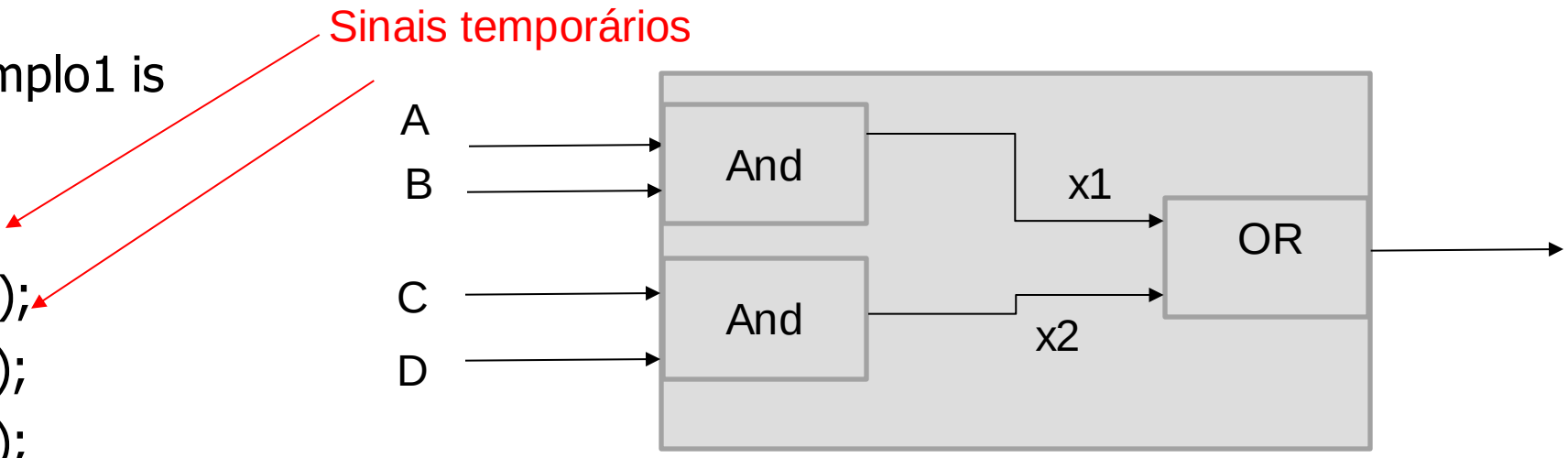
Arquitetura Estrutural

Exemplo: Faça o exemplo f = A.B + C.D conforme a arquitetura estrutural

Primeiro vamos criar os componentes:

```
Architecture structural of exemplo1 is
SIGNAL x1, x2: bit;
BEGIN
    And_0: PORT MAP(a, b, x1);
    And_1: PORT MAP(c, d, x2);
    OR_0: PORT MAP(x1, x2, f);

End structural;
```



Arquitetura Estrutural

Exercício: Faça o exemplo do detector de overflow usando architecture structural

Estruturas Concorrentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Neste tópico, apresenta-se estruturas de condição e de repetição em códigos concorrentes.

COPPE UFRJ

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Exemplos:

```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

Estruturas Concorrentes

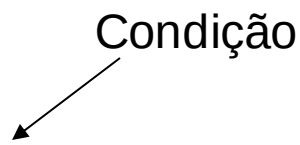
Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;

Exemplos:




```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Exemplos:


```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Exemplos:

```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

1° verifica se sel=0. Caso não, verifica se sel=1

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Exemplos:

Caso não, então a saída é sempre c

```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Exemplos:

```
x<= a when sel=0 else
    b when sel=1 else
    c;
```

Portanto, a saída default c ocorre quando sel!=0 e sel!=1

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Sintaxe (WITH-SELECT-WHEN)

```
WITH identifier SELECT
    assignment when condition else
    assignment when condition else
    ...;
```

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;

Sintaxe (WITH-SELECT-WHEN)

WITH identifier SELECT
assignment when condition else
assignment when condition else
...;

Nesse caso a condition é o valor que variável identifier pdoe assumir

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Sintaxe (WITH-SELECT-WHEN)

```
WITH identifier SELECT
    assignment when condition else
    ...;
```

Exemplo:

```
WITH sel SELECT
   X <= a WHEN 0 else
        b WHEN 1 else
        c WHEN OTHERS;
```

Estruturas Concorrentes

Estruturas de condição em códigos concorrentes.

- The WHEN statement: atua como uma estrutura de condição, tendo duas formas possíveis. Vamos mostrar as sintaxes em exemplos de multiplexadores.

Sintaxe (WHEN-ELSE)

```
Assignment WHEN condition ELSE
Assignment WHEN condition ELSE
...;
```

Sintaxe (WITH-SELECT-WHEN)

```
WITH identifier SELECT
    assignment when condition else
    ...;
```

Exemplo:

```
WITH sel SELECT
   X <= a WHEN 0 else
        b WHEN 1 else
        c WHEN OTHERS;
```

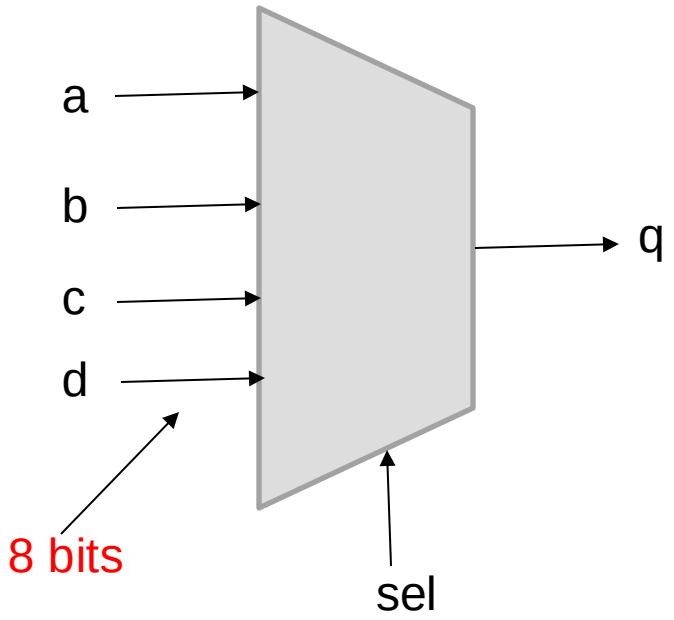
Estruturas Concorrentes

Exemplo: MUX 4:1 COM VETORES DE 8 BITS

Estruturas Concorrentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Exemplo: MUX 4:1 COM VETORES DE 8 BITS

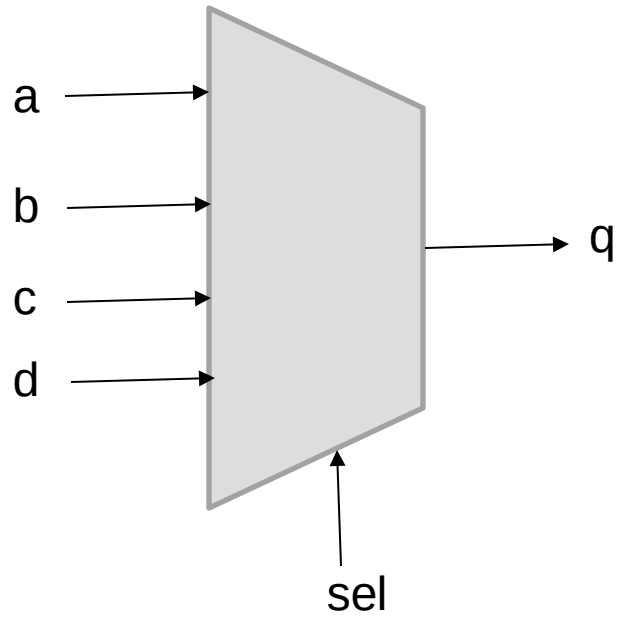


COPPE UFRJ

Estruturas Concorrentes

Exemplo: MUX 4:1 COM VETORES DE 8 BITS

```
library ieee;
use ieee.std_logic_1164.all
Entity mux_4_1 is
   PORT (a, b, c, d:     in std_logic_vector(7 downto 0);
         sel:            in std_logic_vector(1 downto 0);
         q:              out std_logic_vector(7 downto 0));
End mux_4_1;
```



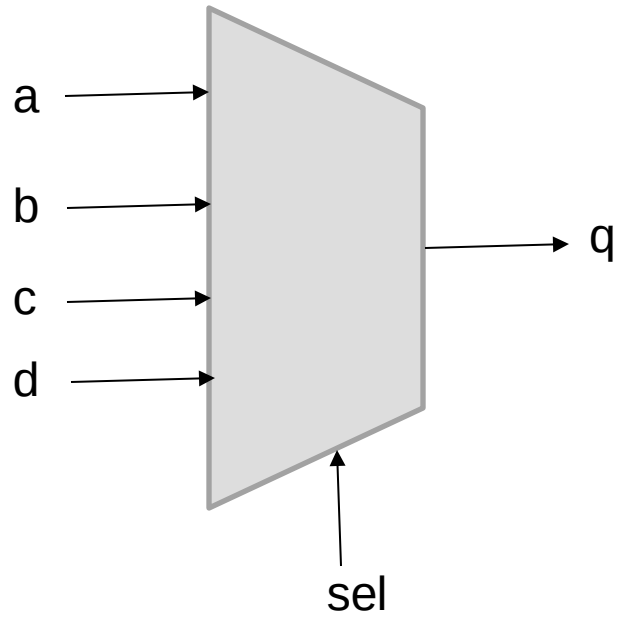
Estruturas Concorrentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Exemplo: MUX 4:1 COM VETORES DE 8 BITS

```
Architecture dataflow of mux_4_1 is
BEGIN
   q <= b when (sel="01") else
        c when(sel="10") else
        d when(sel="11") else
        a;                          --default case

END dataflow;
```



COPPE UFRJ

Estruturas Concorrentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

The GENERATE statement: isso equivale a uma estrutura de repetição, ou seja, a um LOOP.

Vamos ver a sintaxe para implementar um loop em código concorrente.

COPPE UFRJ

Estruturas Concorrentes

The GENERATE statement: isso equivale a uma estrutura de repetição, ou seja, a um LOOP.

- Sintaxe:
  - Label: for identifier in range GENERATE
    Concurrements assignments;
    End GENERATE [label];
    - Label: identifica a estrutura
    - Identifier é a variável que percorre o range
    - Range especifica um certo intervalo

Estruturas Concorrentes

Diferenças de FOR-GENERATE para Estrutura For em linguagem de programação.

Repetição de Código para Components


```
ARCHITECTURE structural of teste is
   COMPONENT func
      PORT(a0, a1:    int std_logic;
                 Y: out std_logic);
    End COMPONENT;
    BEGIN
       G1: for i in (N-1) Downto 0 GENERATE
             func_i: PORT MAP(a0=>sig1(i),  a1=>sig2(i), y=>z(i));
        End GENERATE G1;
    END structural;
```

Repetição de Código para Components

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
ARCHITECTURE structural of teste is
   COMPONENT func
      PORT(a0, a1:    int std_logic;
                 Y: out std_logic);
    End COMPONENT;
    BEGIN
       G1: for i in (N-1) Downto 0 GENERATE
            func_i: PORT MAP(a0=>sig1(i),  a1=>sig2(i), y=>z(i));
        End GENERATE G1;
    END structural;
```

Cria uma repetição de múltiplos componentes

COPPE UFRJ

Estruturas Concorrentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

The GENERATE statement: isso equivale a uma estrutura de repetição, ou seja, a um LOOP.

- Exemplo:

```
G1: for i in 0 to M GENERATE
    b(i) <= a(M-i);
End Generate G1;
```


COPPE UFRJ

Arquitetura Comportamental

Arquitetura utilizada na descrição de sistemas sequenciais. Essa arquitetura inaugura o mundo sequencial em VHDL

Elemento principal: PROCESS

Arquitetura Comportamental

Sintaxe do elemento PROCESS:

```
process_name: PROCESS (sensitivity_list1, ...)
BEGIN
-- comando do process
END PROCESS process_name;
```

process_name: atua como o label identificando o PROCESS

Arquitetura Comportamental

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Sintaxe do elemento PROCESS:

```
process_name: PROCESS (sensitivity_list1, ...)
BEGIN
-- comando do process
END PROCESS process_name;
```

process_name: atua como o label identificando o PROCESS

PROCESS: palavra reservada para indicar início da aprte comportamental

COPPE UFRJ

Arquitetura Comportamental

Sintaxe do elemento PROCESS:

```
process_name: PROCESS (sensitivity_list1, ...)
BEGIN
-- comando do process
END PROCESS process_name;
```

process_name: atua como o label identificando o PROCESS

PROCESS: palavra reservada para indicar início da aprte comportamental

Lista de sensibilidade: lista de sensibilidade que indica quais são as variáveis e sinais cuja alteração deve levar à reavaliação da saída.

Arquitetura Comportamental

Sintaxe do elemento PROCESS:

```
process_name: PROCESS (sensitivity_list1, ...)
BEGIN
-- comando do process
END PROCESS process_name;
```

process_name: atua como o label identificando o PROCESS

PROCESS: palavra reservada para indicar início da aprte comportamental

Lista de sensibilidade: lista de sensibilidade que indica quais são as variáveis e sinais cuja alteração deve levar à reavaliação da saída. Em outras palavras, quais sinais fazem com que os comandos do PROCESS se movimentem, sem alterem.

Arquitetura Comportamental

Sempre uma variável da lista de sensibilidade é alterada, todo o código do PROCESS é executado

Arquitetura Comportamental

Sintaxe do elemento PROCESS:

```
process_name: PROCESS (sensitivity_list1, ...)
BEGIN
-- comando do process
END PROCESS process_name;
```

Os comandos dentro Process são Sequenciais

process_name: atua como o label identificando o PROCESS

PROCESS: palavra reservada para indicar início da aprte comportamental

Lista de sensibilidade: lista de sensibilidade que indica quais são as variáveis e sinais cuja alteração deve levar à reavaliação da saída. Em outras palavras, quais sinais fazem com que os comandos do PROCESS se movimentem, sem alterem.

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade 8 bits

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

A saída do codificador de prioridade mostra o código do binário mais prioritário. Considere prioridade como o seguinte:

Seja $x = x_7...x_0$ seja a entrada do codificador de prioridade.

Se i>j, logo a entrada x_i é mais prioritária que x_j

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

A saída do codificador de prioridade mostra o código do binário mais prioritário. Considere prioridade como o seguinte:

Seja $x = x_7...x_0$ seja a entrada do codificador de prioridade.

Se i>j, logo a entrada x_i é mais prioritária que x_j

Exemplo: Se $x_1=x_5=1$ e todos os demais igual a 0. Logo o codificador de prioridade emitirá a saída sendo o binário de 5

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
use ieee;
use ieee.std_logic_1164.all
ENTITY priority_coder IS
  Port(x0, x1, x2, x3, x4, x5, x6, x7: in std_logic;
       prio_out: out std_logic_vector(7 DOWNTO 0)
       );
End priority_coder;
```

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
       Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
       End process p_code;
End behavorial;
```

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
      Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
      End process p_code;
End behavorial;
```

Há Begin para architecture e outro begin pro process

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
       Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
       End process p_code;
End behavorial;
```

";" para cada fim de if-elseif statement

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

Lista de Sensibilidade

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
       Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
       End process p_code;
End behavorial;
```

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
       Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
           End if;
       End process p_code;
End behavorial;
```

End if

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
       Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
           End if;
       End process p_code;
End behavorial;
```

Comandos Sequencial

Arquitetura Comportamental

Exemplo: Descreva, em VHDL, um codificador de prioridade com 8 bits

```
Architecture behavorial of priority_coder is
Begin
    p_code: process (x0, x1, x2, x3, x4, x5, x6, x7)
      Begin
          if (x7=1) then prio_out="111";
          Elsif(x6=1) then prio_out="110";
           ...
           ...
           else prio_out="000";
          End if;
      End process p_code;
End behavorial;
```

If-elseif-else statement

```
If (condition) then assignments;
Elseif (condition) then assignments;
Else assignment;
End if;
```

Arquitetura Comportamental

Exemplo: Identificador de igualdade de 4 bits usando Process

Arquitetura Comportamental

Exemplo: Identificador de igualdade de 4 bits usando Process

O modelo recebe dois sinais com 4 bits cada, compara-os e emite saída 1, se forem iguais e zero, caso contrário.

Arquitetura Comportamental

Exemplo: Identificador de igualdade de 4 bits usando Process

O modelo recebe dois sinais com 4 bits cada, compara-os e emite saída 1, se forem iguais e zero, caso contrário.

Arquitetura Comportamental

Exemplo: Identificador de igualdade de 4 bits usando Process

```
Entity equal_identifier IS
    Port (
    );
End equal_identifier;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de condição e de repetição em códigos sequenciais.

- The IF statement
- The WAIT statement
- The CASE statement
- The LOOP statement

Só são permitidas dentro da estrutura de PROCESS, pois são comandos SEQUENCIAIS

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de condição e de repetição em códigos sequenciais.

- The IF statement: estrutura de decisão
- Sintaxe

```
IF condition THEN
    assignments;
ELSIF condition THEN
    assignments;
ELSE
   assignments;
End IF;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de condição e de repetição em códigos sequenciais.

- The IF statement: estrutura de decisão
- Sintaxe

```
IF condition THEN
    assignments;
ELSIF condition THEN
    assignments;
ELSE
   assignments;
End IF;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de condição em códigos sequenciais.

- The Case statement: estrutura de decisão
- Sintaxe

```
CASE identifier IS
    WHEN value => assignments;
    WHEN value => assignments;
    ...;
END CASE;
```

Exemplo:

```
Case sel is
   When 0 => a;
   When 1=> b;
   When OTHERS => c;
End Case;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (For - LOOP):

```
[label] FOR identifier in range LOOP
    -- sequential assignments;
END LOOP [label];
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (For - LOOP):

Opcional

[label] FOR identifier in range LOOP

    -- sequential assignments;

END LOOP [label];

Exemplo:

```
FOR i in x'RANGE LOOP
    x(i) <= a(M-i) AND b(i);
End LOOP;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (For - LOOP):

Opcional

```
[label] FOR identifier in range LOOP
    -- sequential assignments;
END LOOP [label];
```

Exemplo:

x'RANGE percorre todos os bits de x

```
FOR i in x'RANGE LOOP
    x(i) <= a(M-i) AND b(i);
End LOOP;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (For - LOOP):

Opcional

[label] FOR identifier in range LOOP

    -- sequential assignments;

END LOOP [label];

Exemplo:

Python:
range(0, len(x))

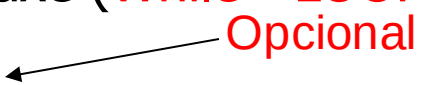
```
FOR i in x'RANGE LOOP
    x(i) <= a(M-i) AND b(i);
End LOOP;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (While - LOOP):




```
[label] WHILE condition LOOP
    -- sequential assignments;
END LOOP [label];
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (While - LOOP):

Opcional

```
[label] WHILE condition LOOP
    -- sequential assignments;
END LOOP [label];
```

Exemplo:

Python:
range(0, len(x))

```
i:=0;
WHILE i<=M LOOP
      x(i) <= a(M-i) AND b(i);
      i := i +1;
End LOOP;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (While - LOOP):

Opcional

```
[label] WHILE condition LOOP
    -- sequential assignments;
END LOOP [label];
```

Exemplo:

Python:
range(0, len(x))

```
i:=0;
WHILE i<=M LOOP
      x(i) <= a(M-i) AND b(i);
      i := i +1;
End LOOP;
```

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (LOOP with EXIT): essa estrutura faz um loop com uma condição de SAÍDA

Estruturas Sequenciais

Neste tópico, apresenta-se estruturas de repetição em códigos sequenciais.

The LOOP statement: isso permite criação de múltiplas assignments por meio de uma estrutura de repetição

Sintaxe (LOOP with EXIT):

```
For identifier in range LOOP
    [label:] EXIT [loop_label] WHEN condition;
    assignments;
End LOOP;
```

Estruturas Sequenciais

Exemplo: Conte o número de zeros presentes em uma sequencia de 8 bits

Estruturas Sequenciais

Exemplo: Conte o número de zeros presentes em uma sequencia de 8 bits

```
Entity counting_zeros_8 IS
      Generic(N: Integer := 8);
      Port(x:         in bit_vector(N-1 downto 0);
            y:        out INTEGER RANGE 0 TO N
      );
End counting_zeros_8;
```

Estruturas Sequenciais

Exemplo: Conte o número de zeros presentes em uma sequencia de 8 bits

```
ARCHITECTURE behavorial of counting_zeros_8 IS
BEGIN
        Process(x)
          VARIABLE temp: INTEGER 0 RANGE N;
          Begin
             temp := 0;
             For i in x'RANGE LOOP
               IF (x(i) = '0') then
                  temp:=temp+1;
               End IF;
             y <= temp;
          End Process;
END behavorial;
```

Estruturas Sequenciais

Exemplo: Conte o número de zeros presentes, em sequência, em uma sequencia de 8 bits

Estruturas Sequenciais

Exemplo: Conte o número de zeros presentes em uma sequencia de 8 bits

```
ARCHITECTURE behavorial of counting_zeros_8 IS
BEGIN
        Process(x)
          VARIABLE temp: INTEGER 0 RANGE N;
          Begin
              temp := 0;
              For i in x'RANGE LOOP
                 EXIT WHEN x(i) = '1';
                  temp := temp + 1;
                 End IF;
               y<= temp;
          End Process;
END behavorial;
```

Signal vs Variable

Signal

- Declarada na Architecture

Variable

- Declarada dentro do PROCESS

Signal vs Variable

Signal

- Declarada na Architecture
- Escopo global

Variable

- Declarada dentro do PROCESS
- Escopo local

Signal vs Variable

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Signal

- Declarada na Architecture
- Escopo global
- Novo valor é considerado após a conclusão do PROCESS

Variable

- Declarada dentro do PROCESS
- Escopo local
- Novo valor já está disponível imediatamente dentro do PROCESS

COPPE UFRJ

Signal vs Variable

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Signal

- Declarada na Architecture
- Escopo global
- Novo valor é considerado após a conclusão do PROCESS
- Atribuição: **<=**

Variable

- Declarada dentro do PROCESS
- Escopo local
- Novo valor já está disponível imediatamente dentro do PROCESS
- Atribuição: **:=**

COPPE UFRJ

Exercício

- Contrua um contador de 4 bits
- Construa um gerador de paridade

Dúvidas

- E-mail de contato: pacheco@gta.ufrj.br