Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

VHDL

Prof. Jacson Luiz Matte
jacson.matte@uffs.edu.br

Introdução

VHDL: Uma linguagem para descrever sistemas digitais

Outras linguagens de descrição de hardware:

VERILOG, AHDL, Handel-C, SDL, ISP, Esterel, … (existem dezenas)

Originalmente para especificar hardware, hoje, simulação e síntese, também!

Origem:

DoD 1980

Linguagem para descrever hardware, no contexto do programa americano

"Very High Speed Integrated Circuits" (VHSIC), iniciado em 1980.

VHDL ➔ **V**HSIC **H**ardware **D**escription **L**anguage

Padrão IEEE em 1986 (Institute of Electrical and Electronics Engineers), revisado em 1993

Linguagem utilizada mundialmente por empresas de CAD (simulação, síntese, propriedade intelectual). Verilog muito usada nos EUA.

Introdução

- ✓ **Benefícios**
  - ✓ Especificação do sistema digital:
    - ✓ Projetos independentes da tecnologia (implementação física é postergada)
    - ✓ Ferramentas de CAD compatíveis entre si
    - ✓ Flexibilidade: re-utilização, escolha de ferramentas e fornecedores
    - ✓ Facilidade de atualização dos projetos
    - ✓ Permite explorar, em um nível mais alto de abstração, diferentes alternativas de implementação
    - ✓ Permite, através de simulação, verificar o comportamento do sistema digital

Introdução

- ✓ **Benefícios**
  - ✓ Nível físico:
    - ✓ Reduz tempo de projeto (favorece níveis abstratos de projeto)
    - ✓ Reduz custo
    - ✓ Elimina erros de baixo nível
    - ✓ **Conseqüência:** reduz “time-to-market”

Introdução

Desvantagens

- ✓ Hardware gerado é menos otimizado
- ✓ Controlabilidade/Observabilidade de projeto reduzidas
- ✓ Falta de pessoal treinado para desenvolver com a linguagem.

Introdução

Aplicações

- ✓ ASICs (Application-Specific Integrated Circuit)
- ✓ Lógica Randômica
- ✓ Integração de Componentes
- ✓ Prototipação

Introdução

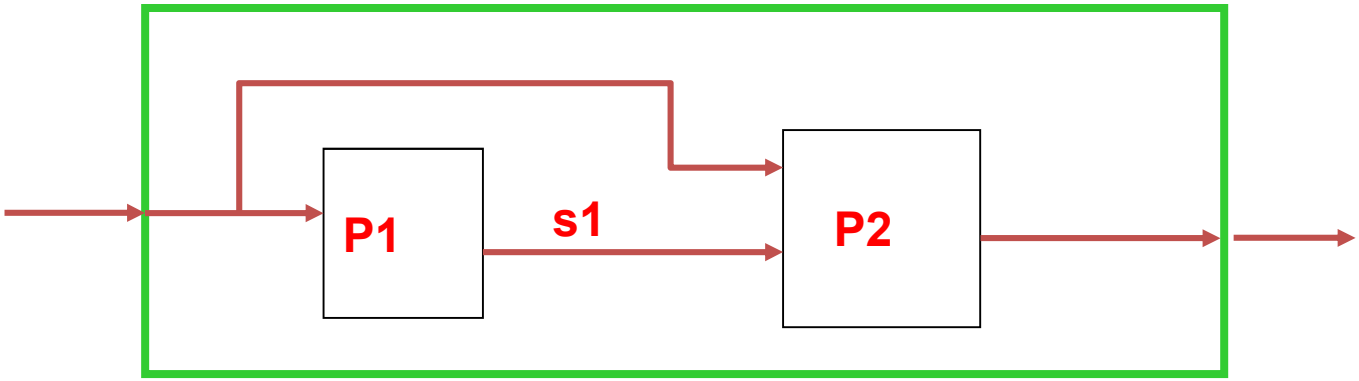
Níveis de Abstração

- ✓ Permite descrever hardware em diversos níveis de abstração
  - Algorítmico, ou Comportamental
  - Transferência entre registradores (RTL)
  - Nível lógico com atrasos unitários
  - Nível lógico com atrasos arbitrários
- ✓ Favorece projeto descendente (“top-down design”)
  - Projeto é inicialmente especificado de forma abstrata, com detalhamento posterior dos módulos
  - Exemplo : **A <= B + C after 5.0 ns;**
    - A forma de realizar a soma pode ser decidida no momento da implementação (e.g. propagação rápida de carry, ou não, paralelo ou série, etc)

Introdução

VHDL é uma Linguagem de Programação ????

- ✓ *Quase …*
- ✓ *Paralelismo entre componentes de um circuito digital*
- ✓ *Comunicação entre processos paralelos*



Processos P1 e P2 rodam em paralelo (podem ser simplesmente tanto duas portas lógicas, como dois módulos arbitrariamente complexos), com algum sinal sincronizando a comunicação entre eles (ex. S1 na figura).

Introdução

VHDL é uma Linguagem de Programação ????

Atraso dos componentes

A <= B + C after 5.0 ns;

D <= A + E;

Temporização

```
x <= y;
y <= z;
wait on clock;
```

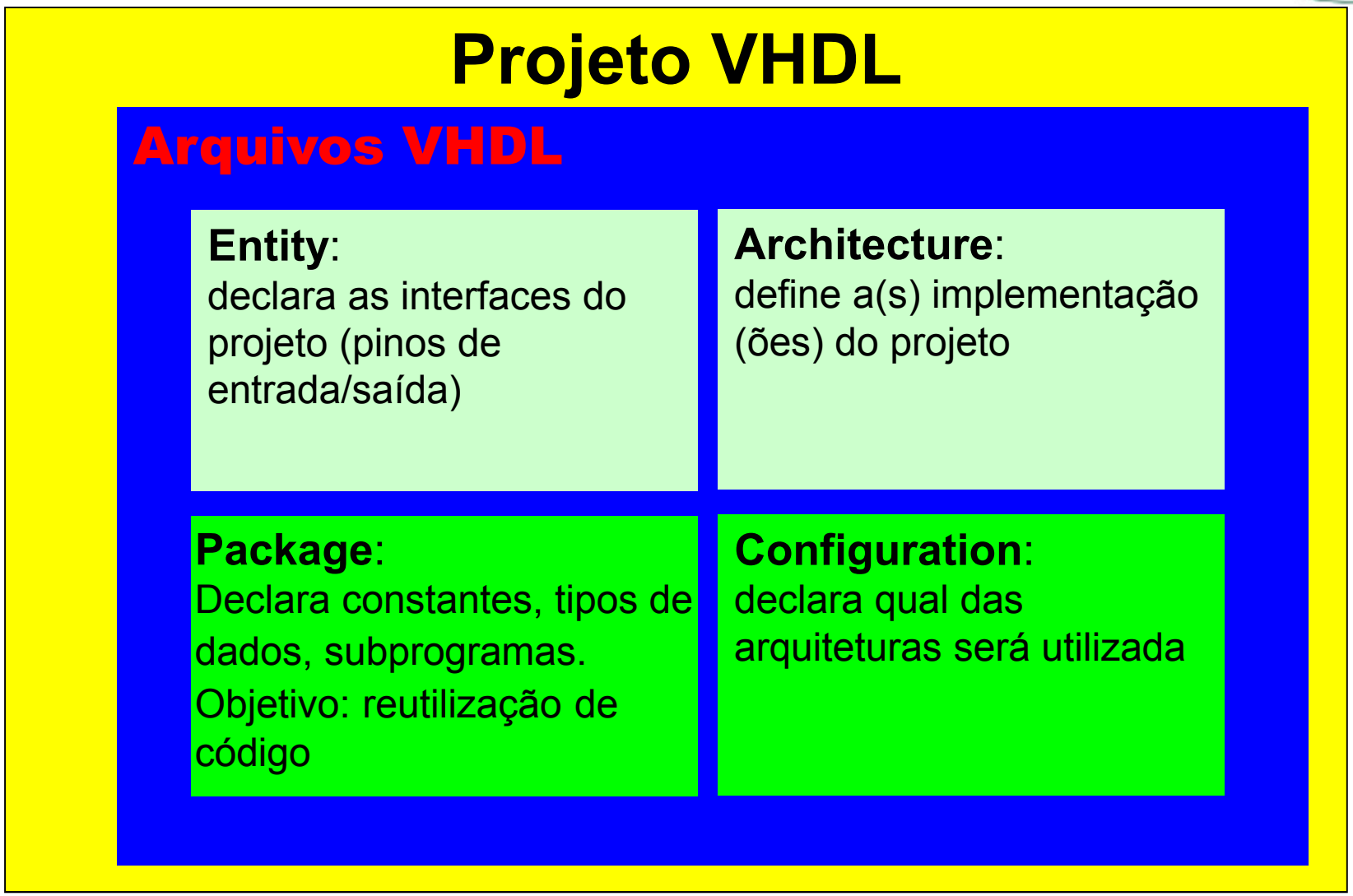
Variáveis: sem temporização – linguagem de programação

Sinais: temporizados

Código é executado em um simulador (ao invés de um compilador), não há um código executável

Controle de versões dos módulos através de "configurações"

Representação de um sistema em VHDL



Representação de um sistema em VHDL

- ✓ Cada módulo tem sua própria "*entity*" e "*architecture*".
- ✓ As arquiteturas podem ser descritas tanto a nível comportamental quanto estrutural ou uma mistura.
  - Comportamental: Algorítmica ou fluxo de dados;
  - Estrutural: indica componentes e conexões;
- ✓ Toda a comunicação ocorre através das portas declaradas em cada *entity*, observando-se o tipo, tamanho, se é sinal ou barramento e a direção.
- ✓ Várias funções e tipos básicos são armazenados em bibliotecas (*library*). A biblioteca "IEEE" sempre é incluída.
- ✓ Biblioteca do usuário (default): work. Todos os arquivos contidos no diretório de trabalho fazem parte da biblioteca do usuário.

Representação de um sistema em VHDL

- ✓ VHDL não é case sensitive

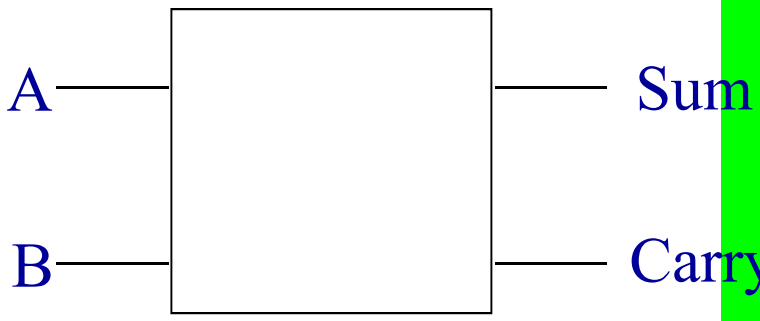
  Bit ↔ bit ↔ BIT
- ✓ Comentários: dois hifens adjascentes (--)

  -- esta linha é um comentário

Representação de um sistema em VHDL

Entity

- ✓ *Especifica somente a interface*
- ✓ *Não contém definição do comportamento*
- ✓ *Direção: in, out, inout, buffer*




```
entity halfadder is
        port (
                A: in STD_LOGIC;
                B: in STD_LOGIC;
                sum: out STD_LOGIC;
                carry: out STD_LOGIC
        );
end halfadder;
```

Representação de um sistema em VHDL

Architecture

- ✓ *Especifica o comportamento da entity*
- ✓ *Deve ser associada a uma entity específica*
- ✓ *Uma entity pode ter várias architectures*

```
architecture comp of halfadd is
    begin
        sum <= A xor B;
        carry <= A and B;
    end comp;
```

Representação de um sistema em VHDL

Entity:

```
entity nome is
  [generic (lista de parâmetros);]
  [port (lista de parâmetros);]
  [ declarações ]
  [begin  sentenças ]
end [ entity ] [ nome ];
```

Architecture:

```
architecture nome of
    nome_entidade is
   [ declarações ]
   begin
    [ sentenças concorrentes ]
end [ architecture ] [ nome ];
```

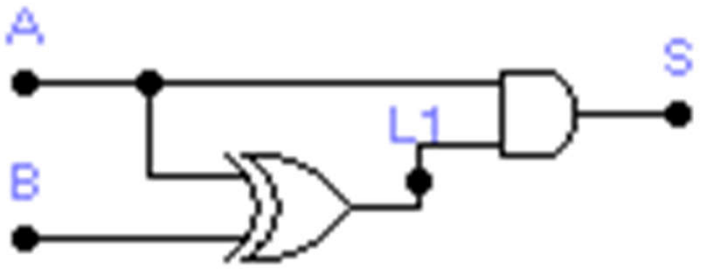
Representação de um sistema em VHDL

Descrições:

a) algorítmica

b) fluxo de dados

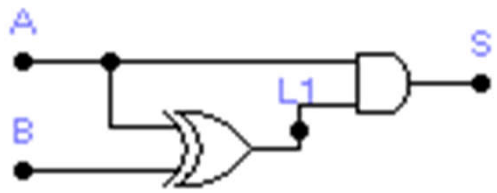
c) estrutural



Representação de um sistema em VHDL

a) algorítmica

```
entity comportamento is
   port (
        A: in STD_LOGIC;
        B: in STD_LOGIC;
        S: out STD_LOGIC
   );
end comportamento;
```




```
architecture comport_algor of
                    comportamento is
begin
   process(A,B)
   begin
            if(B < A) then
                    s <= '1';
            else
                    s <= '0';
            end if;
        end process;
end comport_algor;
```

Representação de um sistema em VHDL

Primitiva de base (concorrência): process

Observar diferença entre variável e sinal:

Variável: interna ao processo, do tipo natural, atribuição IMEDIATA

Sinal: global, com atribuição ao término do processo

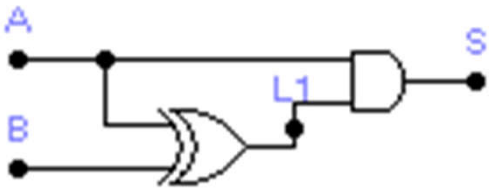
✓Notar que na declaração do processo há uma lista de ativação

Significado: o processo está em espera até um sinal da lista de ativação mudar.

Representação de um sistema em VHDL

b) fluxo de dados

```
entity comportamento is
   port (
        A: in STD_LOGIC;
        B: in STD_LOGIC;
        S: out STD_LOGIC
   );
end comportamento;
```

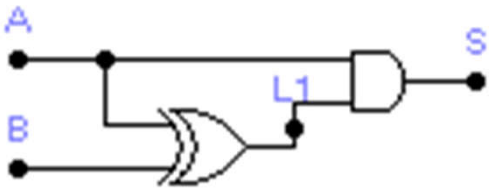



```
architecture comport_fluxo of
                        comportamento is
begin
   s <= ‘1’ when B < A
              else '0';
end comport_fluxo ;
```

Representação de um sistema em VHDL

c) estrutural

```
entity comportamento is
   port (
        A: in STD_LOGIC;
        B: in STD_LOGIC;
        S: out STD_LOGIC
   );
end comportamento;
```




```
architecture comp_estrutural of
                    comportamento is
   signal L1: STD_LOGIC;
   component  XOR2 is
    port (A1,B1: in std_logic; X1: out
std_logic);
    end component;
   component  And2 is
    port (A2,B2: in std_logic; X2: out
std_logic);
    end component
begin
   U1: entity xor2 port map (A,B,L1);
   U2: entity and2 port map (A,L1,S);
end comp_estrutural ;
```

(existem outros 2 arquivos com os pares entidade arquitetura para AND2 e XOR2)

Sintaxe VHDL

VHDL é uma linguagem fortemente tipada

(integer 1 ≠ real 1.0 ≠ bit '1')

- ✓ auxilia a detectar erros no início do projeto
  exemplo: conectar um barramento de 4 bits a um barramento de 8 bits

Tópicos

Escalares

Objetos

Expressões

Sintaxe VHDL

Escalar é o oposto ao array

character / bit / boolean / real / integer / physical_unit

std_logic (IEEE)

✓ *Bit*

Assume valores '0' e '1' (usar aspas simples)

Declaração explícita: bit'('1'), pois neste caso '1' também pode ser 'character'.

bit não tem relação com o tipo boolean.

bit_vector: tipo que designa um conjunto de bits. Exemplo: "001100" ou x"00FF". Não é escalar.

Sintaxe VHDL

- ✓ *Boolean*
  - Assume valores *true* e *false*.
  - Útil apenas para descrições abstratas, onde um sinal só pode assumir dois valores
- ✓ *Real*
  - Utilizado durante desenvolvimento da especificação
  - Sempre com o ponto decimal
  - Exemplos: -1.0 / +2.35 / 37.0 / -1.5E+23
- ✓ *Inteiros*
  - Exemplos: +1 / 1232 / -1234
  - NÃO é possível realizar operações lógicas sobre inteiros (deve-se realizar a conversão explícita)
  - Vendedores provêem versões próprias: signed, **bit_vector** (este tipo permite operações lógicas e aritméticas)

Sintaxe VHDL

✓ *Character*

VHDL não é "case sensitive", exceto para caracteres.

valor entre aspas simples: 'a', 'x', '0', '1', …

declaração explícita: character'('1'), pois neste caso '1' também pode ser 'bit'.

string: tipo que designa um conjunto de caracteres. Exemplo: "xuxu".

✓ *Physical*

Representam uma medida: voltagem, capacitância, tempo

Tipos pré-definidos: fs, ps, ns, um, ms, sec, min, hr

Sintaxe VHDL

✓ *Intervalos (range)*

sintaxe: *range valor_baixo* ***to*** *valor_alto*
range valor_alto ***downto*** *valor_baixo*

integer range 1 to 10 **NÃO** integer range 10 to 1

real range 1.0 to 10.0 **NÃO** integer range 10.0 to 1.0

declaração sem **range** declara todo o intervalo

declaração **range<>** : declaração postergada do intervalo

Sintaxe VHDL

✓ *Enumerações*

Conjunto ordenando de nomes ou caracteres.

Exemplos:

```
type logic_level is (´0´, ´1´, ´X´, ´Z´);
type octal is (´0´, ´1´, ´2´, ´3´, ´4´, ´5´, ´6´,
  ´7´);
```

Sintaxe VHDL

exemplos de arrays pré definidos:

type string is array (positive range <>) of character;

type bit_vector is array (natural range <>) of bit;

preenchimento de um array: posicional ou por nome

type a is array (1 to 4) of character;

posicional: ('f', 'o', 'o', 'd')

por nome: (1 => 'f', 3 => 'o', 4 => 'd', 2 => 'o')

valores default: ('f', 4 => 'd', others => 'o')

Sintaxe VHDL

```
signal z_bus : bit_vector (3 downto 0);
signal c_bus : bit_vector (0 to 3);

z_bus <= c_bus;
```

z_bus(3) ⟵ c_bus(0)
z_bus(2) ⟵ c_bus(1)
z_bus(1) ⟵ c_bus(2)
z_bus(0) ⟵ c_bus(3)

```
z_bus(3) <= c_bus(2);
```

Obs.:

- tamanho dos arrays deve ser o mesmo
- elementos são atribuídos por posição, pelo número do elemento

Sintaxe VHDL

Objetos podem ser escalares ou vetores (arrays)

Devem obrigatoriamente iniciar por uma letra, depois podem ser seguidos de letras e dígitos (o caracter "_" pode ser utilizado). Não são case sensitive, ou seja XuXu é o mesmo objeto que XUXU ou xuxu.

Constantes / Variáveis / Sinais

Sintaxe VHDL

- ✓ ***Constante: nome dado a um valor fixo***

sintaxe: *constant identificador : tipo [:=expressão];*

correto: *constant gnd: real := 0.0;*

incorreto *gnd := 4.5; -- atribuição a constante fora da declaração*

constantes podem ser declaradas em qualquer parte, porém é aconselhável declarar as freqüentemente utilizadas em um package

Sintaxe VHDL

✓ Variáveis

utilizadas em processos, sem temporização, atribuição a elas é imediata.

sintaxe:

variable identificador (es) : tipo [restrição] [:=expressão];

exemplo:

variable indice : **integer range** 1 **to** 50 := 50;

variable ciclo_maquina : **time range** 10 ns to 50 ns := 10ns;

variable memoria : **bit_vector** (0 to 7)

variable x, y : **integer**;

Sintaxe VHDL

- ✓ *Sinais*

  Comunicação entre módulos.

  Temporizados.

  Podem ser declarados em entity, architecture ou em package.

  Não podem ser declarados em processos, podendo serem utilizados no interior destes.

sintaxe:

signal identificador (es) : tipo [restrição] [:=expressão];

exemplo

- **signal** cont : **integer range** 50 **downto** 1;
- **signal** ground : **bit** := ´0´;
- **signal** bus : **bit_vector**;

Sintaxe VHDL

✓ *Operadores e Expressões*

são fórmulas que realizam operações sobre objetos de mesmo tipo.

Operações lógicas: and, or, nand, nor, xor, not
Operações relacionais: =, /=, <, <=, >, >=
Operações aritméticas: +, -
Operações aritméticas: *, /
Operações aritméticas: mod, rem, **
Concatenação: &

Menor

PRIORIDADE

Maior

Sintaxe VHDL

✓ *Operadores*

Operadores Lógicos – trabalham com tipos BIT, BOOLEAN, STD_LOGIC, vetores de tamanho igual e NÃO EM INTEIROS.

Operadores Relacionais – trabalham com qualquer tipo de escalar ou tipo array UNI-DIMENSIONAL cujo tipo de elemento é um tipo discreto (enumeração ou inteiro).

Operadores Aritméticos – trabalham com inteiro, real e STD_LOGIC_VECTOR.

Sintaxe VHDL

Observações:

- ✓ Operações lógicas são realizadas sobre tipos bit e boolean.
- ✓ Operadores aritméticos trabalham sobre inteiros e reais. Por exemplo, pode-se somar vetores de bits.
- ✓ Concatenação é aplicável sobre caracteres, strings, bits, vetores de bits e arrays.
  Exemplos: “ABC” & “xyz” resulta em: “ABCxyz”

  “1001” & “0011” resulta em: “10010011”

Sintaxe VHDL

Qual/quais das linhas abaixo é/são incorreta/s? Justifique a resposta.

variable A, B, C, D : bit_vector (3 downto 0);

variable E,F,G : bit_vector (1 downto 0);

variable H,I,J,K : bit;

[ ] *A := B xor C and D ;*

[ ] *H := I and J or K;*

[ ] *A := B and E;*

[ ] *H := I or F;*

Sintaxe VHDL

VHDL provê facilidades de paralelismo entre diferentes processos e atribuição de sinais.

Dentro dos processos pode-se especificar um conjunto de ações seqüenciais, executadas passo a passo. É um estilo de descrição semelhante a outras linguagens de programação.

Comandos exclusivos de processos: atribuição de variáveis, if, case, for, while, wait (não se pode usá-los fora de processos!)

Variáveis não passam valores fora do processo na qual foram declaradas, são locais. Elas sequer existem fora de um processo.

As atribuições são seqüenciais, ou seja, a ordem delas importa.

Sintaxe VHDL

Comando if (só em processos)

```
if condição_1 then
        <comandos>
elsif condição_2 then
        <comandos>
else
        <comandos>
end if;
```

exemplo 1:

```
if ( A = ‘0’ ) then
    B <= “00”;
else
    B <= “11”;
end if;
```

IMPORTANTE:

teste de borda de subida: if clock'event and clock='1' then …

teste de borda de descida: if clock'event and clock='0' then …

a seqüência na qual estão definidos os ´ifs´ implica na prioridade das ações.

Sintaxe VHDL

Comando if (só em processos)

Qual a implementação em hardware da seguinte seqüência de comandos ?

```
process(A, B, control)
  begin
     if( control = '1' ) then
                    Z <= B;
     else
                    Z <= A;
     end if;
end process;
```

Sintaxe VHDL

Comando for (só em processos)

- ✓ *para descrever comportamento / estruturas regulares*
- ✓ *o "for" declara um objeto, o qual é alterado somente durante o laço*
- ✓ *internamente o objeto é tratado como uma constante e não deve ser alterado.*

```
for item in 1 to last_item loop
    table(item) := 0;
end loop;
```

Comando for (só em processos)

*exit**: termina o laço***

```
for i in 1 to max_str_len loop
        a(i) := buf(i);
        exit when buf(i) = NUL;
 end loop;
```

Sintaxe VHDL

Comando for (só em processos)

Qual a função do laço abaixo ?

```
function conv (byte : word8) return integer is
        variable result : integer := 0;
        variable k : integer := 1;
  begin
        for index in 0 to 7 loop
                if ( std_logic'(byte(index)) = '1')
                        then result := result + k;
                end if;
                k := k * 2;
        end loop;
        return result;
end conv ;
```

Sintaxe VHDL

Comando WHILE (só em processos)

```
while index < length and str(index) /= ' '
 loop
        index := index + 1;
end loop;
```

Sintaxe VHDL

Comando CASE (só em processos)

```
porta_programável :
process (Mode, A, B)
   begin
   case Mode is
        when “000”       => saída <= A and B;
        when “001”       => saída <= A or B;
        when “010”       => saída <= A nand B;
        when “011”       => saída <= A nor B;
        when “100”       => saída <=  not A;
        when “101”       => saída <=  not B;
        when  others     => saída <=   ‘0’
   end case;
end process porta_programavel;
```

Sintaxe VHDL

Comando null

serve, por exemplo, para indicar "faça nada" em uma condição de case.

```
case controller_command is
        when forward => marcha <= '1';
        when reverse => marcha <= '0';
        when idle => null;
end case;
```

Sintaxe VHDL

operador	significado	exemplo
<=	Atribuição de sinal	Aux <= ‘0’
:=	Atribuição de variável	A := ‘1’
:=	Inicialização de constantes, sinais e variáveis	Signal aux : bit := ‘0’
=>	Atribuição de valores únicos em vetores	Vetor <= (0=> ‘0’)
=>	Atribuição de vários valores em vetores junto com a cláusula others	Vetor <= (0 => ‘0’ , others => ‘1’)

Sintaxe VHDL

1. Implemente em VHDL um conversor de binário para Gray de 4 bits.
2. Implemente em VHDL um conversor de Gray para binário de 4 bits.
3. Dados os esquemáticos, obtenha descrições VHDL compatíveis