



Circuitos Digitais II - 6882

André Barbosa Verona
Nardênio Almeida Martins

Universidade Estadual de Maringá
Departamento de Informática

Bacharelado em Ciência da Computação

Aula de Hoje

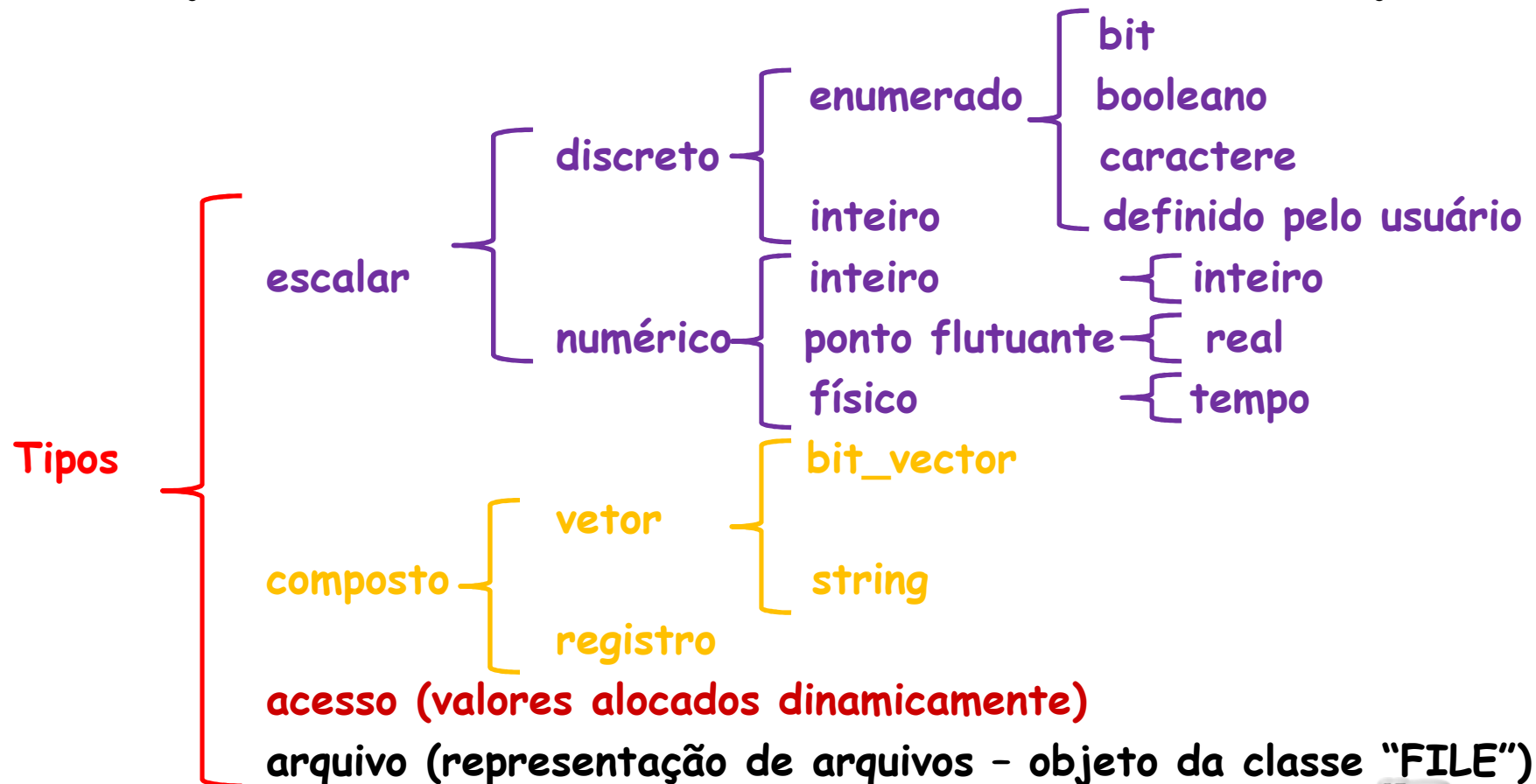
- **Revisão da aula anterior**
- **Classes de Objetos**
- **Operadores**

Revisão

VHDL - Tipos de Dados

Tipos em VHDL

- Representação do conjunto de valores que um objeto pode armazenar e do conjunto de operações que podem ser realizadas com esse objeto.



VHDL - Tipos de Dados

Tipos e subtipos* escalares predefinidos no pacote padrão

TIPO	VALOR	EXEMPLO
BIT	um, zero	'1', '0'
BOOLEAN	falso, verdadeiro	true, false
CHARACTER	caracteres	'a',..., 'z', 'A',..., 'Z', '0',..., '9', '?', '('
INTEGER	$-2^{31}-1 \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#, 2#11_11_011#
NATURAL*	$0 \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#, 2#11_11_011#
POSITIVE*	$1 \leq x \leq 2^{31}-1$	123, 8#173#, 16#7B#, 2#11_11_011#
REAL	$-1.0E+38 \leq x \leq 1.0E+38$	1.23, 1.23E+2, 16#7B#E+1
TIME	fs, ps=10 ³ fs, ns=10 ³ ps, us=10 ³ ns, ms=10 ³ us, sec=10 ³ ms, min=60sec, hr=60min	1us, 100ps, 1fs

- Nota: Objetos declarados com tipos diferentes não permitem a transferência de valores entre eles, a não ser que se faça uma operação de conversão.

VHDL - Tipos de Dados

Implementação dos tipos em VHDL

```
TYPE bit IS ('0', '1'); -- declaração do tipo
```

```
TYPE boolean IS (true, false); -- declaração do tipo
```

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);
```

```
TYPE Impares IS (1, 3, 5, 7, 9);
```

```
TYPE Vogais_Minusculas IS ('a', 'e', 'i', 'o', 'u');
```

```
TYPE BitVector4 IS ARRAY (3 DOWNT0 0) OF BIT;
```

```
TYPE Matriz IS ARRAY (1 TO 10, 1 TO 10) OF INTEGER;
```

```
SUBTYPE Maiusculas IS CHARACTER RANGE 'A' TO 'Z';
```

```
SUBTYPE NatMenor20 IS INTEGER RANGE 0 TO 19;
```

```
SUBTYPE Byte IS STD_LOGIC_VECTOR (7 DOWNT0 0);
```

VHDL - Tipos de Dados

Implementação dos tipos em VHDL

```
TYPE integer IS RANGE -2147483648 TO 2147483648;  
SUBTYPE natural IS integer RANGE 0 TO 2147483648;  
SUBTYPE positive IS integer RANGE 1 TO 2147483648;
```

```
TYPE real IS RANGE -1.0E+38 TO 1.0E+38;
```

```
TYPE resistencia IS RANGE 0 TO 1E9
```

```
    UNITS
```

```
        ohm;
```

```
        Kohm = 1000 ohm;
```

```
        Mohm = 1000 Kohm;
```

```
        Gohm = 1000 Mohm;
```

```
    END UNITS;
```

VHDL - Tipos de Dados

Implementação dos tipos em VHDL

```
TYPE time IS RANGE IMPLEMENTATION_DEFINED
```

```
    UNITS
```

```
        fs;
```

```
        ps = 1000 fs;
```

```
        ns = 1000 ps;
```

```
        us = 1000 ns;
```

```
        ms = 1000 us;
```

```
        sec = 1000 ms;
```

```
        min = 60 sec;
```

```
        hr = 60 min;
```

```
    END UNITS;
```


VHDL - Tipos de Dados

Implementação dos tipos em VHDL

```
TYPE BIT_VECTOR IS ARRAY (INTEGER RANGE <>) OF BIT; -- declaração do tipo
```

```
TYPE STRING IS ARRAY (CHARACTER RANGE <>) OF CHARACTER; -- declaração do tipo
```

```
TYPE std_ulogic IS ('1', '0', 'X', 'Z', 'W', 'L', 'H', 'U', '_'); -- declaração do tipo
```

VHDL - Tipos de Dados

STD_LOGIC (Standard Logic)

- Exemplo 01 do uso do tipo STD_LOGIC (mais de uma fonte):

```
ARCHITECTURE Logica OF Example IS
SIGNAL s1 : STD_LOGIC;
BEGIN
s1 <= '0';
:
:
s1 <= '1';
END Logica;
```

VHDL - Tipos de Dados

STD_LOGIC (Standard Logic)

- Exemplo 01 do uso do tipo STD_LOGIC: (continuação)

```
TYPE stdlogic_table IS ARRAY(std_ulogic, std_ulogic) OF
std_ulogic;
CONSTANT resolution_table : stdlogic_table := (
  --|U    X    0    1    Z    W    L    H    -    |    |
  ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
  ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
  ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X' ), -- | 0 |
  ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X' ), -- | 1 |
  ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X' ), -- | Z |
  ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X' ), -- | W |
  ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X' ), -- | L |
  ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X' ), -- | H |
  ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ) -- | - |
);
```

VHDL - Tipos de Dados

STD_LOGIC (Standard Logic)

- Exemplo 01 do uso do tipo STD_LOGIC: (continuação)
 - Um estado lógico é determinado por duas condições: nível lógico e força
 - Cada valor possui um grau de força e um nível lógico.
 - 'U' é o valor de grau de força mais elevado
 - '-' é o valor de grau de força mais fraco
 - '0' e '1' possuem o mesmo grau de força
 - 'L' e 'H' possuem o mesmo grau de força → 'L' - nível lógico 0 e 'H' - nível lógico 1
 - Conflito de 'U' com qualquer outro valor sempre resultará no valor 'U'
 - Conflito de valores iguais, o resultado será o próprio valor
 - Conflito entre os valores 'X', '0' ou '1', o resultado será o valor 'X'
 - Conflito entre os valores 'W', 'L' ou 'H', o resultado será o valor 'W'

Aula de Hoje

- **Classes de Objetos**

VHDL - Classes de Objetos

Objetos

- Elementos que contêm um valor armazenado.
- Quatro classes:
 - Constante (**CONSTANT**) → objeto com um valor estático.
 - Variável (**VARIABLE**) → o valor imposto pode ser alterado no decorrer do código → empregada em regiões de código sequencial.
 - Sinal (**SIGNAL**) → objeto que pode ter o seu valor alterado → empregado em regiões de código concorrente e sequencial.
 - Arquivo (**FILE**) → objeto associado à criação de arquivos → na versão VHDL-1987 corresponde a um membro da classe Variável.

VHDL - Classes de Objetos

Constantes

- Constantes → valores fixos → imutáveis durante a execução do programa → utilizadas somente para leitura nunca para escrita.
- Declaração → pacotes → uso posterior em entidades de projetos, arquitetura da entidade de uma entidade de projeto, procedimento, bloco, processo ou função.
 - Pacotes → não há necessidade e obrigatoriedade de associar imediatamente a um valor inicial → valor especificado no corpo do pacote.
 - Entidades de projetos, arquitetura da entidade de uma entidade de projeto, procedimento, bloco, processo ou função → local → não sendo utilizada em outras partes do código.
- Especificações → constantes e seus valores → respectivas regiões de declarações.

VHDL - Classes de Objetos

Declaração de objetos da classe Constante

-- classe	lista de nomes	tipo	valor inicial
<i>CONSTANT</i>	nome_da_constante_a	: tipo_x;	-- constante sem valor inicial
<i>CONSTANT</i>	nome_da_constante_a	: tipo_x := valor_inicial;	-- constante com valor inicial
<i>CONSTANT</i>	fixo_1, fixo_2	: tipo_x := valor_inicial;	-- constante com valor inicial

-- exemplo

CONSTANT Pi : REAL := 3.14;

CONSTANT atraso : TIME := 50 ns;

- Nota: Geralmente as constantes são declaradas na arquitetura da entidade, entre as palavras reservadas **IS** e **BEGIN**, sendo visíveis somente nesta arquitetura da entidade.

VHDL - Classes de Objetos

Variáveis

- Variáveis → alterações durante a execução do programa → utilizadas tanto para leitura quanto para escrita.
- Declaração → não há necessidade de um valor inicial → opcional.
- Sem especificação → atribuição de valor inicial → dependência do tipo de dado → associação sempre com o menor valor daquele tipo.

➤ Exemplos:

- ✓ Tipo **INTEGER** → valor inicial igual a -2.147.483.648
- ✓ Tipo **BIT** → valor inicial igual a 0
- ✓ Tipo **STD_LOGIC** → valor inicial igual a -

VHDL - Classes de Objetos

Variáveis

- Variáveis → locais → uso na descrição comportamental de uma entidade de projeto → uso somente dentro de processos, procedimentos e funções, estruturas com execução sequencial.
- Execução de uma operação de atribuição → valor atribuído é associado imediatamente à variável → permanência até o final da execução do processo, procedimento ou função → não manutenção de seu valor para posterior execução.

➤ Exemplo:

✓ `var := var + 1;`

- Próxima execução de um processo, procedimento ou função → retorno do valor inicial da variável.

VHDL - Classes de Objetos

Declaração de objetos da classe Variável

-- classe	lista de nomes	tipo	valor inicial
<i>VARIABLE</i>	nome_da_variável_c	: tipo_z;	-- variável sem valor inicial
<i>VARIABLE</i>	nome_da_variável_d	: tipo_z := valor_inicial;	-- variável com valor inicial
<i>VARIABLE</i>	var_1, var_2	: tipo_z := valor_inicial;	-- variáveis: mesmo tipo e valor

-- exemplo

VARIABLE var : INTEGER;

VARIABLE aux : BIT := '0';

- Nota: As variáveis não podem ser declaradas na região de declarações de uma arquitetura da entidade e, sim, somente dentro de processos, procedimentos, funções ou estruturas com execução sequencial.

VHDL - Classes de Objetos

Sinais

- Declaração na especificação da arquitetura da entidade ou em pacotes → global → utilizado sempre para leitura e para escrita.
- Declaração nunca dentro de processos, procedimentos ou funções.
- Declaração da entidade com a palavra reservada *PORT* → parte da classe de objetos *SIGNAL* → especificação das entradas e saídas → modo de operação.
 - *IN* → entrada
 - *OUT* → saída
 - *BUFFER* → saída com realimentação interna
 - *INOUT* → entrada e saída
- Atenção ao tipo de dado do sinal → existem tipos que não permitem que seus sinais possuam mais de uma fonte para a escrita.

VHDL - Classes de Objetos

Declaração de objetos da classe Sinal

-- classe	lista de nomes	tipo	valor inicial
<i>SIGNAL</i>	nome_do_sinal_a	: tipo_y;	-- sinal sem valor inicial
<i>SIGNAL</i>	nome_do_sinal_b	: tipo_y := valor_inicial;	-- sinal com valor inicial
<i>SIGNAL</i>	nome_x, nome_y, nome_z	: tipo_y := valor_inicial;	-- sinais do mesmo tipo

-- exemplo

SIGNAL tempo : **TIME** := 50 ns;

SIGNAL clk : **BIT** := '0';

- Nota: Os sinais são declaradas na arquitetura da entidade ou em pacotes e, nunca dentro de processos, procedimentos, funções ou estruturas com execução sequencial.

VHDL - Classes de Objetos

Sinais

- Podem possuir um valor inicial associado → utilização do operador ":="
- Podem possuir associação de valores no escopo da arquitetura da entidade → utilização do operador "<=".

```
1      ENTITY signal_ent IS
2          PORT (a, b      : IN BIT;
3                  s      : OUT BIT);
4      END signal_ent;
5
6      ARCHITECTURE signal_arc OF signal_ent IS
7          SIGNAL      tempo    : TIME := 50 ns;
8          SIGNAL      clk      : BIT  := '0';
9      BEGIN
10         s <= a XOR b;
11         clk <= NOT clk AFTER tempo;
12     END signal_arc;
```

VHDL - Classes de Objetos

Sinais

- Variáveis → valores atribuídos imediatamente → esses valores não persistem ao final dos processos.
- Sinais → atualização dos valores ocorrerá somente na suspensão de um processo → persistindo esse valor durante o reinício da execução do processo.

```
1      processo1 : PROCESS(d0, d1, d2)
2          VARIABLE var1 : STD_LOGIC;
3          BEGIN
4              var1 := d0 AND d1;
5              s1 <= var1 OR d2;
6          END processo1;
7      processo2 : PROCESS(d0, d1, d2)
8          BEGIN
9              signal1 <= d0 AND d1;
10             s2 <= signal1 OR d2;
11          END processo2;
```

VHDL - Classes de Objetos

Resumo

Declaração de objetos da classe Constante, Variável e Sinal

-- classe	lista de nomes	tipo	valor inicial
<i>CONSTANT</i>	nome_da_constante_a	: tipo_x;	-- constante sem valor inicial
<i>CONSTANT</i>	nome_da_constante_a	: tipo_x := valor_inicial;	-- constante com valor inicial
<i>CONSTANT</i>	fixo_1, fixo_2	: tipo_x := valor_inicial;	-- constante com valor inicial
<i>VARIABLE</i>	nome_da_variável_c	: tipo_z;	-- variável sem valor inicial
<i>VARIABLE</i>	nome_da_variável_d	: tipo_z := valor_inicial;	-- variável com valor inicial
<i>VARIABLE</i>	var_1, var_2	: tipo_z := valor_inicial;	-- variáveis: mesmo tipo e valor
<i>SIGNAL</i>	nome_do_sinal_a	: tipo_y;	-- sinal sem valor inicial
<i>SIGNAL</i>	nome_do_sinal_b	: tipo_y := valor_inicial;	-- sinal com valor inicial
<i>SIGNAL</i>	nome_x, nome_y, nome_z	: tipo_y := valor_inicial;	-- sinais do mesmo tipo

VHDL - Classes de Objetos

Resumo

Exemplos de transferência de informação entre objetos das classes "*SIGNAL*", "*VARIABLE*" e "*CONSTANT*"

```
sinal_2 <= sinal_1;      -- atribuição de valor para sinal
sinal_3 <= variável_1;   -- atribuição de valor para sinal
sinal_4 <= constante_1; -- atribuição de valor para sinal

variável_2 := sinal_1;   -- atribuição de valor para variável
variável_3 := variável_1; -- atribuição de valor para variável
variável_4 := constante_1; -- atribuição de valor para variável
```

Nota 01: Atribuição de valor para um sinal → delimitador "<="

Nota 02: Atribuição de valor para uma variável → delimitador "!="

VHDL - Classes de Objetos

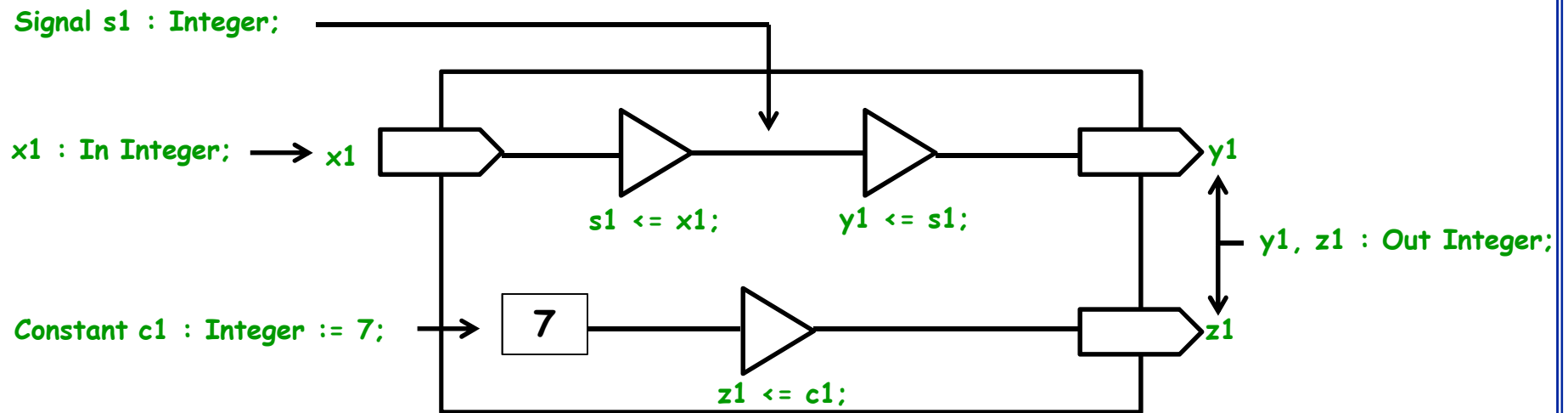
Resumo: Exemplo de uma descrição completa

```
1      ENTITY atrib_1 IS
2          PORT (x1      : IN  INTEGER;
3                y1, z1  : OUT INTEGER);
4      END;
5
6      ARCHITECTURE teste OF atrib_1 IS
7          SIGNAL      s1 :  INTEGER;
8          CONSTANT    c1 :  INTEGER := 7;
9      BEGIN
10         y1 <= s1;
11         s1 <= x1;
12         ...
13         z1 <= c1;
14     END teste;
```

VHDL - Classes de Objetos

Resumo

Representação esquemática do código (slide anterior)



Aula de Hoje

- Operadores

VHDL - Operadores

Operadores

- Realização de operações em sinais, variáveis e constantes.
- Classes de operadores:
 - de atribuição
 - aritméticos
 - de sinais
 - de concatenação
 - de deslocamento
 - relacionais
 - lógicos

VHDL - Operadores

Operadores de atribuição

- Utilizados para a associação de valores às variáveis, sinais e constantes.

OPERADOR	SIGNIFICADO	EXEMPLO
<code><=</code>	Atribuição de sinal	<code>Sig <= '0';</code>
<code>:=</code>	Atribuição de variável	<code>Var := '0';</code>
<code>:=</code>	Inicialização de constantes, sinais e variáveis	<code>Signal sig : BIT := '0';</code>
<code>=></code>	Atribuição de valores únicos em vetores	<code>Vet <= (0 => '1');</code>
<code>=></code>	Atribuição de vários valores em vetores junto com a palavra reservada OTHERS	<code>Vet <= (0 => '1', OTHERS => '0');</code>

VHDL - Operadores

Operadores aritméticos

- Utilizados para a realização de operações matemáticas e empregados em objetos dos tipos INTEGER e REAL.

OPERADOR	SIGNIFICADO	EXEMPLO
+	Soma	$i := i + 1;$
-	Subtração	$j := j - 1;$
*	Multiplicação	$s := a * b;$
/	Divisão	$b := a / c;$
mod	Módulo	$s0 \leftarrow s1 \text{ MOD } s2;$
rem	Resto	$s0 \leftarrow s1 \text{ REM } s2;$
Abs	Valor absoluto	$i0 := \text{ABS } s1;$
**	Exponenciação	$a := b ** c;$

VHDL - Operadores

Operadores aritméticos

- Operadores MOD e REM produzem resultados diferentes quando os valores possuem sinais diferentes na divisão. Usa objetos do tipo inteiro.
 - Operador MOD considera sempre o sinal do dividendo
 - Operador REM considera sempre o sinal do divisor
- Soma de vetores de bits pode ser realizada pelo operador soma (+) que utiliza variáveis, constantes e sinais dos tipos STD_LOGIC e STD_LOGIC_VECTOR, desde que o pacote STD_LOGIC_1164 da biblioteca IEEE seja utilizado.
- Operador ABS retorna o valor absoluto de um número inteiro ou real.
- Operador de exponenciação (**) funciona para valores reais e inteiros na base, mas o expoente deve ser um número inteiro.

VHDL - Operadores

Operadores aritméticos

- Exercício: Considere os valores inteiros $a = 5$, $b = -2$ e $c = 2$. Mostre os resultados para as sentenças com operadores aritméticos em VHDL, a seguir:

$$a/(a+b) = \underline{\hspace{1cm}};$$

$$\text{ABS } (a) + \text{ABS } (b) = \underline{\hspace{1cm}};$$

$$(a/c)*c = \underline{\hspace{1cm}};$$

$$(a+b)**c = \underline{\hspace{1cm}};$$

$$(a*c)/c = \underline{\hspace{1cm}};$$

$$a \text{ REM } b = \underline{\hspace{1cm}};$$

$$a/c = \underline{\hspace{1cm}};$$

$$a \text{ MOD } b = \underline{\hspace{1cm}};$$

VHDL - Operadores

Operadores aritméticos

- Solução: $a = 5$, $b = -2$, $c = 2$.

$$a/(a+b) = 1;$$

$$\text{ABS } (a) + \text{ABS } (b) = 7;$$

$$(a/c)*c = 4;$$

$$(a+b)**c = 9;$$

$$(a*c)/c = 5;$$

$$a \text{ REM } b = 0;$$

$$a/c = 2;$$

$$a \text{ MOD } b = 1;$$

VHDL - Operadores

Operadores de sinais

- Utilizados como função identidade e negação de um valor numérico, podendo alterar o sinal de determinado valor.

OPERADOR	SIGNIFICADO	EXEMPLO
+	Identidade	Var1 := + var2;
-	Negação	Var1 := - var1;

VHDL - Operadores

Operador de concatenação

- Operador & concatena dois vetores, 'vet_a' e 'vet_b', de um mesmo tipo em um terceiro vetor, 'vet_c'. Neste caso, 'vet_c' deve possuir tamanho igual à soma dos tamanhos de 'vet_a' e 'vet_b'.
- Concatenação entre elementos também pode ser realizada, produzindo como resultado um vetor do mesmo tipo dos elementos.
- Exercício: Forneça os resultados da operação de concatenação abaixo.

```
SIGNAL    a : BIT := '0';  
SIGNAL vet0 : BIT_VECTOR (2 DOWNTO 0) := "001";  
SIGNAL vet1 : BIT_VECTOR (3 DOWNTO 0);  
SIGNAL vet2 : BIT_VECTOR (4 DOWNTO 0);  
vet1 <= a & vet0;  
vet2 <= a & vet0 & vet0(0);
```

VHDL - Operadores

Operador de concatenação

- Solução:

```
SIGNAL    a : BIT := '0';  
SIGNAL vet0 : BIT_VECTOR (2 DOWNT0 0) := "001";  
SIGNAL vet1 : BIT_VECTOR (3 DOWNT0 0);  
SIGNAL vet2 : BIT_VECTOR (4 DOWNT0 0);  
  
vet1 <= a & vet0;           --"0001"  
vet2 <= a & vet0 & vet0(0); --"00011"
```

VHDL - Operadores

Operadores de deslocamento

- Utilizados para a realização de operações de deslocamento aritmético, deslocamento lógico e rotação em vetores com elementos dos tipos BIT, STD_LOGIC, STD_ULOGIC ou BOOLEAN.

OPERADOR	SIGNIFICADO
sll	Shift left - deslocamento lógico para a esquerda
srl	Shift right - deslocamento lógico para a direita
sla	Shift left arithmetic - deslocamento aritmético para a esquerda
sra	Shift right arithmetic - deslocamento aritmético para a direita
rol	Rotate left - rotação para a esquerda
ror	Rotate right - rotação para a direita

VHDL - Operadores

Operadores de deslocamento

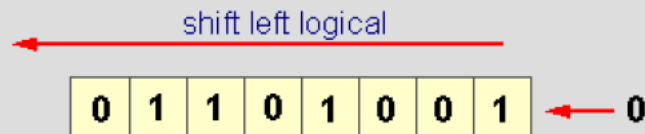
- Nas operações de deslocamento, enquanto o vetor é movido, elementos vão sendo descartados do vetor e novos elementos são inseridos.
- A diferença entre deslocamento aritmético e deslocamento lógico está justamente nos novos elementos a serem inseridos.
 - No deslocamento lógico, os novos elementos são sempre '0' para vetores dos tipos `BIT_VECTOR`, `STD_LOGIC_VECTOR` e `STD_ULOGIC_VECTOR` e `false` para vetores do tipo `BOOLEAN`.
 - No deslocamento aritmético, os novos elementos são repetições do elemento mais à esquerda (`sra`) ou do elemento mais à direita (`sla`).
- Nas operações de rotação ocorre rotatividade dos elementos, ou seja, os elementos são recolocados no final do vetor.

VHDL - Operadores

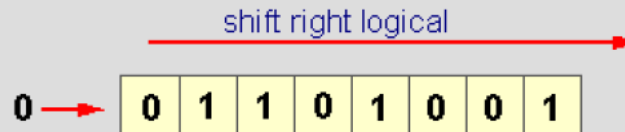
Operadores de deslocamento

- Exemplo:

sll →



srl →



sla →



sra →



rol →



ror →



VHDL - Operadores

Operadores de deslocamento

- Os operadores de deslocamento e de rotação são binários, operando à esquerda um vetor e à direita um valor inteiro que indica o valor de deslocamento ou rotação.

- Exercício:

```
SIGNAL VET1 : BIT_VECTOR (5 DOWNT0 0) := "110011";
```

```
SIGNAL VET2 : BIT_VECTOR (5 DOWNT0 0);
```

```
vet2 <= vet1 SLL 2;
```

```
vet2 <= vet1 SRL 3;
```

```
vet2 <= vet1 SLA 2;
```

```
vet2 <= vet1 SRA 3;
```

```
vet2 <= vet1 ROL 2;
```

```
vet2 <= vet1 ROR 3;
```

VHDL - Operadores

Operadores de deslocamento

- Solução:

```
SIGNAL VET1 : BIT_VECTOR (5 DOWNT0 0) := "110011";
```

```
SIGNAL VET2 : BIT_VECTOR (5 DOWNT0 0);
```

```
vet2 <= vet1 SLL 2;    -- resultado "001100"
```

```
vet2 <= vet1 SRL 3;    -- resultado "000110"
```

```
vet2 <= vet1 SLA 2;    -- resultado "001111"
```

```
vet2 <= vet1 SRA 3;    -- resultado "111110"
```

```
vet2 <= vet1 ROL 2;    -- resultado "001111"
```

```
vet2 <= vet1 ROR 3;    -- resultado "011110"
```

VHDL - Operadores

Operadores relacionais

- Utilizados para a comparação de dois elementos de um mesmo tipo e o resultado de uma operação relacional é sempre do tipo booleano.

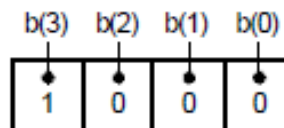
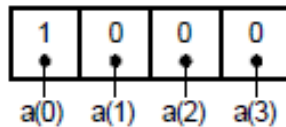
OPERADOR	SIGNIFICADO
=	Equivalência
/=	Desigualdade
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

VHDL - Operadores

Operadores relacionais

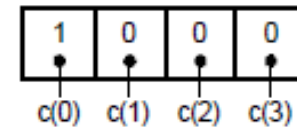
- Exercício: Com base nos vetores de bits abaixo, marque V para verdadeiro (true) ou F para falso (false).

CONSTANT a: BIT_VECTOR(0 TO 3) := "1000"

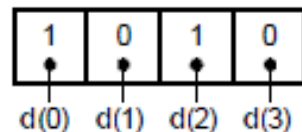


CONSTANT b: BIT_VECTOR(3 DOWNT0 0) := "1000"

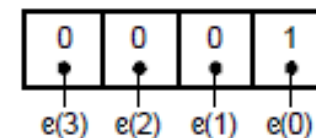
CONSTANT c: BIT_VECTOR(0 TO 3) := "1000"



CONSTANT d: BIT_VECTOR(0 TO 3) := "1010"



CONSTANT e: BIT_VECTOR(3 DOWNT0 0) := "0001"



☐ a = b

☐ e < c

☐ b > d

☐ d > a

☐ a = c

☐ e <= a

☐ b /= c

☐ d < e

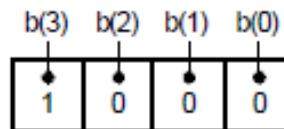
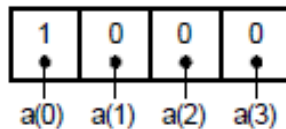
☐ c >= d

VHDL - Operadores

Operadores relacionais

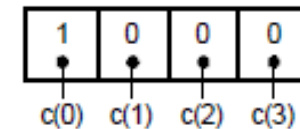
- Solução:

CONSTANT a: BIT_VECTOR(0 TO 3) := "1000"

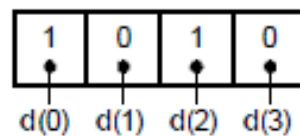


CONSTANT b: BIT_VECTOR(3 DOWNTO 0) := "1000"

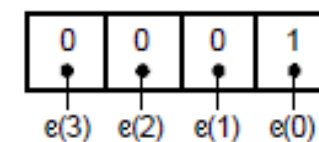
CONSTANT c: BIT_VECTOR(0 TO 3) := "1000"



CONSTANT d: BIT_VECTOR(0 TO 3) := "1010"



CONSTANT e: BIT_VECTOR(3 DOWNTO 0) := "0001"



(V) a = b

(V) e < c

(F) b > d

(V) d > a

(V) a = c

(V) e <= a

(F) b /= c

(F) d < e

(F) c >= d

VHDL - Operadores

Operadores lógicos

- Utilizados para a implementação das principais funções lógicas da álgebra booleana.
- Os operadores lógicos são definidos para os tipos BIT, BOOLEAN, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR, STD_ULOGIC e STD_ULOGIC_VECTOR.
- No caso de vetores, os operandos devem possuir o mesmo tamanho e a operação ocorre comparando os elementos dos arrays bit a bit.

VHDL - Operadores

Operadores lógicos

OPERADOR	EXPRESSÃO	EXEMPLO
NOT	\bar{a}	NOT a
AND	$a \bullet b$	a AND b
OR	$a + b$	a OR b
XOR	$a \oplus b$	a XOR b
NAND	$\overline{a \bullet b}$	a NAND b
NOR	$\overline{a + b}$	a NOR b
XNOR	$\overline{a \oplus b}$	a XNOR b

VHDL - Operadores

Operadores lógicos

- As operações lógicas possuem a mesma prioridade, exceto a operação NOT, o que muitas vezes torna necessário o uso de parênteses para determinar a precedência correta dos operadores.
- Exercício: Use parênteses, quando for necessário, para determinar a prioridade de operadores lógicos das expressões booleanas abaixo:

$$s1 = a \bullet b + c$$

$$s2 = a \bullet b + \overline{c \bullet d}$$

$$s3 = \overline{a} \bullet d + \overline{b + c}$$

VHDL - Operadores

Operadores lógicos

- Solução:

$$s1 = a \bullet b + c$$

$$s2 = a \bullet b + \overline{c \bullet d}$$

$$s3 = \overline{a} \bullet d + \overline{b + c}$$

s1 <= a AND b OR c;	-- incorreto
s1 <= (a AND b) OR c;	-- correto
s2 <= a AND b OR c NAND d;	-- incorreto
s2 <= (a AND b) OR (c NAND d);	-- correto
s3 <= NOT a AND d OR b NOR c;	-- incorreto
s3 <= ((NOT a) AND d) OR (b NOR c);	-- correto

VHDL - Operadores

Precedência de operadores

PRECEDÊNCIA	OPERADORES	FUNÇÃO
<p>Menor</p> <p>↓</p> <p>Maior</p>	AND, OR, NAND, NOR, XOR, XNOR	Operadores lógicos
	=, /=, <, >, =<, >=	Operadores relacionais
	SLL, SLR, SLA, SRA, ROL, ROR	Operadores de deslocamento
	+, -, &	Soma, subtração e concatenação
	+, -	Sinais positivo e negativo
	*, /, MOD, REM	Multiplicação, divisão, módulo e resto
	**, ABS, NOT	Exponenciação, valor absoluto e negação lógica

VHDL - Operadores

Precedência de operadores

- Nota 1: Os operadores de uma mesma classe possuem o mesmo nível de precedência.
- Nota 2: Uma forma de explicitar que determinadas operações sejam executadas antes que outras é pelo uso de parênteses.
- Nota 3: A função NOT possui precedência sobre as demais e é executada primeiro, caso a prioridade não seja forçada com o uso de parênteses.
- Nota 4: Quando uma expressão contém duas operações que estão em um mesmo nível de precedência, a operação mais à esquerda será analisada primeiro.
- Nota 5: Devido à rigidez da linguagem, VHDL, com relação a tipos, normalmente os operandos de uma operação são do mesmo tipo.

Resumo da Aula de Hoje

Tópicos mais importantes:

- Classes de Objetos
- Operadores

Próxima da Aula

- **Comandos Condicionais**
 - Comando *WHEN ELSE*
 - Comando *IF THEN ELSE*
 - Comando *CASE WHEN*