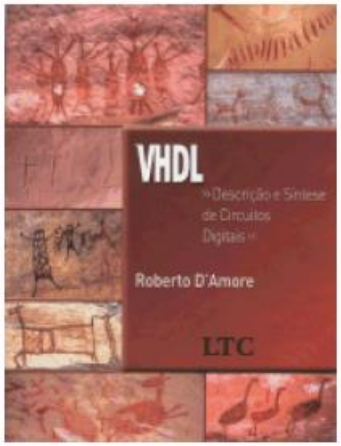
SEL0384/SEL0606 – Laboratório de Sistemas Digitais
Aula 03 – Revisão

Prof. Dr. Maximiliam Luppe

Livro adotado:

VHDL - Descrição e Síntese de Circuitos Digitais
Roberto d'Amore

ISBN 85-216-1452-7
Editora LTC www.ltceditora.com.br



Para informações adicionais consulte: **www.ele.ita.br/~damore/vhdl**

Primeiro contato com a linguagem

Tópicos

- **Entidade de projeto**

  Declaração da entidade e declaração da arquitetura

- **Classes de objetos**: constante, variável e sinal
- **Tipos**

  Tipos escalares

  Tipos compostos

  Definição de novos tipos

- **Operadores**
- **Exemplos de utilização**

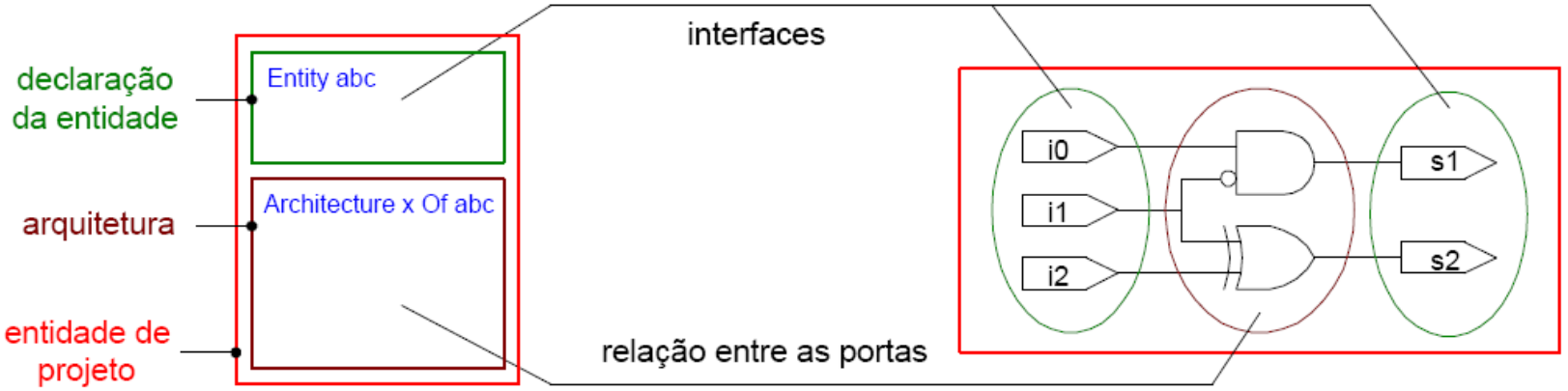
Entidade de projeto

- **Pode representar:**

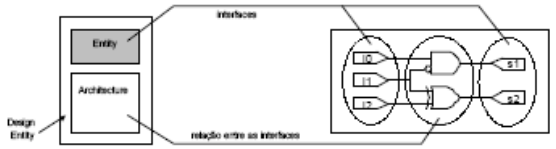
  uma simples porta lógica ..... a um sistema completo

- **Composta de duas partes:**

  - Declaração da entidade
    - define portas de entrada e saída da descrição
    - equivalente ao símbolo de um bloco em captura esquemática
  - Arquitetura
    - descreve as relações entre as portas
    - equivalente ao esquema contido no bloco em cap. esquemática



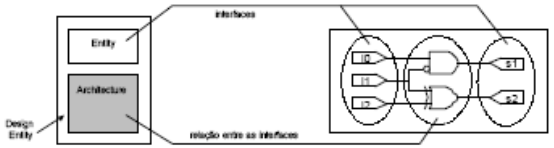
Declaração da entidade



- ENTITY: inicia a declaração
- PORT: define modo e tipo das portas
  - modo IN : entrada
  - modo OUT : saída
  - modo BUFFER : saída - pode ser referenciada internamente
  - modo INOUT : bidirecional
- END: termina a declaração

```
ENTITY entidade_abc IS

  PORT (x0, x1     : IN        tipo_a;    -- entradas
        y0, y1     : OUT       tipo_b;    -- saidas
        y2         : BUFFER    tipo_c;    -- saida
        z0, z1     : INOUT     tipo_d);   -- entrada / saida

END entidade_abc;
```

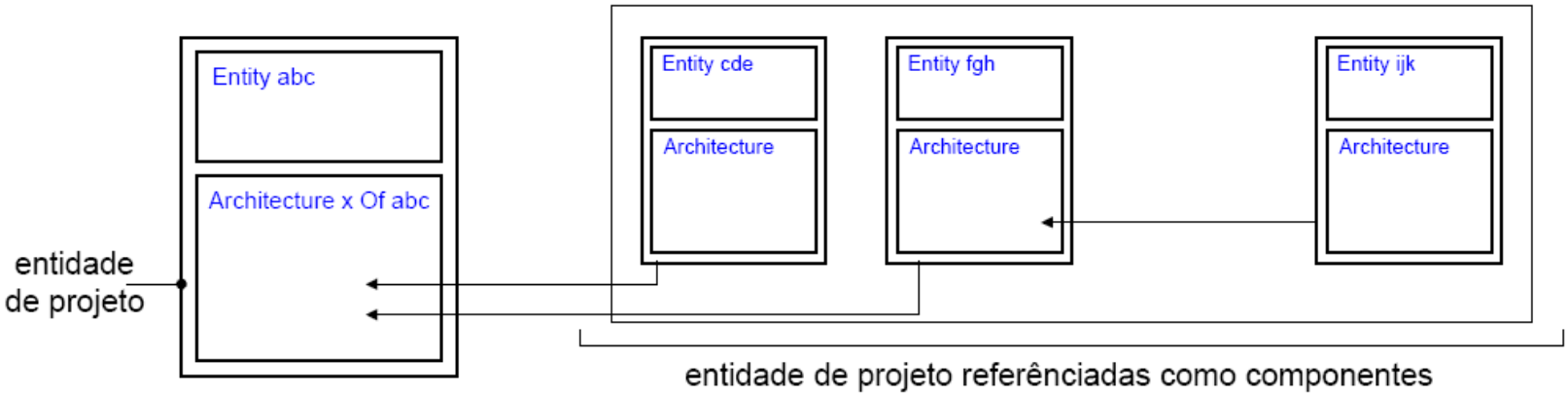
Declaração da arquitetura



- ARCHITECTURE: inicia a declaração
- linhas que seguem podem conter:
  - declaração de sinais e constantes
  - declaração de componentes referenciados
  - descrição de subprogramas locais
  - definição de novos tipos
- BEGIN: inicia a descrição
- END: termina a descrição

```
ARCHITECTURE estilo_abc OF entidade_abc IS
  -- declaracoes de sinais e constantes
  -- declaracoes de componentes referenciados
  -- descricao de sub-programas locais
  -- definicao de novos tipos de dados locais
  --
BEGIN
  --
  -- declaracoes concorrentes
  --
END estilo_abc;
```

- **Uma entidade de projeto**
  - pode ser descrita na forma: interligação de outras entidades
  - estabelece um projeto hierárquico
  - não existe limite para o nível de hierarquia
  - diferentes estilos de descrição podem ser empregados



Componentes

- **Componente:**
  - uma descrição (entidade + arquitetura) empregada por uma outra entidade
- **Emprego:**
  - interligação de múltiplas entidades de projeto
    - projeto hierárquico
- **Declaração de um componente** (primeiro passo para utilização de um componente)
  - similar a declaração de entidade
  - exemplo:

```
COMPONENT nome_componente
  PORT   (sinal_a      : modo_a  tipo_sinal_a;
          sinal_b      : modo_b  tipo_sinal_b;
          sinal_c      : modo_c  tipo_sinal_c);
END COMPONENT;
```

Componentes

- **A solicitação de um componente contém:**
  - rótulo de denominação qualquer (x1, y2 ...)
  - nome do componente (nome_componente)
  - mapa de ligações (Port Map (....))

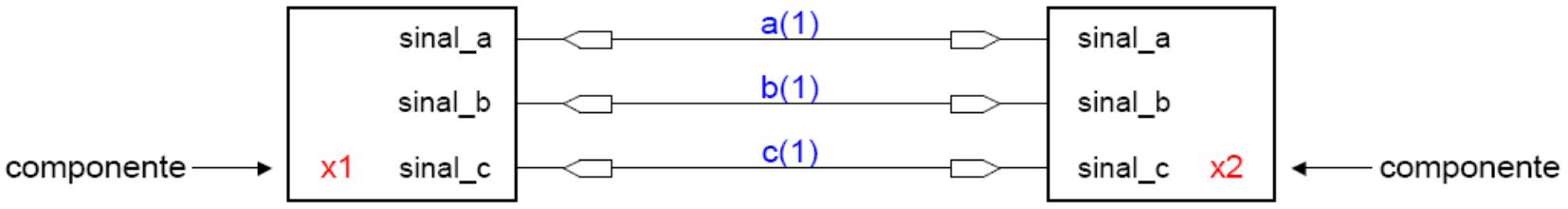
- **Mapa de ligações**:

  - pode seguir mesma ordem estabelecida na declaração do componente:

```
x1: nome_componente PORT MAP(a(1), b(1), c(1));
```

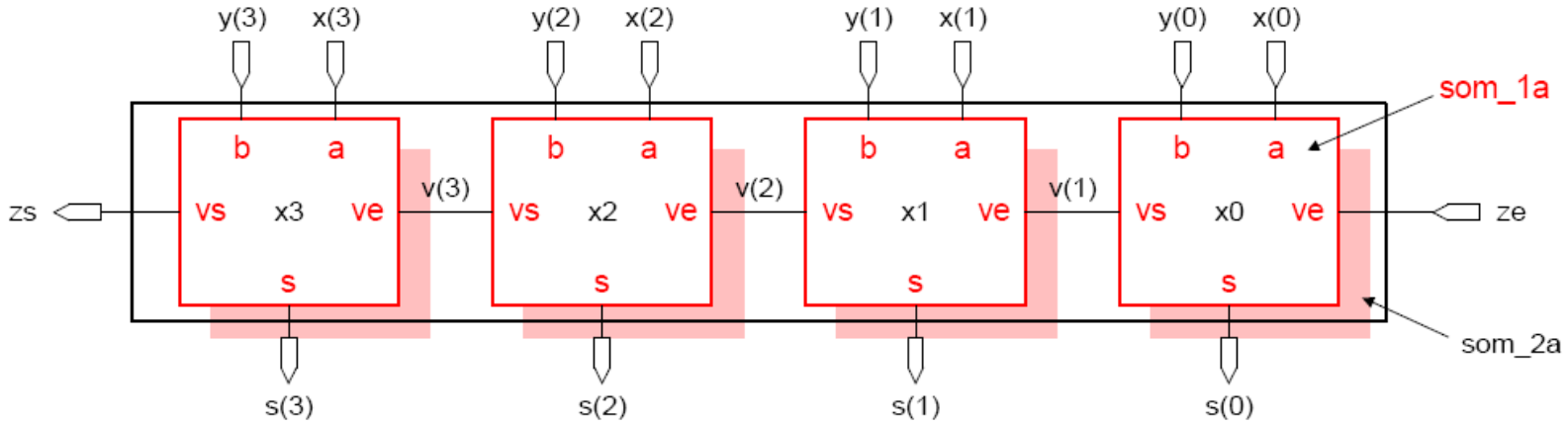
  - pode seguir uma nova seqüência definida no mapa:

```
x2: nome_componente PORT MAP(sinal_b => b(1),sinal_a => a(1),sinal_c => c(1));
```



Componentes

- **Exemplo**: Somador de 4 bits formado por somadores de 1 bit

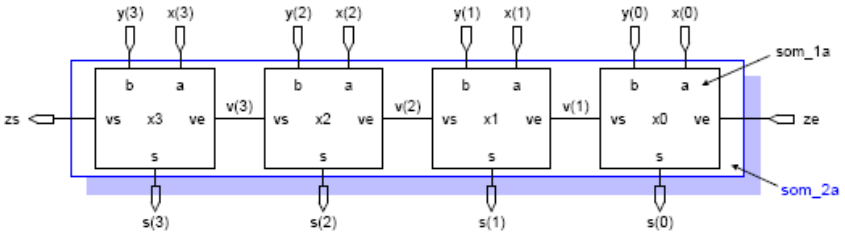


- **Descrição do somador de 1 bit** (componente a ser referenciado)

```
 1 ENTITY som_1a IS
 2   PORT    (a, b, ve  : IN  BIT;
 3            s, vs     : OUT BIT);
 4 END som_1a;
 5
 6 ARCHITECTURE teste OF som_1a IS
 7
 8 BEGIN
 9   s  <=  a XOR b  XOR ve;                        -- soma
10   vs <= (a AND b) OR (a AND ve) OR (b AND ve); -- vai um
11 END teste;
```

Componentes

- **Descrição do somador empregando o componente**: somador de 1 bit




```
 1 ENTITY som_2a IS
 2   PORT   ( x, y   : IN  BIT_VECTOR (3 DOWNTO 0);  -- entradas do somador
 3            ze     : IN  BIT;                      -- entrada  vem um
 4            s      : OUT BIT_VECTOR (3 DOWNTO 0);  -- soma
 5            zs     : OUT BIT );                    -- vai um
 6 END som_2a;
 7
 8 ARCHITECTURE estrutural OF som_2a IS
 9
10 COMPONENT som_1a
11   PORT    (a, b, ve : IN  BIT;  s, vs : OUT BIT);
12 END COMPONENT;
13
14 SIGNAL v    : BIT_VECTOR (3 DOWNTO 1);  -- vai um interno
15
16 BEGIN
17   x0: som_1a PORT MAP(    x(0),      y(0),     ze,        s(0),      v(1));
18   x1: som_1a PORT MAP(    x(1),      y(1),     v(1),      s(1),      v(2));
19   x2: som_1a PORT MAP(b =>y(2), a => x(2), s =>s(2), ve =>v(2), vs =>v(3));
20   x3: som_1a PORT MAP(    x(3),      y(3),     v(3),      s(3),      zs);
21 END estrutural;
```

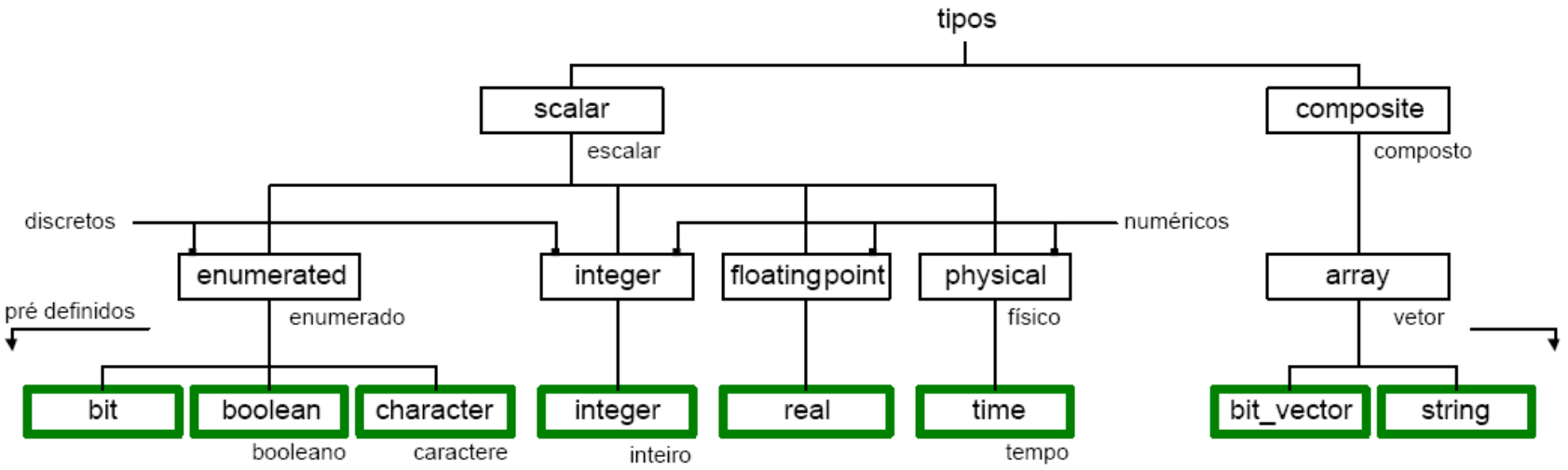
Classe de objetos: constante, variável e sinal

- Objetos: são elementos que contêm um valor armazenado
- Quatro tipos:
  - CONSTANT: valor estático
  - VARIABLE: valor imposto pode ser alterado
    regiões de código seqüencial
  - SIGNAL: valor imposto pode ser alterado
    regiões de código seqüencial e concorrente
  - FILE: associado a criação de arquivos
- Exemplos de atribuição de valores entre objetos

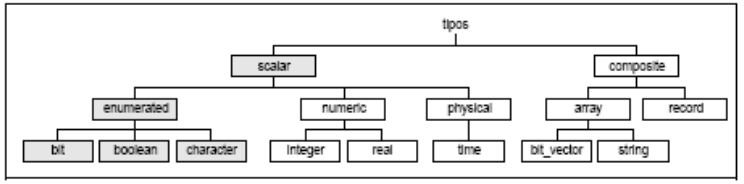
```
sinal_2    <= sinal_1;        -- atribuicao valor para sinal
sinal_3    <= variavel_1;     -- atribuicao valor para sinal
sinal_4    <= constante_1;    -- atribuicao valor para sinal

variavel_2 := sinal_1;        -- atribuicao valor para variavel
variavel_3 := variavel_1;     -- atribuicao valor para variavel
variavel_4 := constante_1;    -- atribuicao valor para variavel
```

Tipos

- **Objetos**: devem ser declarados segundo uma especificação de tipo
- **Objetos de tipos diferentes**: não é permitida a transferência de valores
- **Alguns tipos definidos no pacote padrão VHDL**:



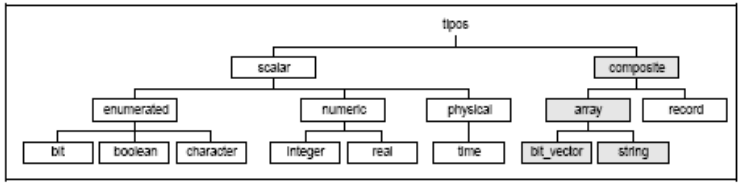
Tipos escalares



- **Classe**: enumerados
  - BIT: empregado para representar níveis lógicos alto e baixo
  - BOOLEAN: empregado em comandos que executam uma decisão

classe	tipo	valor	exemplos
enumerado	BIT	um, zero	1, 0
	BOOLEAN	verdadeiro, falso	TRUE, FALSE

Tipos compostos

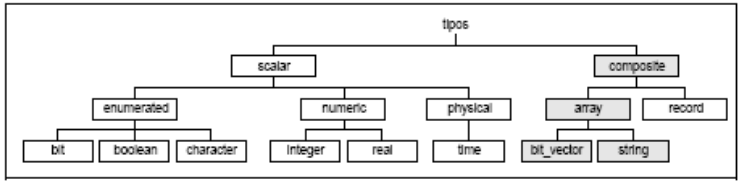


- **Classes**:
  - vetor (array): agrupa elementos de um mesmo tipo

- Classe array (definidos no pacote padrão):
  - BIT_VECTOR: vetor contendo elementos tipo bit

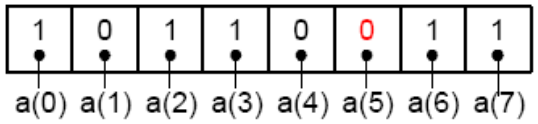
Classe	tipo	valor	exemplos
vetor	BIT_VECTOR	1 , 0	”1010”, B”10_10”, O”12”, X”A”

Tipos compostos

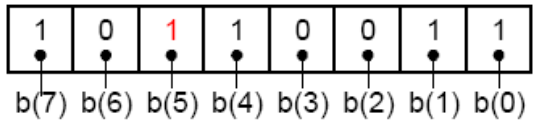


- **Declarações**:
  - limites definidos por TO e DOWNTO
- **Exemplos de declarações tipos**: bit_vector

SIGNAL a: BIT_VECTOR(0 TO 7) := "10110011"



SIGNAL b: BIT_VECTOR(7 DOWNTO 0) := "10110011"



- **Operadores lógicos**:

  - operandos: tipos bit e boolean

    vetores unidimensionais compostos de elementos bit e boolean

    (exemplo: bit_vector)

  - os vetores devem ter o mesmo tamanho

  - operação executada entre elementos de mesma posição

operadores	operando L	operando R	retorna
not and or	bit	bit	bit
nand nor xor xnor	boolean	boolean	boolean
Nota: O operador not pertence a classe diversos			

- tipo bit: 0,1 tipo boolean: 0 = false 1 = true

L	R	not L	L and R	L nand R	L or R	L nor R	L xor R	L xnor R
1	1	0	1	0	1	0	0	1
1	0	0	0	1	1	0	1	0
0	1	1	0	1	1	0	1	0
0	0	1	0	1	0	1	0	1

- **Operadores adição**
  - adição e subtração: tipo numérico
  - concatenação: vetor unidimensional e elementos (mesmo tipo)

operadores	operando L	operando R	retorna
+ -	tipo numérico	o mesmo tipo de L	mesmo tipo
&	vetor	mesmo vetor de L	vetor mesmo tipo
	vetor	elemento	vetor mesmo tipo
	elemento	vetor	vetor mesmo tipo
	elemento	elemento	vetor

- **Exemplo**:

```
a <= b & c;    -- "a" bit_vector 8 elementos,"b","c" bit_vetor 4 elementos
x <= y & '1'; -- "x" bit_vector 5 elementos,    "y" bit_vetor 4 elementos
```

- **Exemplo**: Atribuição de valores em sinais, tipos BIT_VECTOR
  - operação: valor 1011 atribuído a todas as portas de saída
  - diferentes bases de representação
    tipos integer, real formato: 16#B# 16#B.0# (vide exemplo anterior)
    tipos bit_vector formato: X"B"
  - linha 12: caracter _ em 01_0_11 melhora leitura do valor
  - linhas 14 e 15: valor definido para uma parte do vetor pal. res.: DOWNTO

```
 5 ARCHITECTURE teste OF std_a IS
 6   CONSTANT c1   : BIT_VECTOR(4 DOWNTO 0) := "01011";  -- constante
 7   CONSTANT zero : BIT := '0';
 8   CONSTANT um   : BIT := '1';
 9 BEGIN
10   s1 <= c1;                -- valor atraves de constante
11   s2 <= "01011";           -- valor (01011) direto - base binaria
12   s3 <= B"01_0_11";        -- valor (01011) direto - base binaria com separadores
13   s4 <= '0' & X"B";        -- bit (0) concatenado com valor hexadecimal (1011)
14   s5(4 DOWNTO 3) <= "01";           -- valor (01), parte do vetor
15   s5(2 DOWNTO 0) <= zero & um & um;  -- valor (010), parte com concatenacao
16 END teste;
```

- **Exemplo**: Atribuição de valores - expressões com operadores lógicos

  - linhas 8 a 14: comentário no código

  - linha 15: x nand y equivale a not (a and b) → ordem das operações importa

```
 1 ENTITY std_xa1 IS
 2   PORT( a, b, c, d          : IN  BIT;
 3         x1, x2, x3, x4, x5 : OUT BIT);
 4 END std_xa1;
 5
 6 ARCHITECTURE exemplo OF std_xa1 IS
 7 BEGIN
 8    x1 <= a OR NOT b;              -- Certo:  operador NOT tem precedencia
 -                                   --         mais elevada
 9    x2 <= a AND b AND c;           -- Certo:  operadores iguais
10 -- x3 <= a AND b  OR c;           -- Errado: expressao ambigua x3=(a.b)+c
 -                                   --                      ou x3=a.(b+c) ?
11    x3 <=(a AND b) OR c;           -- Certo:  empregando parentesis
12 -- x4 <= a AND b  OR  c AND d;    -- Errado: expressao ambigua, operadores
 -                                   --         com mesma precedecia
13    x4 <=(a AND b) OR (c AND d);   -- Certo:  x4 = a.b + c.d
14 -- x5 <= a NAND b  NAND c;        -- Errado: operadores com negacao
 -                                   --         necessitam parentesis
15    x5 <=(a NAND b) NAND c;        -- Certo:  operador com negacao entre
 -                                   --         parentesis
16 END exemplo;
```

- **Exemplo**: Operadores classe adição

  - linhas 10 e 11: operação de concatenação de dois vetores (tipos bit_vector)

  - linha 12: soma de dois tipos inteiros

```
 1 ENTITY std_xc IS
 2   PORT (bv_a,  bv_b  : IN  BIT_VECTOR(1 DOWNTO 0);
 3         int_a, int_b : IN  INTEGER RANGE -32 TO 31;
 4         bv_c,  bc_d  : OUT BIT_VECTOR(3 DOWNTO 0);
 5         int_c        : OUT INTEGER RANGE -64 TO 63);
 6 END std_xc;
 7
 8 ARCHITECTURE teste OF std_xc IS
 9 BEGIN
10   bv_c  <= bv_a & bv_b;
11   bc_d  <= bv_a & '1' & '0';
12   int_c <= -int_a +int_b;
13 END teste;
```