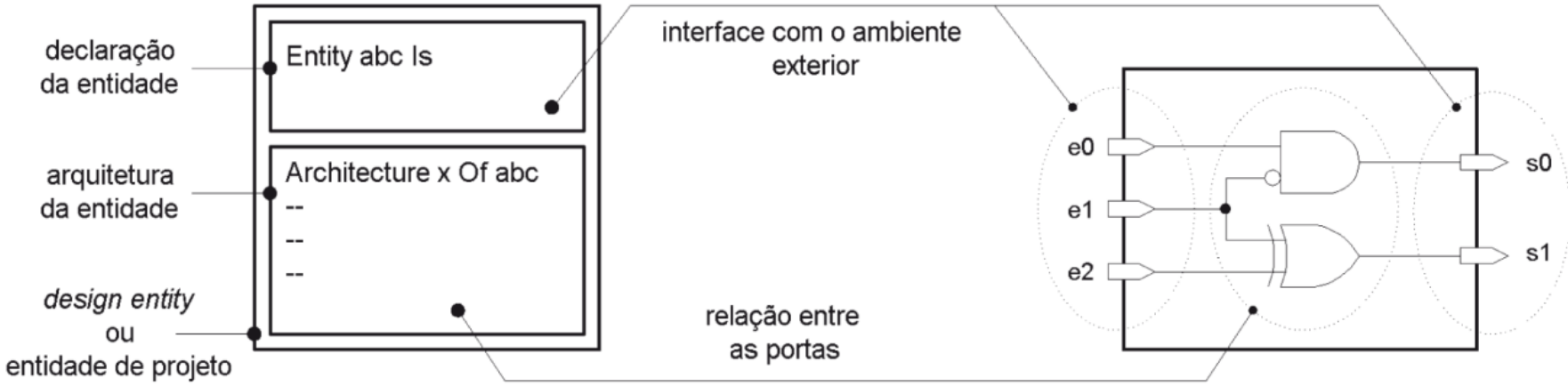
Elementos básicos de um código VHDL

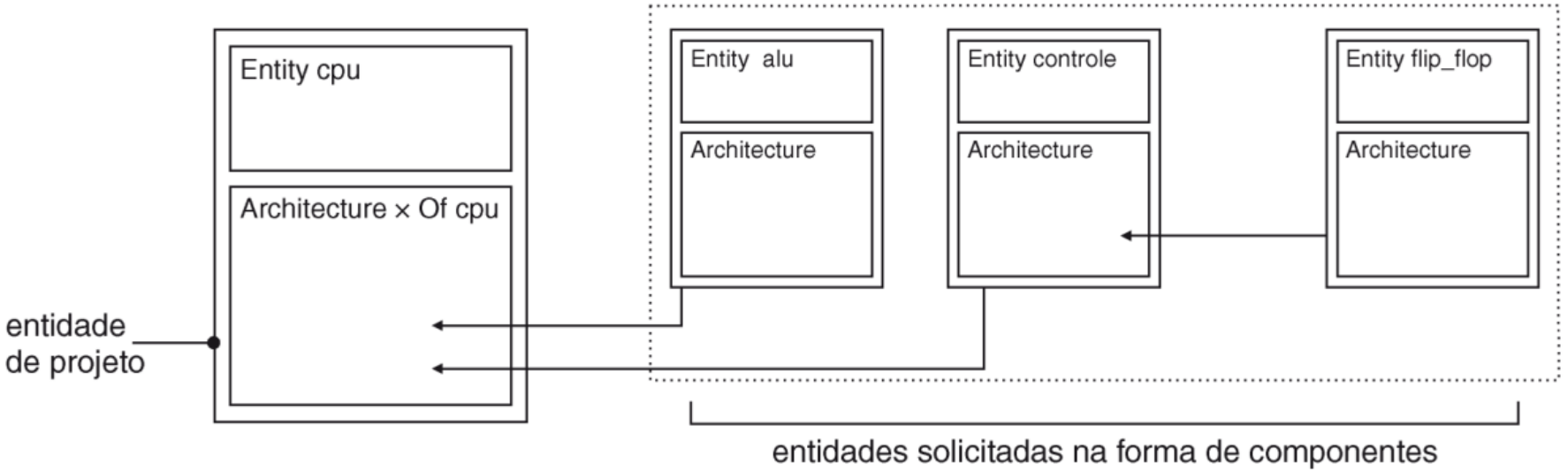
FGA - UnB

Prática de Eletrônica Digital 1

Prof. Henrique M. T. Menegaz

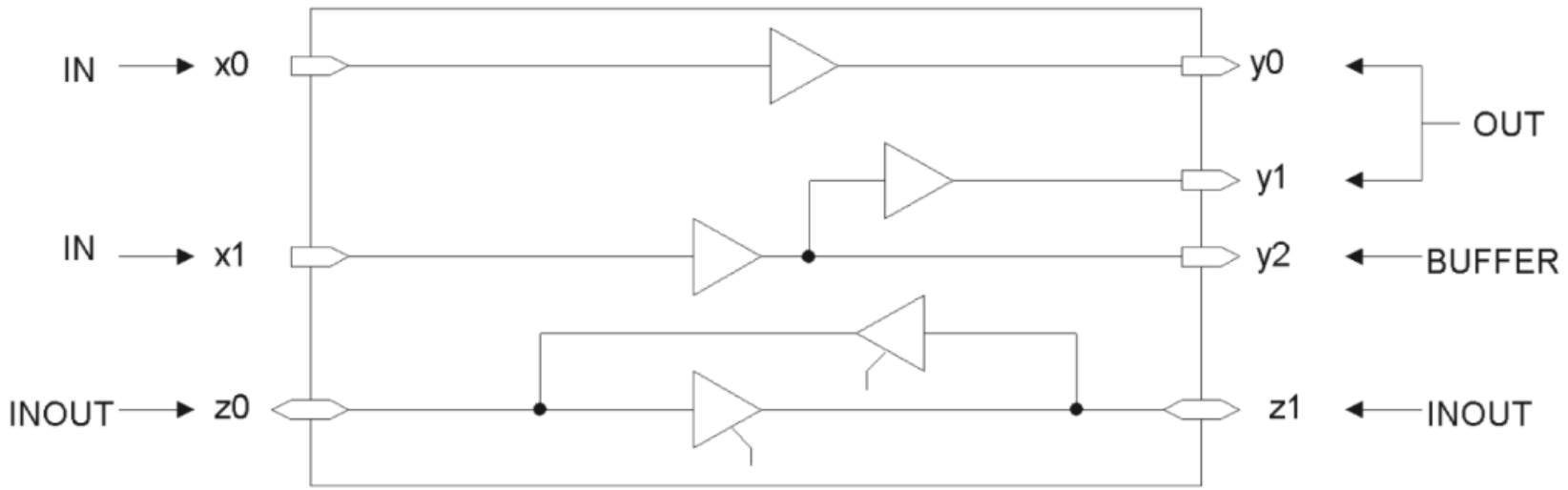
Entidade de projeto (Design entity)





Declaração da entidade (ENTITY)

```
ENTITY entidade_abc IS

  GENERIC (n        : INTEGER := 5);

  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;
```



Arquitetura (ARCHITECTURE)

```
ARCHITECTURE nome_identificador OF entidade_abc IS
  --
  -- regiao de declaracoes:
  --   declaracoes de sinais e constantes
  --   declaracoes de componentes referenciados
  --   declaracao e corpo de subprogramas
  --   definicao de novos tipos de dados locais
  --
BEGIN
  --
  -- comandos concorrentes
  --
END;
```

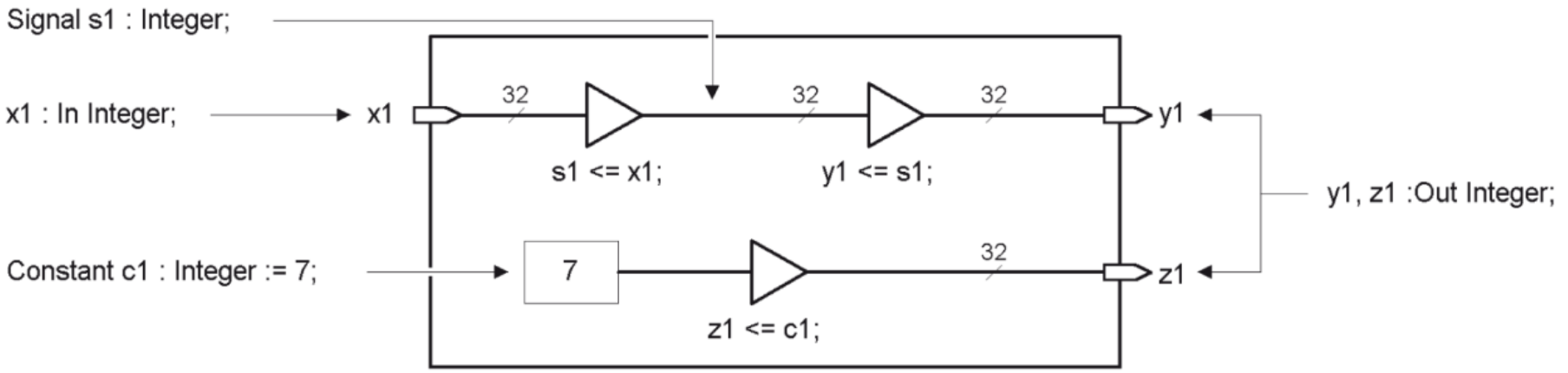
Classes de Objeto

```
-- classe   lista de nomes          tipo      valor inicial

   CONSTANT nome_da_constante_a : tipo_x;                        -- constante sem valor inicial
   CONSTANT nome_da_constante_a : tipo_x := valor_inicial;
   CONSTANT fixo_1, fixo_2      : tipo_x := valor_inicial;

   VARIABLE nome_da_variavel_c  : tipo_z;                        -- variavel sem valor inicial
   VARIABLE nome_da_variavel_d  : tipo_z := valor_inicial;       -- variavel com valor inicial
   VARIABLE var_1, var_2        : tipo_z := valor_inicial;       -- variaveis: mesmo tipo e valor

   SIGNAL   nome_do_sinal_a       : tipo_y;                      -- sinal sem valor inicial
   SIGNAL   nome_do_sinal_b       : tipo_y := valor_inicial;     -- sinal com valor inicial
   SIGNAL   nome_x, nome_y, nome_z : tipo_y;                     -- sinais do mesmo tipo
```

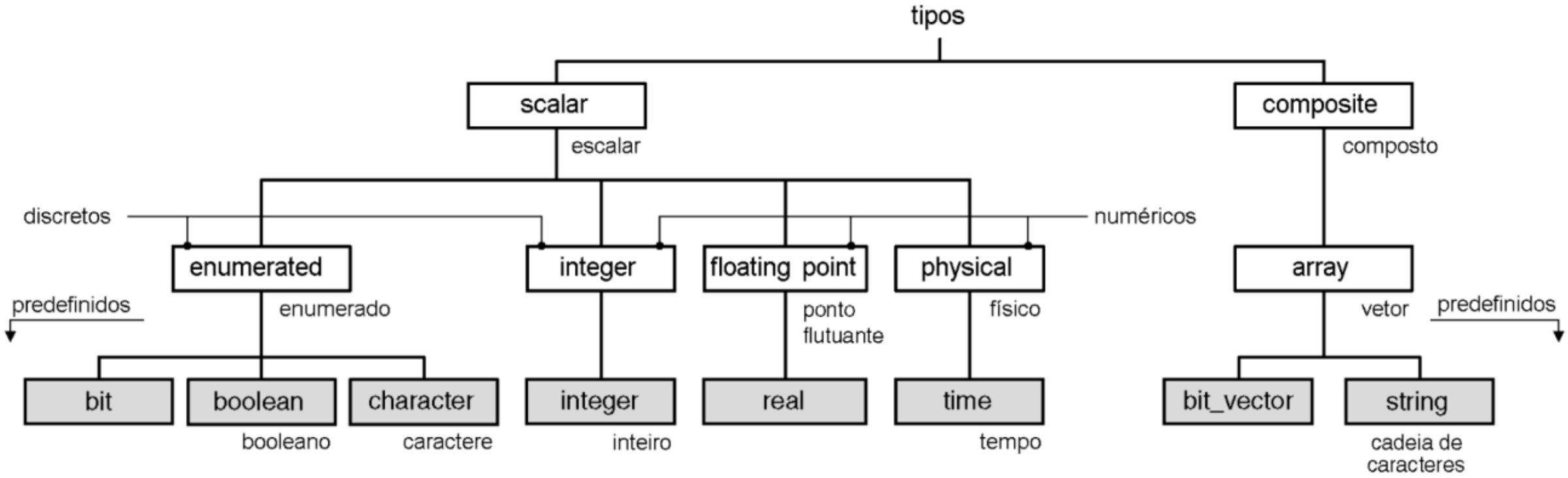
```
  sinal_2    <= sinal_1;        -- atribuicao valor para sinal
  sinal_3    <= variavel_1;     -- atribuicao valor para sinal
  sinal_4    <= constante_1;    -- atribuicao valor para sinal

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

Classes de Objeto

```
 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;
```



Tipos do Pacote Padrão (“LIBRARY std; USE std.standard.all;”)



Tipos simples do Pacote Padrão (“LIBRARY std; USE std.standard.all;”)

Tipo predefinido	Valor	Exemplos
BIT	um, zero	`1, 0`
BOOLEAN	verdadeiro, falso	`TRUE, FALSE`
CHARACTER	caracteres ASCII	`a, b, c, A, B, C, ?, (`
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#`
POSITIVE	$1 \leq x \leq 2^{31}-1$	`2#11_11_011#`
REAL	$-3.65\times10^{47} \leq x \leq +3.65\times10^{47}$	`1.23, 1.23E+2, 16#7.B#E+1`
TIME	ps = 10^3 fs ns = 10^3 ps us = 10^3 ns ms = 10^3 us sec = 10^3 ms min = 60 sec hr = 60 min	`1 us, 100 ps, 1 fs`

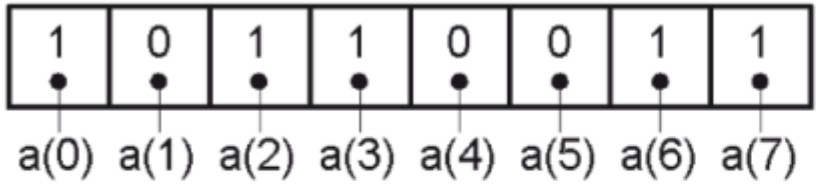
Nota a: “ NATURAL ” e “ POSITIVE ” são subtipos “ INTEGER ”.

Nota b: na versão VHDL-1993, valores para tipos “ CHARACTER ” estendidos para o conjunto ISO8859-1.

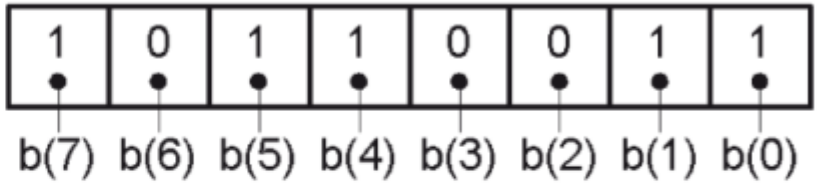
Tipos compostos

Tipo predefinido	Valor	Exemplos
BIT_VECTOR	1, 0	`"1010", B"10_10", O"12", X"A"`
STRING	tipo character	`"texto", ""incluindo_aspas""`

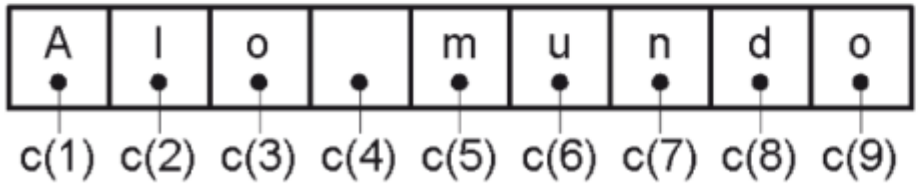
Signal a: Bit_Vector(0 TO 7) := "10110011"



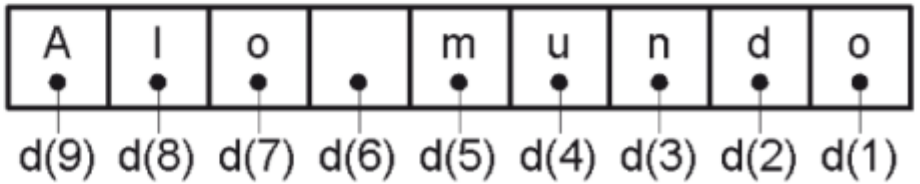
Signal b: Bit_Vector(7 DOWNTO 0) := "10110011"



Constant c: String(1 TO 9) := "Alo mundo"

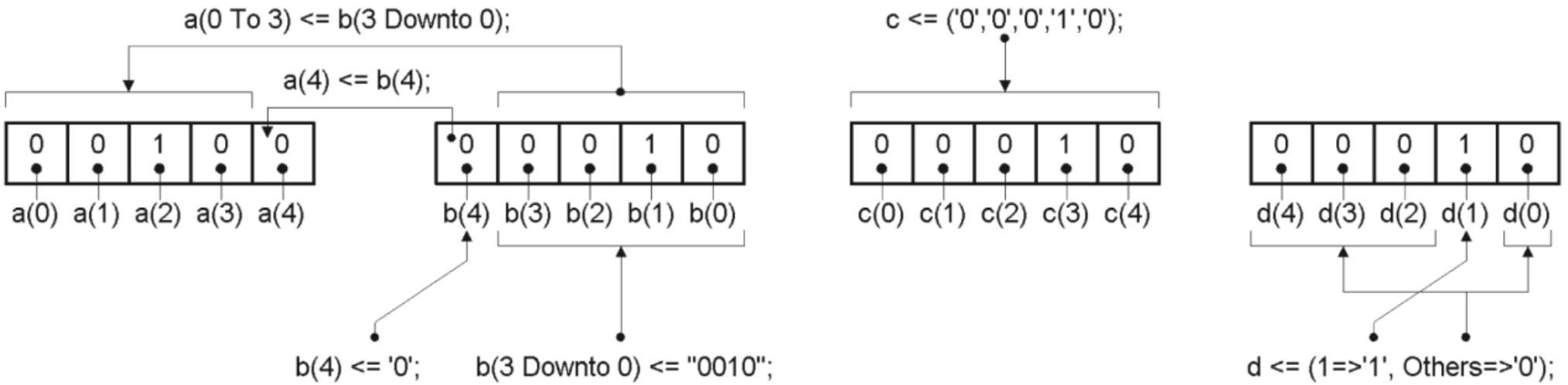


Variable d: String(9 DOWNTO 1) := "Alo mundo"



Tipos compostos

```
Signal   a, c: Bit_Vector(0 To 4);
Signal   b, d: Bit_Vector(4 Downto 0);
```




```
a(4)       <= b(4);              -- 1 elemento     <= 1 elemento
a(0 TO 3)  <= b(3 DOWNTO 0);     -- parte do vetor <= parte do vetor
b(4)       <= '0';               -- 1 elemento     <= valor
b(3 DOWNTO 0) <= "0010";         -- parte do vetor <= valor

c <= ('0','0','0','1','0');     -- valor 00010 agregado notacao posicional
d <= (1=>'1', OTHERS=>'0');      -- valor 00010 agregado associacao por nomes
```

Operadores

Precedência	Classe	Operadores
menor	lógicos	and or nand nor xor xnor
.	relacionais	= /= < <= > >=
.	deslocamento	sll srl sla sra rol ror
.	adição	+ – &
.	sinal	+ –
.	multiplicação	* / mod rem
maior	diversos	** abs not

Nota a: o operador " not " está inserido na classe diversos devido à sua maior precedência.

Nota b: as operações " xor ", " sll ", " srl ", " sla ", " rol " e " ror " não são definidas na versão VHDL-1987.

Operadores lógicos

Operadores	Operando L	Operando R	Retorna
not	bit	mesmo tipo de L	mesmo tipo de L
and or xor	boolean	mesmo tipo de L	mesmo tipo de L
nand nor xnor	vetor unidimensional com elementos do tipo bit ou boolean	mesmo tipo de L	mesmo tipo de L
Exemplos: z <= x AND y; sinal_k <= NOT var_a1; x1:= a OR NOT b;			
Nota a: o operador " not " pertence à classe diversos e possui unicamente o operando L. Nota b: o operador " xnor " não é definido na versão VHDL-1987.			

Operadores relacionais

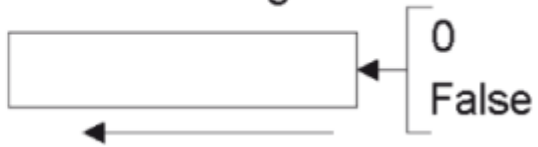
Operadores	Operando L	Operando R	Retorna
= /=	qualquer tipo	mesmo tipo de L	boolean
> < >= <=	qualquer tipo escalar	mesmo tipo de L	boolean
	vetor unidimensional com elementos do tipo inteiro ou enumerado	mesmo tipo de L	boolean
Exemplos: IF x /= y THEN ....; WHILE NOT z = valor_final			

Operadores de deslocamento

Operadores	Operando L	Operando R	Retorna
sll srl sla sra rol ror	vetor unidimensional com elementos bit ou boolean	integer	mesmo tipo de L
Exemplos: z <= x SLL 1; sinal_a <= x ROR n_shift;			

sla shift left arithmetic

sll shift left logical



rol

Operadores de adição

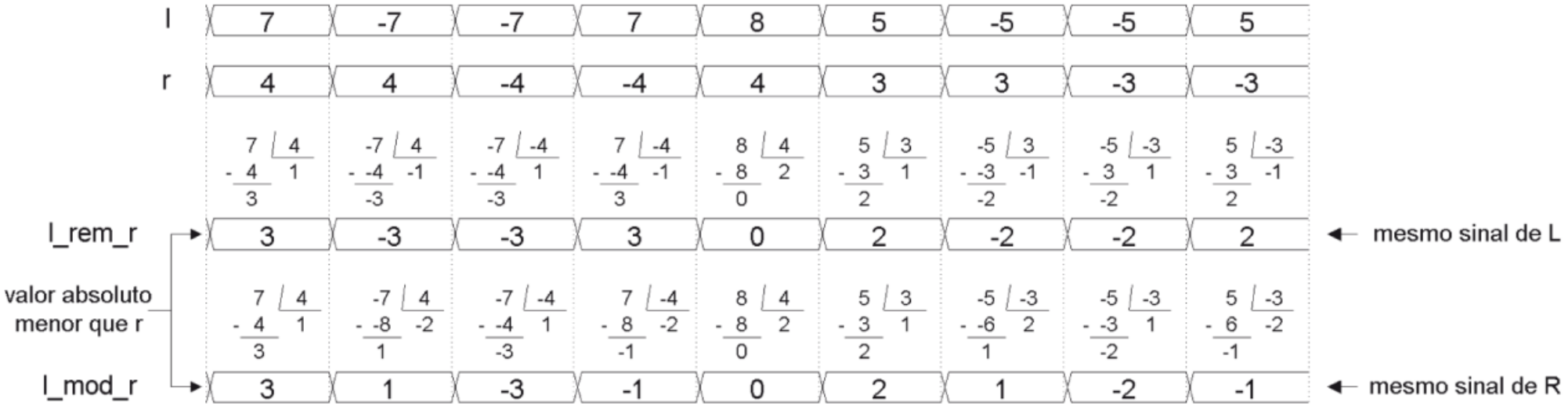
Operadores	Operando L	Operando R	Retorna
+ -	tipo numérico	o mesmo tipo de L	mesmo tipo
&	vetor unidimensional	mesmo vetor de L	vetor mesmo tipo
	vetor unidimensional	elemento	vetor mesmo tipo
	elemento	vetor unidimensional	vetor mesmo tipo
	elemento	elemento	vetor
Exemplos: z := x + 1; s4 <= '0' & X"B";			
Nota: na operação " & " os vetores ou elementos devem ser do mesmo tipo.			

Operadores de sinal

Operadores	Operando	Retorna
+ -	qualquer tipo numérico	mesmo tipo
Exemplos: a <= -b;		

Operadores aritméticos de multiplicação

Operadores	Operando L	Operando R	Retorna
* /	qualquer tipo integer	mesmo tipo	mesmo tipo
	qualquer tipo real	mesmo tipo	mesmo tipo
mod rem	qualquer tipo integer	mesmo tipo	mesmo tipo
Exemplos: l_div_r <= l / r; l_rem_r <= l REM r;			



Operadores multiplicação e divisão com tipos físicos

Operadores	Operando L	Operando R	Retorna
*	qualquer tipo physical	integer	mesmo tipo de L
	qualquer tipo physical	real	mesmo tipo de L
	integer	qualquer tipo physical	mesmo tipo de R
	real	qualquer tipo physical	mesmo tipo de R
/	qualquer tipo physical	integer	mesmo tipo de L
	qualquer tipo physical	real	mesmo tipo de L
	qualquer tipo physical	mesmo tipo	integer

Operadores da classe diversos

Operadores	Operando L	Operando R	Retorna
abs	qualquer tipo numérico	-	mesmo tipo
**	qualquer tipo integer	integer	mesmo tipo de L
	qualquer tipo real	integer	mesmo tipo de L
Exemplos: sinal_a <= ABS(-1.7); var_b := a **2;			

Exemplo: atribuição de valores: “INTEGER” e ”REAL”

```
 1 ENTITY int_real IS
 2   PORT (ci1,ci2     : OUT INTEGER RANGE 0    TO 31;
 3         ci3,ci4     : OUT INTEGER RANGE 31   DOWNTO 0;
 4         ci5,ci6,ci7 : OUT INTEGER RANGE -15  TO 15;
 5         cr1,cr2     : OUT REAL    RANGE 0.0  TO 31.0;
 6         cr3,cr4     : OUT REAL    RANGE 31.0 DOWNTO 0.0);
 7 END int_real;
 8
 9 ARCHITECTURE teste OF int_real IS
10   CONSTANT i1 : INTEGER  :=  11;                  -- valor 11, base 10
11   CONSTANT i2 : INTEGER  :=  10#11#;              -- valor 11, base 10
12   CONSTANT i3 : INTEGER  :=  2#01011#;            -- valor 11, base 2
13   CONSTANT i4 : INTEGER  :=  2#01_01_1#;          -- valor 11, base 2
14   CONSTANT i5 : INTEGER  :=  5#21#;               -- valor 11, base 5
15   CONSTANT i6 : NATURAL  :=  8#13#;               -- valor 11, base 8
16   CONSTANT i7 : POSITIVE :=  16#B#;               -- valor 11, base 16
17
18   CONSTANT r1 : REAL     :=  11.0;                -- valor 11, base 10
19   CONSTANT r2 : REAL     :=  1.1E01;              -- valor 11, base 10 formato nn.nExx
20   CONSTANT r3 : REAL     :=  2#01011.0#;          -- valor 11, base 2
21   CONSTANT r4 : REAL     :=  8#1.3#E01;           -- valor 11, base 8  formato nn.nExx
22   CONSTANT r5 : REAL     :=  16#B.0#;             -- valor 11, base 16
23 BEGIN
24   ci1 <= i1; ci2 <= i2; ci3 <= i3; ci4 <= i4; ci5 <= i5; ci6 <= i6; ci7 <= i7;
25   cr1 <= r1; cr2 <= r2; cr3 <= r3; cr4 <= r5;
26 END teste;
```

Exemplo: atribuição de valores: vetores

```
 1 ENTITY std_a IS
 2   PORT (s1, s2, s3, s4, s5  : OUT BIT_VECTOR (4 DOWNTO 0));
 3 END std_a;
 4
 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 (011), parte do vetor com concatenacao
16 END teste;
```

Exemplo: atribuição de valores: vetores

```
 1 ENTITY std_a1 IS
 2  PORT(s2, s3, s4, s5    : OUT BIT_VECTOR(4 DOWNTO 0));
 3 END std_a1;
 4
 5 ARCHITECTURE teste OF std_a1 IS
 6   CONSTANT zero : BIT := '0';
 7   CONSTANT um   : BIT := '1';
 8 BEGIN
 9   s2 <= ('0','0','0','1','0');                    -- valor 00010, agregado notacao posicional
10   s3 <= (1=>'1', 0=>'1', OTHERS=>'0');            -- valor 00011, agregado associacao por nomes
11   s4 <= (zero, '0', um OR '0', '0', '0');         -- valor 00100, agregado com operacoes
12   s5 <= (4 DOWNTO 3 =>'0', 1=>'0', OTHERS=>'1'); -- valor 00101, agregado faixa discreta
13 END teste;
```

Exemplo: operadores lógicos

```
 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 parenteses
12 -- x4 <= a AND b  OR  c AND d;   -- Errado: expressao ambigua, operadores com mesma precedencia
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 parenteses
15    x5 <=(a NAND b) NAND c;       -- Certo:  operador com negacao entre parenteses
16 END exemplo;
```

Exemplo: operadores da classe adição

```
 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;
```

Exemplo: operações com STRING

```
 1 ENTITY lista1 IS
 2   PORT (c, d   : OUT STRING(1 TO 9));
 3 END lista1;
 4
 5 ARCHITECTURE teste OF lista1 IS
 6    SIGNAL x : STRING(1 TO 3) := "Alo";
 7    SIGNAL y : STRING(1 TO 5) := "mundo";
 8 BEGIN
 9   c <= x & " " & y;
10   d <= x(1 TO 2) & "o " & y;
11 END teste;
```