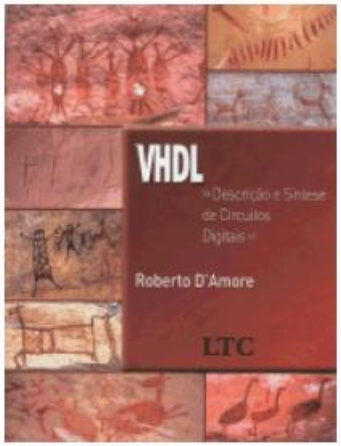
SEL5752/SEL0632 – Linguagens de Descrição de Hardware
Aula 05 – Atraso, Alias e Atributos

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**

Atraso, pseudônimos e atributos

Tópicos

- **Atraso**
- **Pseudônimos**
- **Atributos**
  - Atributos pré-definidos relativos a sinais
  - Atributos pré-definidos relativos a vetores
  - Atributos pré-definidos relativos a tipos

Atraso

- **Modelagem do atraso**: 2 tipos
  - com inércia (*inertial*)
  - com transporte (*transport*)

- **Modelagem com inércia**: cláusula AFTER
  - similar a operação de uma porta lógica
  - pulso menor que o atraso: não é propagado

```
sinal_a <= valor_x AFTER tempo_1;
```

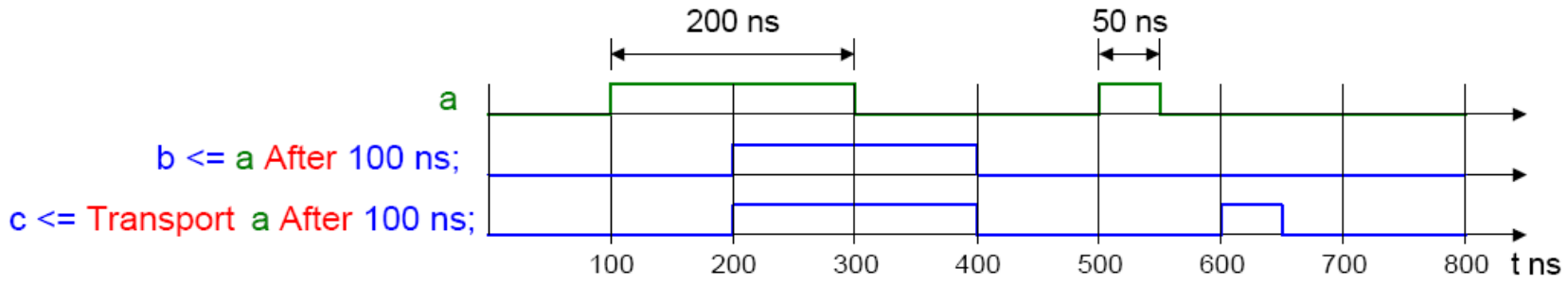
- **Modelagem com transporte**: cláusulas TRANSPORT e AFTER
  - pulso é sempre propagado

```
sinal_d <= TRANSPORT valor_x AFTER tempo_1;
```

Exemplo - Comparação da modelagem com inércia e transporte

- **b**: pulso com duração menor que 100 ns não é propagado
- **c**: pulso sempre propagado

```
 1 ENTITY c04_atr1 IS
 2   PORT (a     :  IN  BIT;
 3         b, c :  OUT BIT);
 4 END c04_atr1;
 5
 6 ARCHITECTURE teste OF c04_atr1 IS
 7  BEGIN
 8    b <= a AFTER 100 ns;
 9    c <= TRANSPORT a AFTER 100 ns;
10 END teste;
```



Exemplo cláusulas com inércia e transporte

- **Formato geral das cláusulas**

```
sinal_a <= valor_1 AFTER tempo_1;
sinal_b <= valor_1, valor_2 AFTER tempo_2, valor_3 AFTER tempo_3;
sinal_c <= expressao_1 AFTER tempo_1, expressao_2 AFTER tempo_2;

sinal_d <= TRANSPORT valor_1 AFTER tempo_1;
sinal_e <= TRANSPORT valor_1, valor_2 AFTER tempo_2, valor_3 AFTER tempo_3;
sinal_f <= TRANSPORT expressao_1 AFTER tempo_1;
```

- **Ferramentas de síntese**
  - Atraso não é sintetizado:
    - inexistência de circuitos para implementar atraso com precisão

- **Cláusulas AFTER e TRANSPORT são ignoradas**

Outra aplicação da cláusula AFTER

- **Gerar formas de onda**
  - estímulos teste interno (própria entidade)
  - entidade especial para teste de uma outra entidade

- **Exemplo:** geração de uma forma de onda aleatória

```
sinal <= '0', '1' AFTER 10 ns,'0' AFTER 20 ns, '1' AFTER 50 ns,'0' AFTER 70 ns ;
```

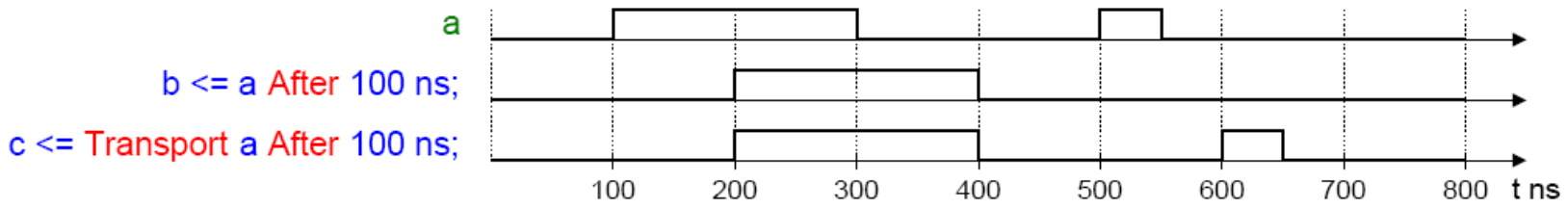
- **Exemplo**: geração de um sinal periódico

```
x0 <= NOT x0 AFTER 100 ns;
```

Exemplo: Geração de formas de onda

- **Descrição com estímulo interno para teste**

```
 1 ENTITY c04_atr2 IS
 2   PORT (a     :  BUFFER BIT;
 3         b, c :  OUT BIT);
 4 END c04_atr2;
 5
 6 ARCHITECTURE teste OF c04_atr2 IS
 7  BEGIN
 8   a <='0','1' AFTER 100 ns,'0' AFTER 300 ns,'1' AFTER 500 ns,'0' AFTER 550 ns;
 9   b <= a AFTER 100 ns;
10   c <= TRANSPORT a AFTER 100 ns;
11 END teste;
```



Pseudônimos Versão VHDL-1987

- **Pseudônimo** - ALIAS
- **Nome alternativo para um objeto das classes**:
  - constante, sinal variável
- **Pseudônimo pode ser**:
  - todo um objeto ou uma parte dele
- **Objetivo**:
  - facilitar o acesso ao objeto
  - melhorar a leitura do código
- **Um pseudônimo**:
  - não é um novo objeto,
  - somente uma designação alternativa para o objeto (parte ou todo)
- **Pseudônimos não são válidos para vetores multidimensionais**

Pseudônimos Versão VHDL-1987 exemplo

- **2 nomes alternativos propostos para o objeto dado**
- **dado**: vetor de 8 bits
- **dado_alto** : novo nome para 4 bits mais significativos de dado
- **dado_baixo** : novo nome para 4 bits restantes.

```
 1 ENTITY alias_a IS
 2   PORT(h, l   : IN  BIT_VECTOR(3 DOWNTO 0);
 3        h_l    : OUT BIT_VECTOR(7 DOWNTO 0));
 4 END alias_a;
 5
 6 ARCHITECTURE teste OF alias_a IS
 7   SIGNAL dado       : BIT_VECTOR(7 DOWNTO 0);
 8   ALIAS  dado_alto  : BIT_VECTOR(3 DOWNTO 0) IS dado(7 DOWNTO 4);
 9   ALIAS  dado_baixo : BIT_VECTOR(3 DOWNTO 0) IS dado(3 DOWNTO 0);
10 BEGIN
11   dado_alto  <= NOT h;              -- equivalente a: dado(7 DOWNTO 4) <= NOT h;
12   dado_baixo <= NOT l;              -- equivalente a: dado(3 DOWNTO 0) <= NOT l;
13   h_l <= dado_alto & dado_baixo;
14 END teste;
```

Pseudônimos Versão VHDL-1993

- **O escopo de um pseudônimo é ampliado**:
  - um nome alternativo para um item nomeado
- **Item objeto**, (*object aliasis*):
  - corresponde ao conceito da versão VHDL-87:
  - objetos das classes: constante variável sinal arquivo
- **Item não objeto**, (*nonobject aliasis*):
  - nome de subprogramas,
  - nome de tipos,
  - nome de atributos,
  - pacotes, etc.
- **Pseudônimos não são permitido nos casos:**
  - rótulos
  - parâmetros de declarações LOOP GENERATE

Pseudônimos Versão VHDL-1993 exemplo em itens não objetos

- **Designações alternativas para dois subprogramas**: logica_funcao logica_proced
- **Pseudônimos de subprogramas**:
  - necessário identificar os tipos na ordem como eles foram definidos entre [ ]
- **A palavra reservada RETURN**: necessária somente para funções

```
 7 ARCHITECTURE teste OF alias_b IS
 8 FUNCTION logica_funcao (a, b : BIT_VECTOR) RETURN BIT_VECTOR IS
 9   BEGIN
10     RETURN a AND NOT b;
11 END logica_funcao;
12
13 PROCEDURE logica_proced(a, b : IN BIT_VECTOR; SIGNAL d, e : OUT BIT_VECTOR) IS
14   BEGIN
15     d <= a  OR NOT b;
16     e <= a XOR NOT b;
17  END logica_proced;
18
19 -- identificador    item       [          assinatura                        ];
20 ALIAS lg_func IS logica_funcao [BIT_VECTOR, BIT_VECTOR  RETURN BIT_VECTOR];
21 ALIAS lg_proc IS logica_proced [BIT_VECTOR, BIT_VECTOR, BIT_VECTOR, BIT_VECTOR];
22
23 BEGIN
24  s <= lg_func(x, y);
25  lg_proc(x, y, t, u);
26 END teste;
```

Atributos

- **Informações adicionais associadas a:**
  - tipos, objetos, unidades de projeto - (entre outros)

- **Exemplo**:
  - um objeto: (num dado instante de tempo)
    - pode conter um único valor
    - pode possuir vários atributos
  - o atributo:
    - pode ou não estar relacionado com o valor do objeto

- **Exemplos de atributos**:
  - um valor
  - um tipo
  - uma constante
  - um sinal
  - os limites de um vetor: (0 TO 3) (7 DOWNTO 0)

Atributos

- **Um atributo pode ser**:
  - pré definido
  - ou definido pelo usuário
- **Formato para referenciar um atributo de um item**:

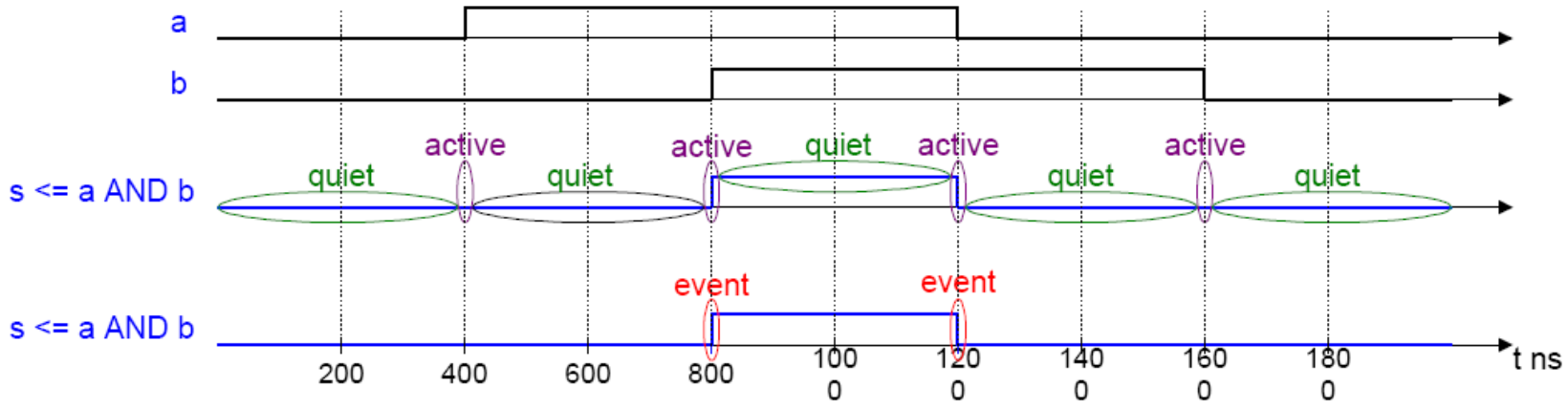
```
prefixo'nome_atributo
```

- **prefixo**: corresponde ao item
- **apóstrofo**: separa prefixo do nome_atributo
- **nome_atributo**: o atributo desejado do item
- **Exemplo**
  - item referenciado: sinal s

```
s'QUIET       -- referenciado o atributo QUIET  do sinal s
s'STABLE      -- referenciado o atributo STABLE do sinal s
```

Definição de termos relativos a sinais (útil para entender alguns atributos)

- **active**: novo valor atribuído ao sinal (o valor pode ser o mesmo)
- **quiet**: condição oposta a active
- **event**: mudança do valor de um sinal
- **Exemplo**:



Atributos pré-definidos relativos a sinais → resultam novos sinais

- **Relação parcial**:

atributo	descrição
s'DELAYED [t]	novo sinal equivalente a **s**, atrasado **t** unidades de tempo
s'STABLE [t]	novo sinal tipo boolean - verdadeiro se não ocorreu uma troca de valor por um período de tempo **t**, - falso caso contrario
s'QUIET [t]	novo sinal tipo boolean - verdadeiro se nenhum valor foi atribuído por um período de tempo **t** - falso caso contrario
s'TRANSACTION	novo sinal tipo bit - valor complementado com relação ao anterior a cada atribuição de valor
Nota: o parâmetro [t] corresponde ao valor de tempo, se omitido subentende-se "t=0 s"	

- **A seguir**: exemplos relacionado cada atributo

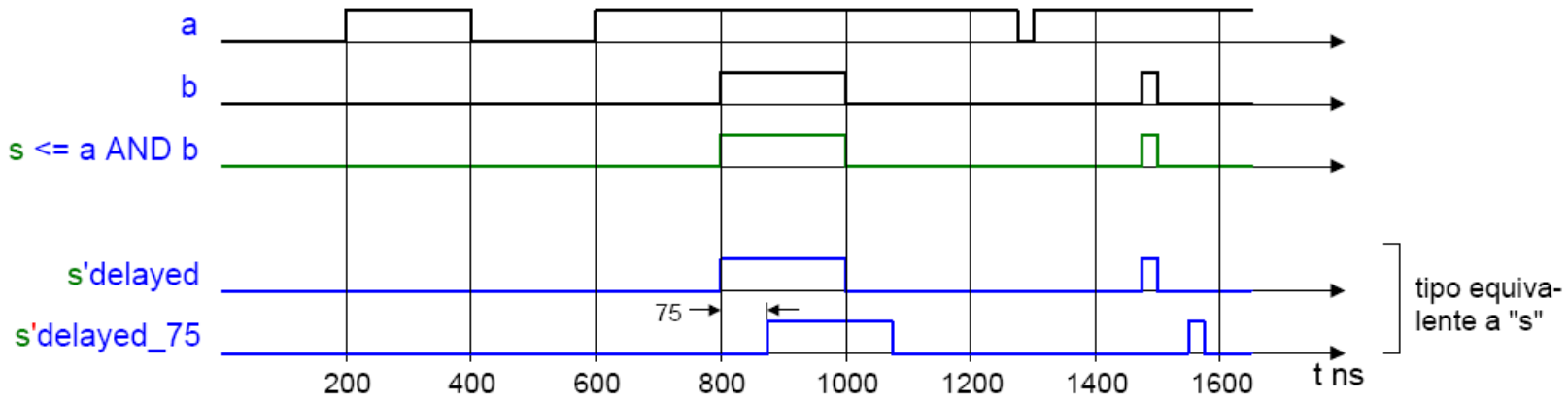
Atributos pré-definidos relativos a sinais → resultam novos sinais

- **Exemplo**: descrição para teste

```
 1 ENTITY c04_at0 IS
 2   PORT (a, b                              : IN  BIT;
 3         s                                 : BUFFER BIT;
 4         s_delayed, s_delayed_75           : OUT BIT;
 5         s_stable, s_stable_50             : OUT BOOLEAN;
 6         s_quiet, s_quiet_50               : OUT BOOLEAN;
 7         s_transaction                     : OUT BIT);
 8 END c04_at0;
 9
10 ARCHITECTURE teste OF c04_at0 IS
11 BEGIN
12   s <= a AND b;
13   s_delayed     <= s'DELAYED;    s_delayed_75 <= s'DELAYED(75 ns);
14   s_stable      <= s'STABLE;     s_stable_50  <= s'STABLE(50 ns);
15   s_quiet       <= s'QUIET;      s_quiet_50   <= s'QUIET(50 ns);
16   s_transaction <= s'TRANSACTION;
17 END teste;
```

Atributos pré-definidos relativos a sinais → resultam novos sinais

s'DELAYED [t] novo sinal equivalente a **s**, atrasado **t** unidades de tempo



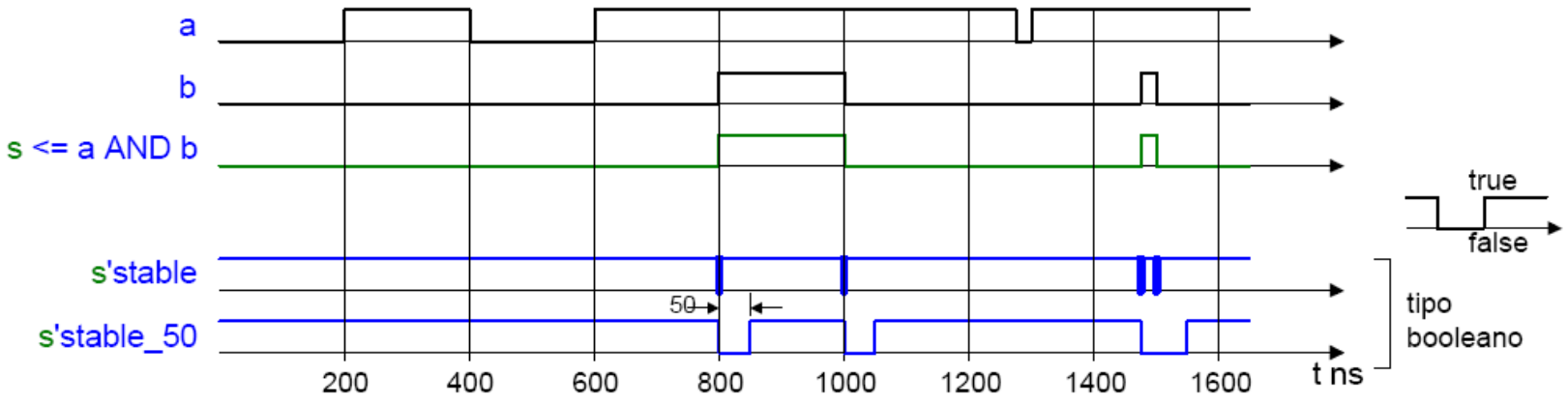
Nota: o parâmetro [t] corresponde ao valor de tempo, se omitido subentende-se "t=0 s"

Atributos pré-definidos relativos a sinais → resultam novos sinais

s'STABLE [t] novo sinal tipo boolean

-verdadeiro se não ocorreu uma troca de valor por um período de tempo **t**,

- falso caso contrario

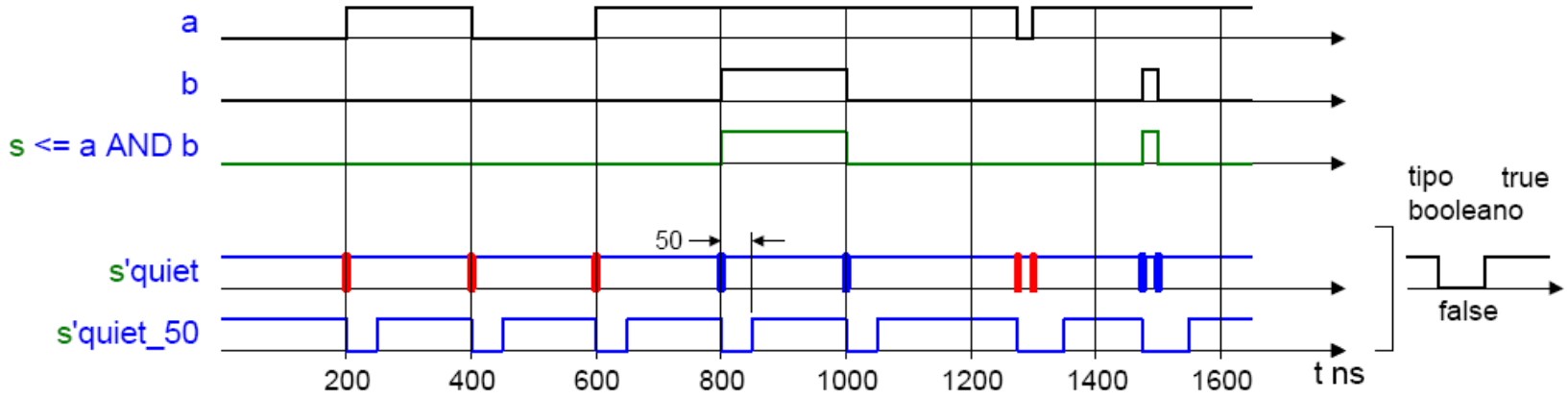


Nota: o parâmetro [t] corresponde ao valor de tempo, se omitido subentende-se "t=0 s"

Atributos pré-definidos relativos a sinais → resultam novos sinais

s'QUIET [t] novo sinal tipo boolean

- verdadeiro se nenhum valor foi atribuído por um período de tempo **t**
- falso caso contrario



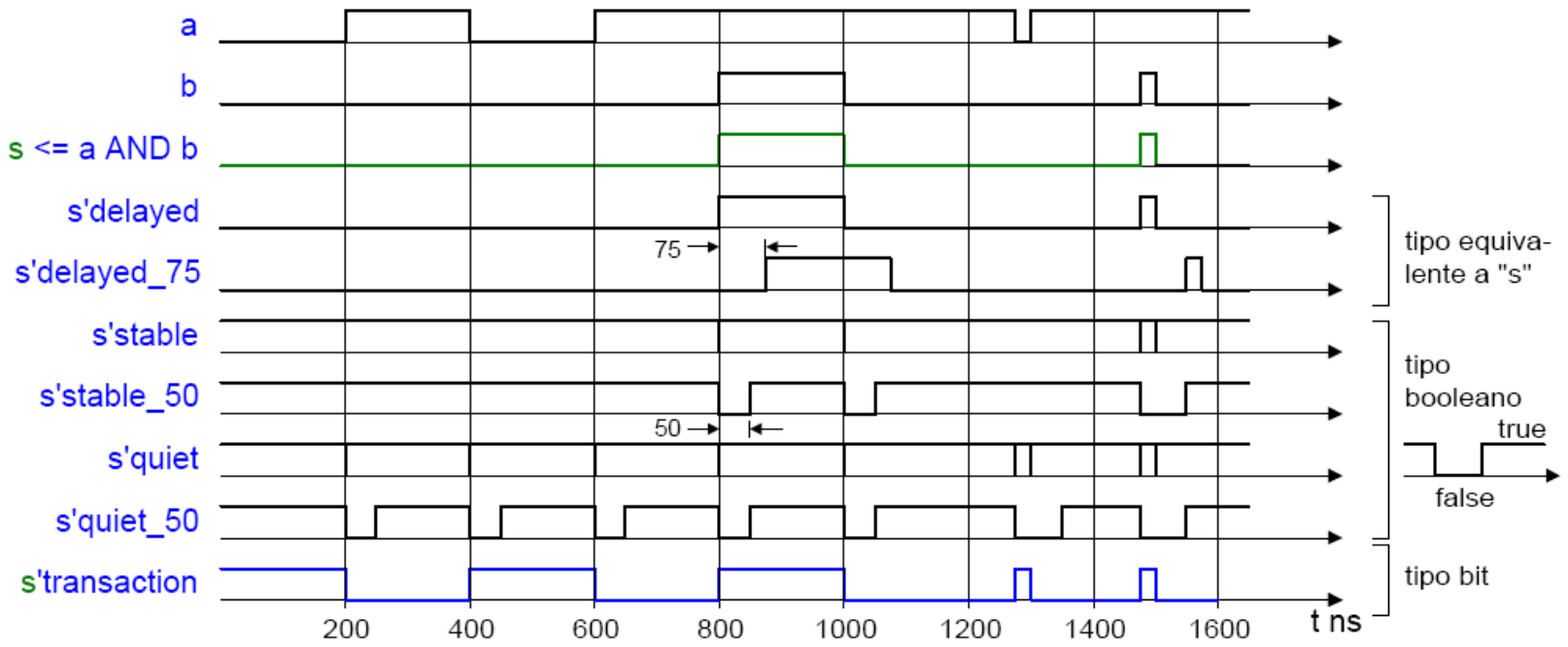
Nota: o parâmetro [t] corresponde ao valor de tempo, se omitido subentende-se "t=0 s"

Note: em vermelho somente atribuição de valor, não ocorreu uma troca no valor

Atributos pré-definidos relativos a sinais → resultam novos sinais

s'TRANSACTION novo sinal tipo bit

- valor complementado com relação ao anterior a cada atribuição de valor (i.e. ocorreu ou não a troca de valor)



Atributos pré-definidos relativos a sinais → não resultam novos sinais

- **Relação parcial**:

atributo	descrição
s'EVENT	- verdadeiro: ocorreu uma troca de valor no ciclo corrente da simulação - falso: caso contrario
s'ACTIVE	- verdadeiro: foi atribuído um valor durante o ciclo corrente da simulação - falso: caso contrario
s'LAST_VALUE	valor do sinal antes do último evento
s'LAST_EVENT	tempo decorrido desde a última troca de valor do sinal

- **A seguir**: exemplos relacionado cada atributo

Atributos pré-definidos relativos a sinais → não resultam novos sinais

- **Exemplo**: descrição para teste

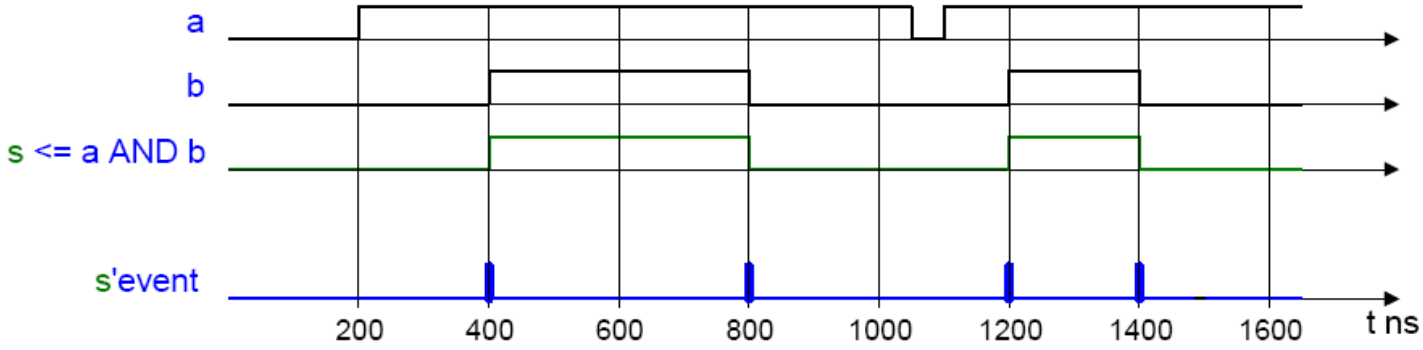
```
 1 ENTITY c04_at1 IS
 2   PORT (a, b               : IN  BIT;
 3         s                  : BUFFER BIT;
 4         s_event, s_active  : BUFFER BOOLEAN;
 5         s_last_event       : BUFFER TIME;
 6         s_last_value       : BUFFER BIT);
 7 END c04_at1;
 8
 9 ARCHITECTURE teste OF c04_at1 IS
10   SIGNAL a_d, b_d  : BIT;
11 BEGIN
12   s <= a AND b;
13   a_d <= a; b_d <=b; --ativos mesmo ciclo
14                      --delta de s
15
16   abc: PROCESS(a_d, b_d, s_event)
17   BEGIN
18     IF s'EVENT THEN s_event <= TRUE;
19     ELSE            s_event <= FALSE;
20     END IF;
21   END PROCESS;
22   def: PROCESS (a_d, b_d, s_active)
23   BEGIN
24     IF s'ACTIVE THEN s_active <= TRUE;
25     ELSE             s_active <= FALSE;
26     END IF;
27   END PROCESS;
28
29   ghi: PROCESS(s, s_last_value)
30   BEGIN
31     IF s'LAST_VALUE = '1' THEN s_last_value <= '1';
32     ELSE                       s_last_value <= '0';
33     END IF;
34   END PROCESS;
35
36   jkl: PROCESS
37   BEGIN
38     WAIT FOR 150 ns;
39     s_last_event <= s'LAST_EVENT;
40   END PROCESS;
41 END teste;
```

Atributos pré-definidos relativos a sinais → não resultam novos sinais

s'EVENT - verdadeiro:
ocorreu uma troca de valor no ciclo corrente da simulação

- falso:
caso contrario

```
16   abc: PROCESS(a_d, b_d, s_event)
17   BEGIN
18     IF s'EVENT THEN s_event <= TRUE;
19     ELSE            s_event <= FALSE;
20     END IF;
21   END PROCESS;
```

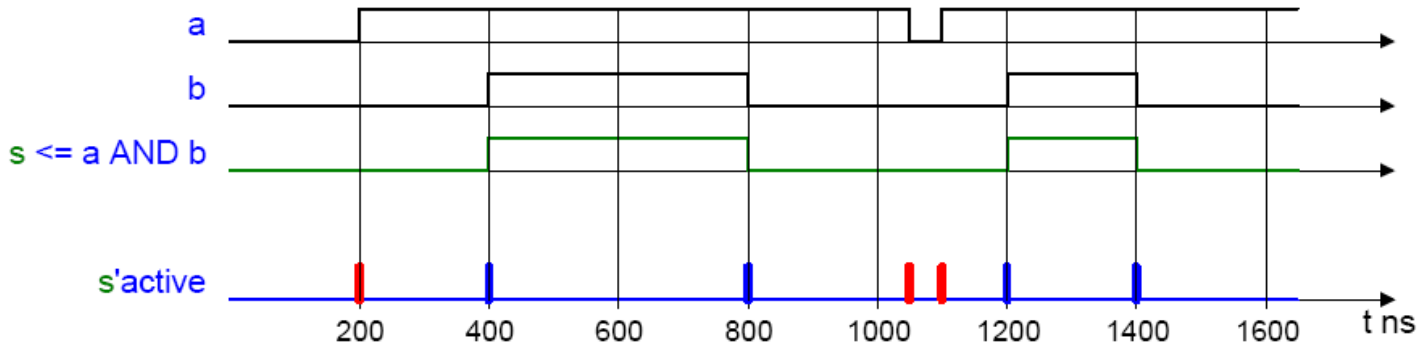


Atributos pré-definidos relativos a sinais → não resultam novos sinais

s'ACTIVE

- verdadeiro:
  foi atribuído um valor durante o ciclo corrente da simulação
- falso:
  caso contrario

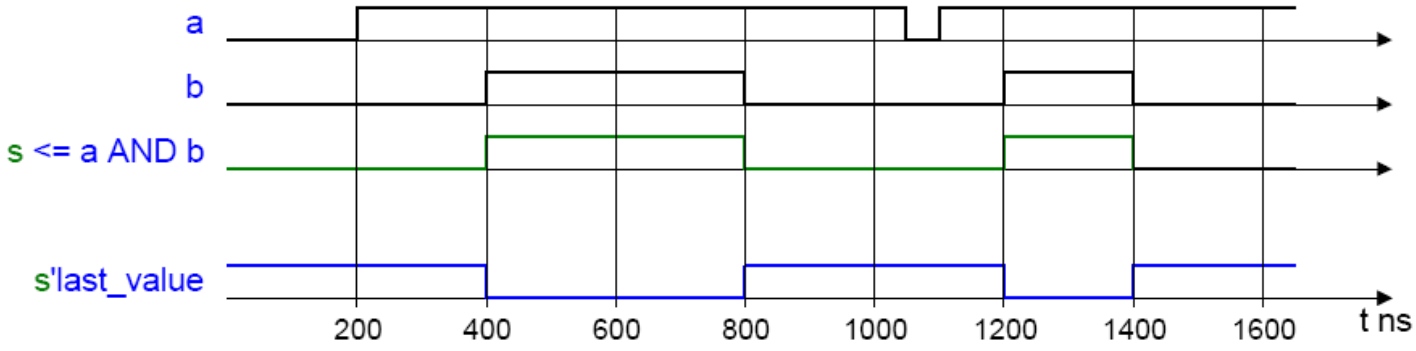
```
22   def: PROCESS (a_d, b_d, s_active)
23   BEGIN
24     IF s'ACTIVE THEN s_active <= TRUE;
25     ELSE             s_active <= FALSE;
26     END IF;
27   END PROCESS;
```



Note: em vermelho somente atribuição de valor, não ocorreu uma troca no valor

Atributos pré-definidos relativos a sinais → não resultam novos sinais

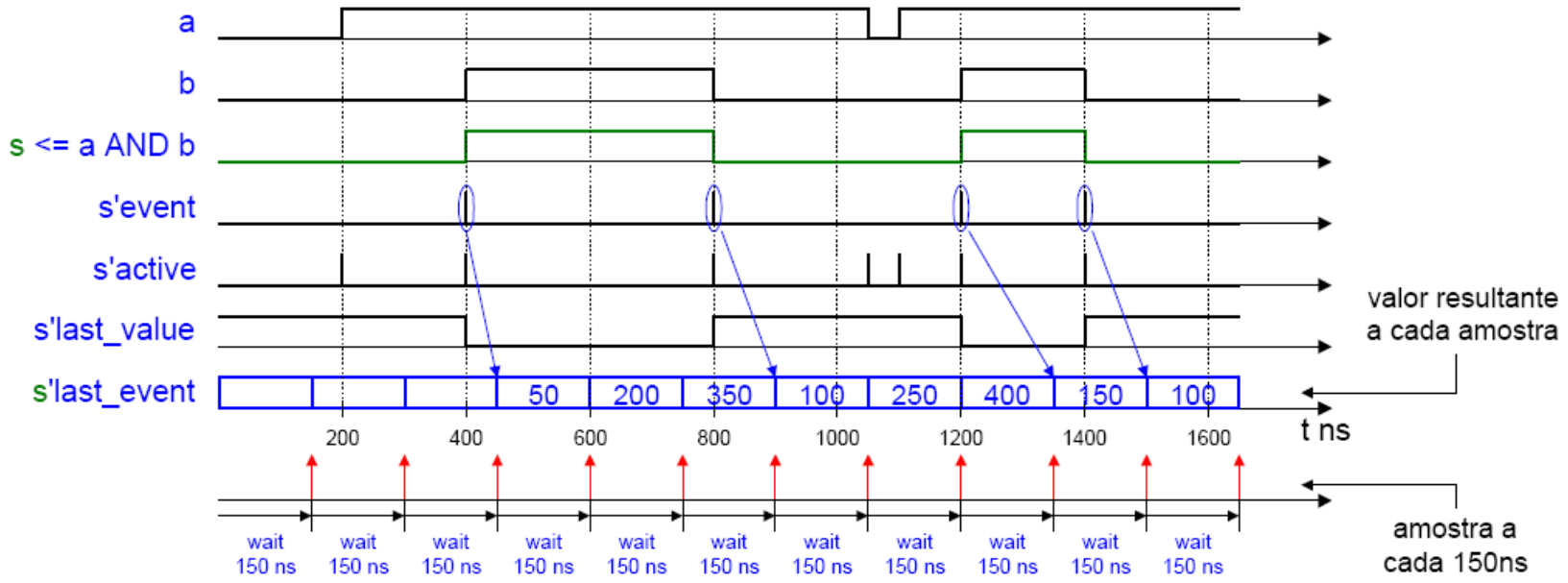
s'LAST_VALUE valor do sinal antes do último evento




```
29   ghi: PROCESS(s, s_last_value)
30   BEGIN
31     IF s'LAST_VALUE = '1' THEN s_last_value <= '1';
32     ELSE                       s_last_value <= '0';
33     END IF;
34   END PROCESS;
```

Atributos pré-definidos relativos a sinais → não resultam novos sinais

s'LAST_EVENT tempo decorrido desde a última troca de valor do sinal




```
36   jkl: PROCESS
37   BEGIN
38     WAIT FOR 150 ns;
39     s_last_event <= s'LAST_EVENT;     -- amostra os eventos a cada 150 ns
40   END PROCESS;
```

Atributos pré-definidos relativos a vetores

- **Determinam as características de vetores**:
  - número de elementos, limites e faixas
- **Parâmetro [n] é opcional**:
  - indica a dimensão do vetor
  - quando omitido subentende-se o valor 1

atributo	descrição
a'HIGH[n]	limite superior da dimensão [n] no vetor
a'LOW[n]	limite inferior da dimensão [n] no vetor
a'RIGHT[n]	limite direito da dimensão [n] no vetor
a'LEFT[n]	limite esquerdo da dimensão [n] no vetor
a'LENGTH[n]	número de elementos da dimensão [n] no vetor
a'RANGE[n]	faixa do vetor da dimensão [n]: (x DOWNTO y) ou (y TO x)
a'REVERSE_RANGE[n]	faixa oposta a obtida com o atributo RANGE

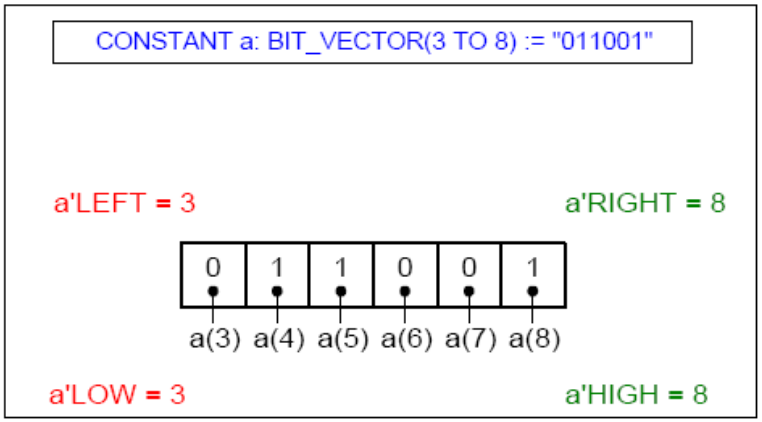
Atributos pré-definidos relativos a vetores

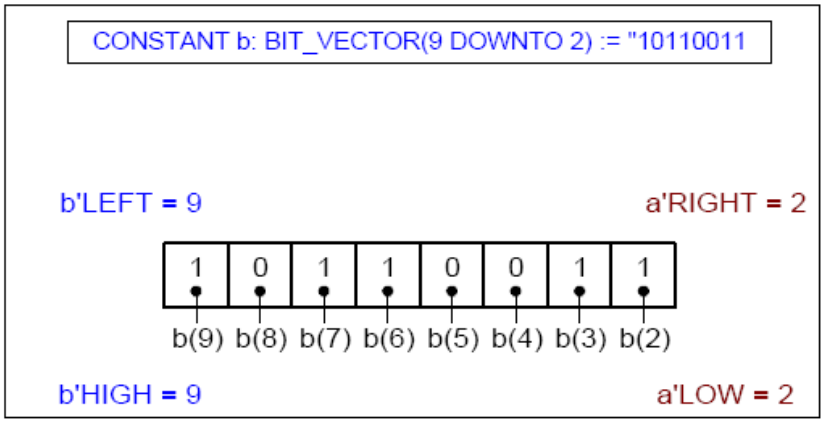
- **Exemplo**: descrição para teste
  - vetor de uma dimensão

```
 1 ENTITY c04_at2 IS
 2   PORT (a          : IN  BIT_VECTOR(3 TO     8);
 3         b          : IN  BIT_VECTOR(9 DOWNTO 2);
 4         a_right, b_right, a_left, b_left : OUT INTEGER;
 5         a_high,  b_high,  a_low,  b_low  : OUT INTEGER;
 6         c_length, d_right                : OUT INTEGER);
 7 END c04_at2;
 8
 9 ARCHITECTURE teste OF c04_at2 IS
10   SIGNAL c : BIT_VECTOR(a'RANGE);             -- a'RANGE = 3 TO 8
11   SIGNAL d : BIT_VECTOR(a'REVERSE_RANGE);     -- a'REVERSE_RANGE = 8 DOWNTO 3
12 BEGIN
13   a_right  <= a'RIGHT;   b_right <= b'RIGHT; -- a'RIGHT =8  b_right =2
14   a_left   <= a'LEFT;    b_left  <= b'LEFT;  -- a'LEFT =3   b_left =9
15   a_high   <= a'HIGH;    b_high  <= b'HIGH;  -- a'HIGH =8   b_high =9
16   a_low    <= a'LOW;     b_low   <= b'LOW;   -- a'LOW =3    b_low=2
17   c_length <= c'LENGTH;                      -- c'LENGTH =6
18   d_right  <= d'RIGHT;                       -- d'RIGHT =3
19 END teste;
```

Atributos pré-definidos relativos a vetores

- **para designações crescentes**: a'left = a'low a'right = a'high
- **para designações decrescentes**: b'left = b'high b'right = b'low





Atributos pré-definidos relativos a vetores

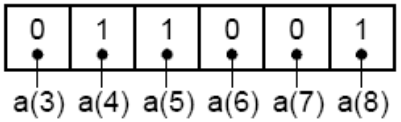
a'LENGTH	número de elementos no vetor
a'RANGE	faixa do vetor: (x DOWNTO y) ou (y TO x)
a'REVERSE_RANGE	faixa oposta a obtida com o atributo RANGE

CONSTANT a: BIT_VECTOR(3 TO 8) := "011001"

a'LENGHT = 6

a'RANGE = 3 TO 8

a'REVERSE_RANGE = 8 DOWNTO 3

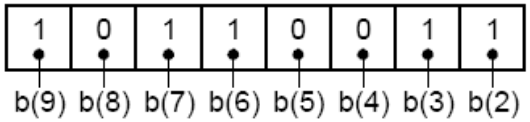


CONSTANT b: BIT_VECTOR(9 DOWNTO 2) := "10110011

b'LENGHT = 8

b'RANGE = 9 DOWNTO 2

b'REVERSE_RANGE = 2 TO 9



Atributos pré-definidos relativos a tipos

- **Identificam os elementos de tipos e sub-tipos escalares**
- **Observar**: alguns atributos têm a mesma designação de atributos de vetor

atributo	descrição
t'HIGH	elemento de posição mais elevada no tipo ou sub-tipo
t'LOW	elemento de posição mais inferior no tipo ou sub-tipo
t'SUCC(e)	elemento que sucede o elemento e no tipo ou sub-tipo
t'PRED(e)	elemento que precede o elemento e no tipo ou sub-tipo
t'RIGHT	elemento de posição mais a direita no tipo ou sub-tipo
t'LEFT	elemento de posição mais a esquerda no tipo ou sub-tipo
t'RIGHTOF(e)	elemento a direta do elemento e no tipo ou sub-tipo
t'LEFTOF(e)	elemento a esquerda do elemento e no tipo ou sub-tipo
t'VAL(p)	elemento de posição p no tipo ou sub-tipo
t'POS(e)	posição do elemento e no tipo ou sub-tipo
Nota: e = elemento, p = posição referenciada	

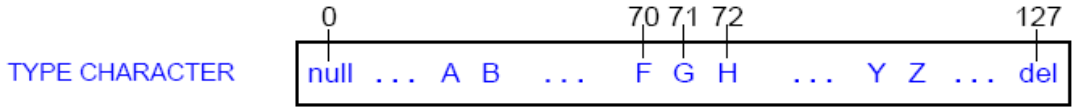
Atributos pré-definidos relativos a tipos

- **Exemplo**:
  - definidos dois sub-tipos do tipo CHARACTER:
    - as conjunto de A a Z ascendente (linha 13)
    - ds conjunto de Z a A descendente (linha 14)
  - possível transferência de valores entre: tipo ⇔ sub-tipo

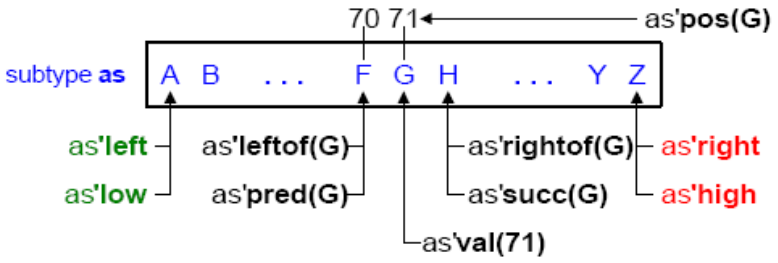
```
13    SUBTYPE as IS CHARACTER RANGE 'A' TO 'Z';
14    SUBTYPE ds IS CHARACTER RANGE 'Z' DOWNTO 'A';
15  BEGIN
17    as_right   <= as'RIGHT;      ds_right   <= ds'RIGHT;
18    as_left    <= as'LEFT;       ds_left    <= ds'LEFT;
19    as_high    <= as'HIGH;       ds_high    <= ds'HIGH;
20    as_low     <= as'LOW;        ds_low     <= ds'LOW;
21
22    as_succ_G    <= as'SUCC('G');     ds_succ_G    <= ds'SUCC('G');
23    as_pred_G    <= as'PRED('G');     ds_pred_G    <= ds'PRED('G');
24    as_rightof_G <= as'RIGHTOF('G'); ds_rightof_G <= ds'RIGHTOF('G');
25    as_leftof_G  <= as'LEFTOF('G');   ds_leftof_G  <= ds'LEFTOF('G');
```

Atributos pré-definidos relativos tipos

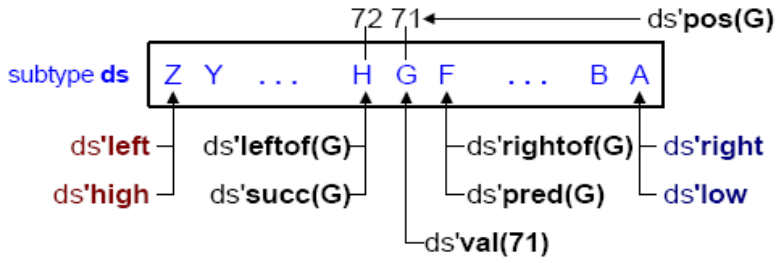
- **Designações crescentes**: as'left = as'low as'right = as'high
- **Designações decrescentes**: ds'left = ds'high ds'right = ds'low



SUBTYPE **as** IS CHARACTER RANGE 'A' TO 'Z';



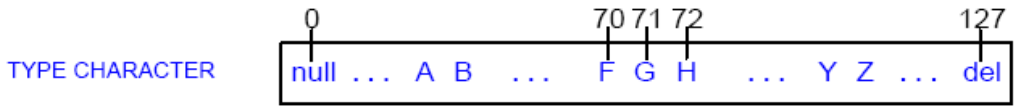
SUBTYPE **ds** IS CHARACTER RANGE 'Z' DOWNTO 'A';



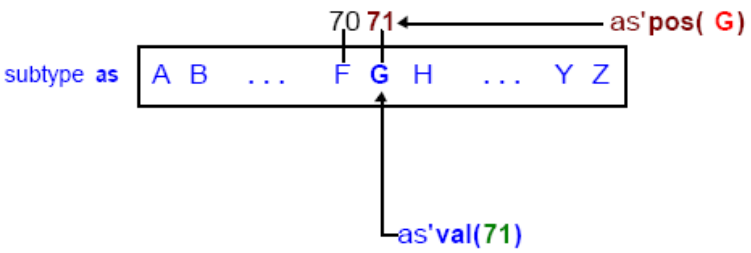
Atributos pré-definidos relativos tipos

t'VAL(p) elemento de posição p no tipo ou sub-tipo

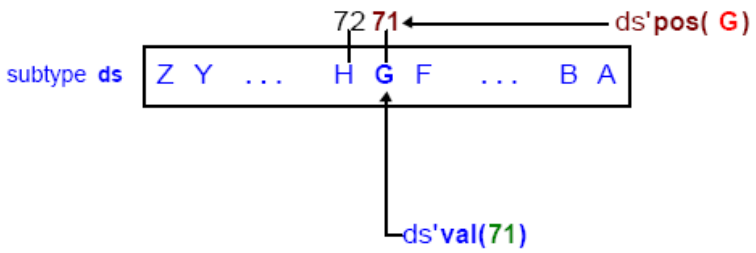
t'POS(e) posição do elemento e no tipo ou sub-tipo



SUBTYPE as IS CHARACTER RANGE 'A' TO 'Z';



SUBTYPE ds IS CHARACTER RANGE 'Z' DOWNTO 'A';



User-Defined Attributes

Attributes can also be user defined. In this case, the attribute first has to be declared, with a type, and then its value can be set on a signal or other object. This value can then be used with the " ' " construct. The following is an example:

```
signal my_vector : bit_vector (4 downto 0) ;
attribute MIDDLE : integer ;
attribute MIDDLE of my_vector : signal is  my_vector'LENGTH/2 ;
....
      my_vector'MIDDLE          -- returns integer 2
```