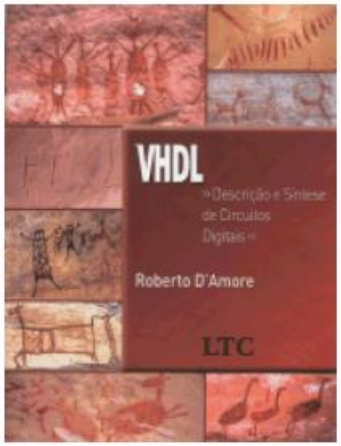
SEL5752/SEL0632 – Linguagens de Descrição de Hardware
Aula 10 – Padrão IEEE 1076.3

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**

Padrão IEEE 1076.3

Tópicos

- Tipos definidos
- Operações definidas
- Exemplos de utilização
- Exemplos de síntese
- Sumário

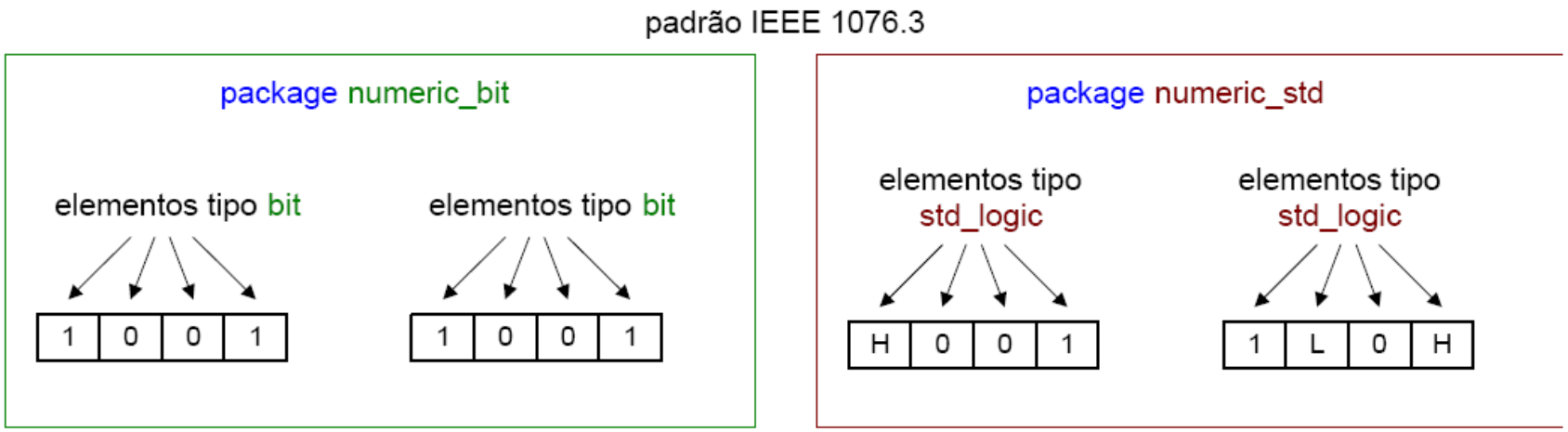
Padrão IEEE 1076.3

Tipos definidos

- **Dois tipos numéricos** → vetores compostos de elementos:
  - tipo bit
  - tipo std_logic

- **Vetores compostos de elementos tipo bit** → pacote numeric_bit

- **Vetores compostos de elementos tipo std_logic** → pacote numeric_std



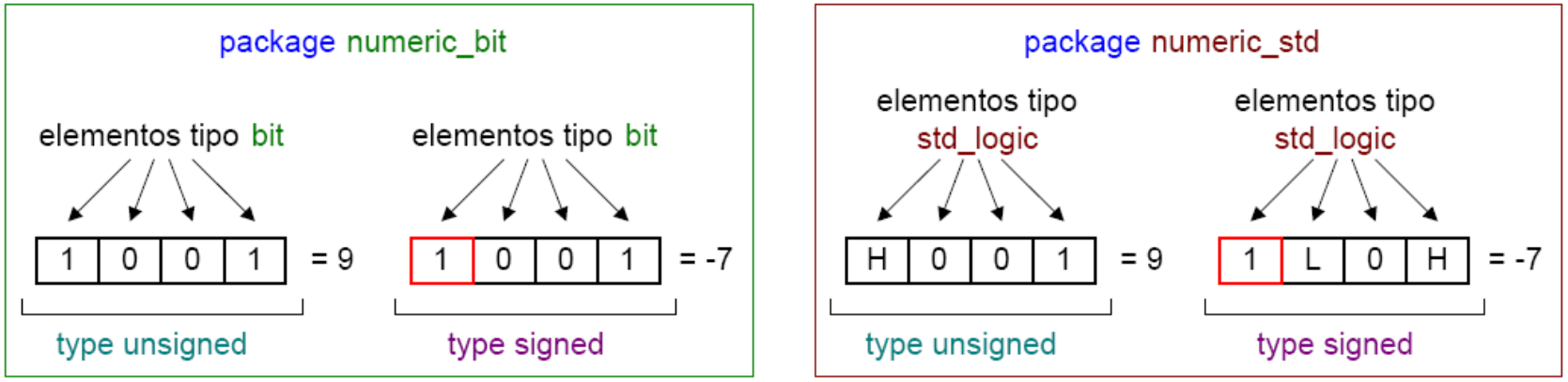
Tipos definidos

- **Tipos numéricos definidos**:
  - unsigned → número sem sinal
  - singed → número com sinal (formato complemento de 2)

- **No tipo signed a posição mais a esquerda representa o sinal**

- **Somente um pacote pode ser empregado por vez**: numeric_bit ou numeric_std
  - os tipos unsigned e singed possuem a mesma designação!

padrão IEEE 1076.3



```
package NUMERIC_BIT is
 constant CopyRightNotice: STRING
      := "Copyright 1995 IEEE. All rights reserved.";

  --============================================================================
  -- Numeric array type definitions
  --============================================================================

  type UNSIGNED is array (NATURAL range <> ) of BIT;
  type SIGNED is array (NATURAL range <> ) of BIT;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package NUMERIC_STD is
 constant CopyRightNotice: STRING
      := "Copyright 1995 IEEE. All rights reserved.";

  --============================================================================
  -- Numeric array type definitions
  --============================================================================

  type UNSIGNED is array (NATURAL range <> ) of STD_LOGIC;
  type SIGNED is array (NATURAL range <> ) of STD_LOGIC;
```

Funções definidas no pacote

- **Mesma designação de operações pré definidas**: + , - , AND etc.
  - funções sobrecarregam funções definidas no pacote padrão

- **Vários tipos são suportados, conforme a operação**
  - unsigned , signed , integer e natural

- **Operações aritméticas 1 operando**:
  - abs -

- **Operações aritméticas 2 operandos**:
  - + - * / mod rem

- **Operações lógicas**:
  - NOT AND OR NAND NOT XOR

```
-- Id: A.3
 function "+" (L, R: UNSIGNED) return UNSIGNED;
 -- Result subtype: UNSIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
 -- Result: Adds two UNSIGNED vectors that may be of different lengths.

 -- Id: A.4
 function "+" (L, R: SIGNED) return SIGNED;
 -- Result subtype: SIGNED(MAX(L'LENGTH, R'LENGTH)-1 downto 0).
 -- Result: Adds two SIGNED vectors that may be of different lengths.

 -- Id: A.5
 function "+" (L: UNSIGNED; R: NATURAL) return UNSIGNED;
 -- Result subtype: UNSIGNED(L'LENGTH-1 downto 0).
 -- Result: Adds an UNSIGNED vector, L, with a non-negative INTEGER, R.

 -- Id: A.6
 function "+" (L: NATURAL; R: UNSIGNED) return UNSIGNED;
 -- Result subtype: UNSIGNED(R'LENGTH-1 downto 0).
 -- Result: Adds a non-negative INTEGER, L, with an UNSIGNED vector, R.

 -- Id: A.7
 function "+" (L: INTEGER; R: SIGNED) return SIGNED;
 -- Result subtype: SIGNED(R'LENGTH-1 downto 0).
 -- Result: Adds an INTEGER, L(may be positive or negative), to a SIGNED
 -- vector, R.

 -- Id: A.8
 function "+" (L: SIGNED; R: INTEGER) return SIGNED;
 -- Result subtype: SIGNED(L'LENGTH-1 downto 0).
 -- Result: Adds a SIGNED vector, L, to an INTEGER, R.
```

```
-- this internal function computes the addition of two SIGNED
-- with input carry
-- * the two arguments are of the same length

function ADD_SIGNED (L, R: SIGNED; C: BIT) return SIGNED is
  constant L_LEFT: INTEGER := L'LENGTH-1;
  alias XL: SIGNED(L_LEFT downto 0) is L;
  alias XR: SIGNED(L_LEFT downto 0) is R;
  variable RESULT: SIGNED(L_LEFT downto 0);
  variable CBIT: BIT := C;
begin
  for I in 0 to L_LEFT loop
    RESULT(I) := CBIT xor XL(I) xor XR(I);
    CBIT := (CBIT and XL(I)) or (CBIT and XR(I)) or (XL(I) and XR(I));
  end loop;
  return RESULT;
end ADD_SIGNED;
```

```
-- Id: A.4
function "+" (L, R: SIGNED) return SIGNED is
  constant L_LEFT: INTEGER := L'LENGTH-1;
  constant R_LEFT: INTEGER := R'LENGTH-1;
  constant SIZE: NATURAL := MAX(L'LENGTH, R'LENGTH);
  variable L01 : SIGNED(SIZE-1 downto 0);
  -- Synthesis directives :
  attribute IS_SIGNED of L:constant is TRUE ;
  attribute IS_SIGNED of R:constant is TRUE ;
  attribute is_signed of L01:variable is TRUE ;
  attribute SYNTHESIS_RETURN of L01:variable is "ADD" ;
begin
  if ((L'LENGTH < 1) or (R'LENGTH < 1)) then return NAS;
  end if;
  return ADD_SIGNED(RESIZE(L, SIZE), RESIZE(R, SIZE), '0');
end "+";
```

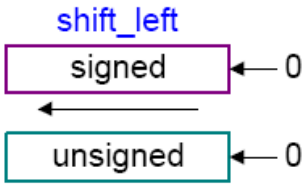
```
constant NAS: SIGNED(0 downto 1) := (others => '0');
```

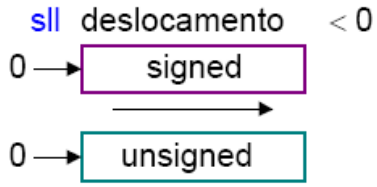
```vhdl
-- Id: A.5
function "+" (L: UNSIGNED; R: NATURAL) return UNSIGNED is
begin
  return L + TO_UNSIGNED(R, L'LENGTH);
end "+";

-- Id: A.6
function "+" (L: NATURAL; R: UNSIGNED) return UNSIGNED is
begin
  return TO_UNSIGNED(L, R'LENGTH) + R;
end "+";

-- Id: A.7
function "+" (L: SIGNED; R: INTEGER) return SIGNED is
begin
  return L + TO_SIGNED(R, L'LENGTH);
end "+";

-- Id: A.8
function "+" (L: INTEGER; R: SIGNED) return SIGNED is
begin
  return TO_SIGNED(L, R'LENGTH) + R;
end "+";
```

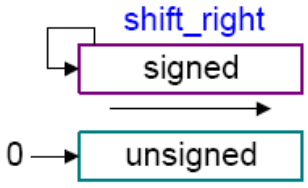
Funções definidas no pacote

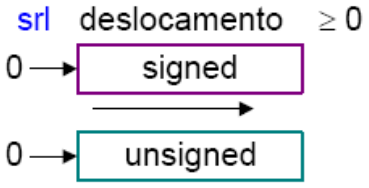
- **Operações relacionais**:
  - `>` `<` `<=` `>=` `=` `/=`

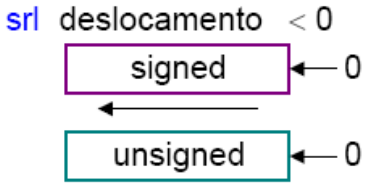
- **Operações de deslocamento**:
  - shift_right shift_left → valor do deslocamento tipo natural
  - srl sll → valor do deslocamento tipo integer (VHDL-1993)





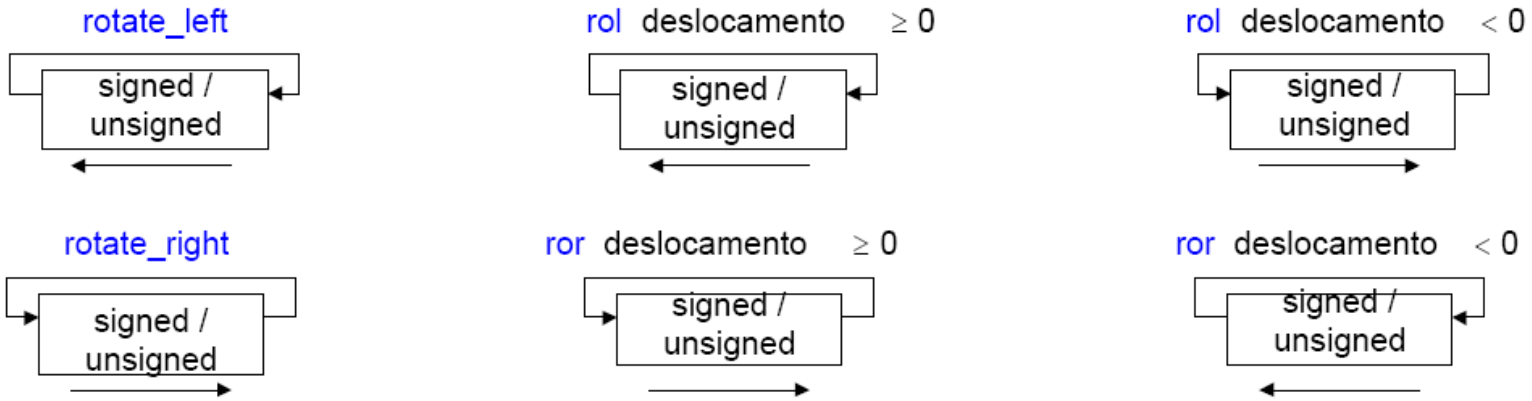






Funções definidas no pacote

- **Operações de rotação**
  - rotate_right rotate_left → valor do deslocamento tipo natural
  - ror rol → valor do deslocamento tipo integer (VHDL-1993)



Funções definidas no pacote

- **Operações de conversão de tipos**
  - to_integer
  - to_unsigned
  - to_signed
  - to_01

```
--============================================================================
-- Conversion Functions
--============================================================================

-- Id: D.1
function TO_INTEGER (ARG: UNSIGNED) return NATURAL;
-- Result subtype: NATURAL. Value cannot be negative since parameter is an
--         UNSIGNED vector.
-- Result: Converts the UNSIGNED vector to an INTEGER.

-- Id: D.2
function TO_INTEGER (ARG: SIGNED) return INTEGER;
-- Result subtype: INTEGER
-- Result: Converts a SIGNED vector to an INTEGER.

-- Id: D.3
function TO_UNSIGNED (ARG, SIZE: NATURAL) return UNSIGNED;
-- Result subtype: UNSIGNED(SIZE-1 downto 0)
-- Result: Converts a non-negative INTEGER to an UNSIGNED vector with
--         the specified size.

-- Id: D.4
function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED;
-- Result subtype: SIGNED(SIZE-1 downto 0)
-- Result: Converts an INTEGER to a SIGNED vector of the specified size.
```

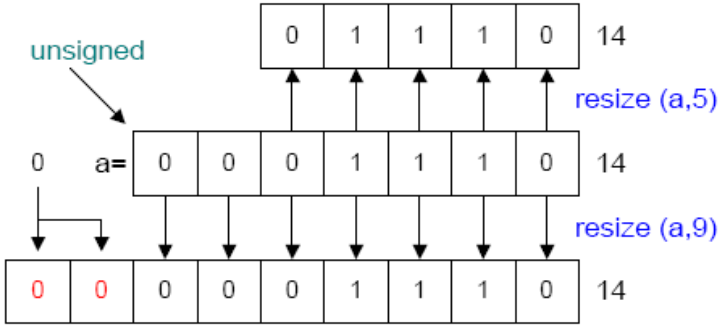
```
-- Id: D.1
 function TO_INTEGER (ARG: UNSIGNED) return NATURAL is
   constant ARG_LEFT: INTEGER := ARG'LENGTH-1;
   alias XARG: UNSIGNED(ARG_LEFT downto 0) is ARG;
   variable RESULT: NATURAL := 0;
   -- Synthesis directives :
   attribute SYNTHESIS_RETURN of RESULT:variable is "FEED_THROUGH" ;
 begin
   if (ARG'LENGTH < 1) then
     assert NO_WARNING
         report "NUMERIC_BIT.TO_INTEGER: null detected, returning 0"
         severity WARNING;
     return 0;
   end if;
   for I in XARG'RANGE loop
     RESULT := RESULT+RESULT;
     if XARG(I) = '1' then
       RESULT := RESULT + 1;
     end if;
   end loop;
   return RESULT;
 end TO_INTEGER;
```

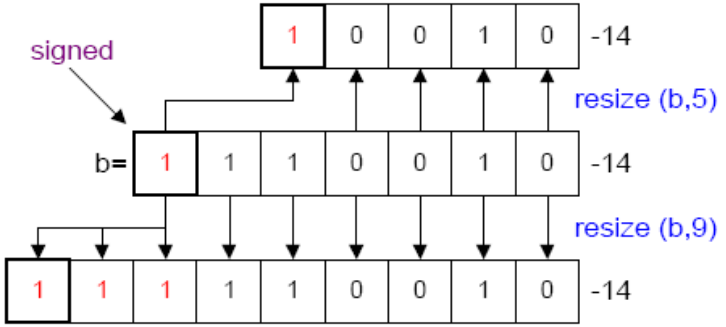
```
-- Id: D.2
 function TO_INTEGER (ARG: SIGNED) return INTEGER is
   -- Synthesis directives :
   variable RESULT : INTEGER ;
   attribute IS_SIGNED of ARG:constant is TRUE ;
   attribute SYNTHESIS_RETURN of RESULT:variable is "FEED_THROUGH";
begin
   if (ARG'LENGTH < 1) then
     assert NO_WARNING
         report "NUMERIC_BIT.TO_INTEGER: null detected, returning 0"
         severity WARNING;
     return 0;
   end if;
   if ARG(ARG'LEFT) = '0' then
     return TO_INTEGER(UNSIGNED(ARG));
   else
     return (- (TO_INTEGER(UNSIGNED(- (ARG + 1)))) -1);
   end if;
 end TO_INTEGER;
```

```
-- Id: D.4
function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL) return SIGNED is
  variable RESULT: SIGNED(SIZE-1 downto 0);
  variable B_VAL: BIT := '0';
  variable I_VAL: INTEGER := ARG;
  -- Synthesis directives :
  attribute SYNTHESIS_RETURN of RESULT:variable is "FEED_THROUGH" ;
begin
  if (SIZE < 1) then return NAS;
  end if;
  if (ARG < 0) then
    B_VAL := '1';
    I_VAL := -(ARG+1);
  end if;
  for I in 0 to RESULT'LEFT loop
    if (I_VAL mod 2) = 0 then
      RESULT(I) := B_VAL;
    else
      RESULT(I) := not B_VAL;
    end if;
    I_VAL := I_VAL/2;
  end loop;
  if ((I_VAL/=0) or (B_VAL/=RESULT(RESULT'LEFT))) then
    assert NO_WARNING
        report "NUMERIC_BIT.TO_SIGNED: vector truncated"
        severity WARNING;
  end if;
  return RESULT;
end TO_SIGNED;
```

Funções definidas no pacote

- **Operações de escalonamento no tamanho de vetores**
  - resize(vetor,novo_tamanho)
  - tipos unsigned:
    - resultado vetor de menor tamanho: bits descartados
    - resultado vetor de maior tamanho: 0 inseridos
  - tipos singed sinal preservado (bit mais a esquerda)
  - exemplo de operações:





```
  -- Id: R.1
  function RESIZE (ARG: SIGNED; NEW_SIZE: NATURAL) return SIGNED is
    alias INVEC: SIGNED(ARG'LENGTH-1 downto 0) is ARG;
    variable RESULT: SIGNED(NEW_SIZE-1 downto 0) := (others => '0');
    constant BOUND: INTEGER := MIN(ARG'LENGTH, RESULT'LENGTH)-2;
    -- VERIFIC: The RESIZE() function for signed does NOT behave the same as
the FEEDTHROUGH pragma does. It does truncation a bit different. Cannot use
pragma. Issue 2044
    -- attribute IS_SIGNED of ARG:constant is TRUE ;
    -- attribute SYNTHESIS_RETURN of RESULT:variable is "FEED_THROUGH" ;
  begin
    if (NEW_SIZE < 1) then return NAS;
    end if;
    if (ARG'LENGTH = 0) then return RESULT;
    end if;
    RESULT := (others => ARG(ARG'LEFT));
    if BOUND >= 0 then
      RESULT(BOUND downto 0) := INVEC(BOUND downto 0);
    end if;
    return RESULT;
  end RESIZE;
```

Funções definidas no pacote

- **Funções para teste de equivalência entre objetos std_logic**
  - std_match
  - semântica dos valores std_logic é considerada
  - tabela de equivalência adotada:

	U	X	0	1	Z	W	L	H	-
U	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
X	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
0	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
1	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
Z	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
W	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
L	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
H	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

```
-- support constants for STD_MATCH:

type BOOLEAN_TABLE is array(STD_ULOGIC, STD_ULOGIC) of BOOLEAN;

constant MATCH_TABLE: BOOLEAN_TABLE := (
    ---------------------------------------------------------------------
    -- U      X      0      1      Z      W      L      H      -
    ---------------------------------------------------------------------
    (FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,  TRUE), -- | U |
    (FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,  TRUE), -- | X |
    (FALSE, FALSE,  TRUE, FALSE, FALSE, FALSE,  TRUE, FALSE,  TRUE), -- | 0 |
    (FALSE, FALSE, FALSE,  TRUE, FALSE, FALSE, FALSE,  TRUE,  TRUE), -- | 1 |
    (FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,  TRUE), -- | Z |
    (FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,  TRUE), -- | W |
    (FALSE, FALSE,  TRUE, FALSE, FALSE, FALSE,  TRUE, FALSE,  TRUE), -- | L |
    (FALSE, FALSE, FALSE,  TRUE, FALSE, FALSE, FALSE,  TRUE,  TRUE), -- | H |
    ( TRUE,  TRUE,  TRUE,  TRUE,  TRUE,  TRUE,  TRUE,  TRUE,  TRUE)  -- | - |
    );

-- Id: M.1
function STD_MATCH (L, R: STD_ULOGIC) return BOOLEAN is
  variable VALUE: STD_ULOGIC;
begin
  return MATCH_TABLE(L, R);
end STD_MATCH;
```

```vhdl
-- Id: E.1
function RISING_EDGE (signal S: BIT) return BOOLEAN is
begin
  return S'EVENT and S = '1';
end RISING_EDGE;

-- Id: E.2
function FALLING_EDGE (signal S: BIT) return BOOLEAN is
begin
  return S'EVENT and S = '0';
end FALLING_EDGE;
```

Descrições empregando os pacotes numeric_bit numeric_std

- **Na maioria das ferramentas**
  - previamente compilado e armazenado na biblioteca ieee

- **Para tornar os itens do pacote visíveis:**
  - incluir cláusulas :

```
LIBRARY ieee;
USE ieee.numeric_bit.ALL;
```

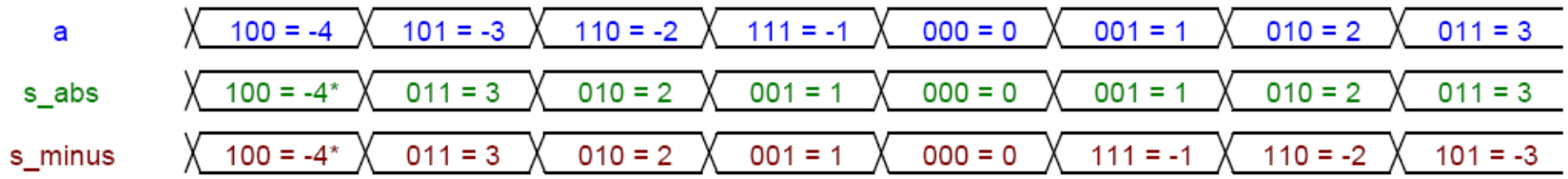
  - ou :

```
LIBRARY ieee;
USE ieee.numeric_bit.ALL;
```

Descrições empregando os pacotes numeric_bit numeric_std

- **Exemplo de funções aritméticas**: teste das funções abs -

```
 1 LIBRARY ieee;
 2 USE ieee.numeric_std.ALL;
 3
 4 ENTITY std1076a IS
 5   PORT (a               : IN  SIGNED(2 DOWNTO 0);
 6         s_abs, s_minus : OUT SIGNED(2 DOWNTO 0));
 7 END std1076a;
 8
 9 ARCHITECTURE teste OF std1076a IS
10 BEGIN
11   s_abs   <= abs(a);
12   s_minus <= -a;
13 END teste;
```

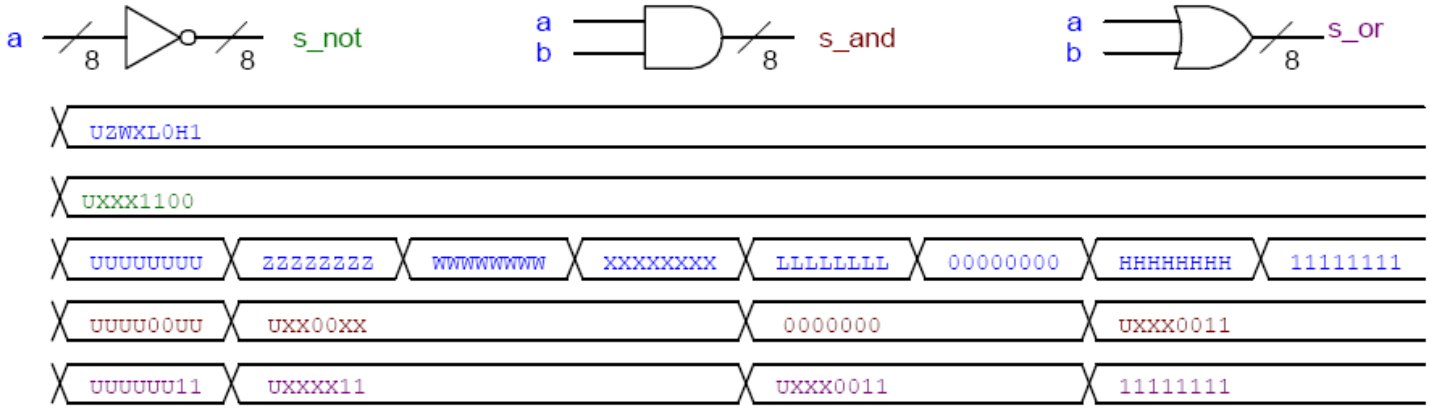


* não é possível representar +4 com 3 bits

Descrições empregando os pacotes numeric_bit numeric_std

- **Exemplo de funções lógicas**: funções NOT AND OR

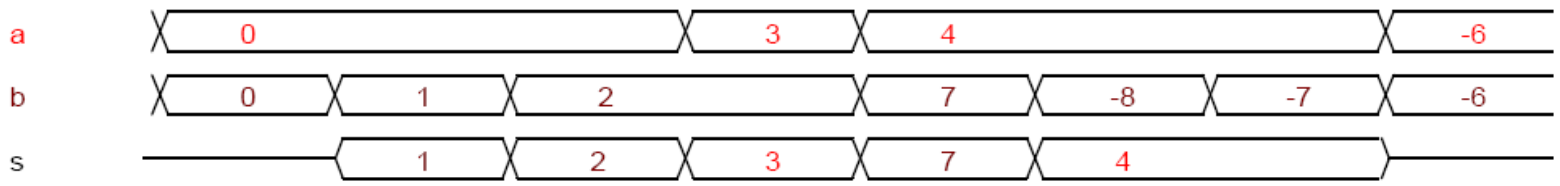
```
 1 LIBRARY ieee;
 2 USE ieee.numeric_std.ALL;
 3
 4 ENTITY std1076b IS
 5   PORT( a,b                : IN  SIGNED (7 DOWNTO 0);
 6         s_not, s_and, s_or : OUT SIGNED (7 DOWNTO 0));
 7   END std1076b;
 8
 9 ARCHITECTURE exemplo OF std1076b IS
10 BEGIN
11   s_not <=   NOT a;
12   s_and <= a AND b;
13   s_or  <= a OR  b;
14  exemplo;
```



Descrições empregando os pacotes numeric_bit numeric_std

- **Exemplo de saídas com estado de alta impedância**
  - tipos unsigned e signed (pacote numeric_std) → elementos std_logic
  - tipo std_logic possui função de resolução
  - portanto:
    - tipos unsigned e signed podem ser acionados por mais de um controlador

```
 1 LIBRARY ieee;
 2 USE ieee.numeric_std.ALL;
 3
 4 ENTITY std1076c IS
 5   PORT( a,b    : IN  SIGNED(3 DOWNTO 0);
 6         s      : OUT SIGNED(3 DOWNTO 0));
 7   END std1076c;
 8
 9 ARCHITECTURE exemplo OF std1076c IS
10 BEGIN
11   s <= a WHEN a > b ELSE "ZZZZ";
12   s <= b WHEN a < b ELSE "ZZZZ";
13 END exemplo;
```



Exemplos de síntese

- **Síntese com o valor não importa "-" e a função std_match**

  - saídas sm1 e sm2 → semântica de "-" considerada por std_match
    [tm igual a "0-1"] verdadeiro na condição [t(2), t(0)=0,1]
    [tm(1) igual a '-'] verdadeiro sempre

  - saídas s1 e s2 → semântica de "-" não considerada por =
    [t igual a "0-1"] nunca verdadeiro
    [t(1) igual a '-'] nunca verdadeiro

```
 5 ENTITY std1076h IS
 6   PORT (t,tm          : IN  STD_LOGIC_VECTOR (2 DOWNTO 0);
 7         s1,s2,sm1,sm2 : OUT STD_LOGIC);
 8 END std1076h;
 9
10 ARCHITECTURE teste OF std1076h IS
11 BEGIN
12   sm1 <= '1' WHEN std_match(tm, "0-1") ELSE 'Z';
13   sm2 <= '0' WHEN std_match(tm(1),'-') ELSE 'Z';
14
15   s1 <= '1' WHEN (t = "0-1") ELSE 'Z';
16   s2 <= '0' WHEN (t(1)= '-') ELSE 'Z';
17 END teste;
```

Exemplos de síntese

- **Resultado da síntese**

- saídas sm1 e sm2 → semântica de "-" considerada por std_match

[tm igual a "0-1"] verdadeiro na condição [t(2), t(0)=0,1]

```
sm1 <= '1' WHEN std_match(tm, "0-1") ELSE 'Z';
```

[tm(1) igual a '-'] verdadeiro sempre

```
sm2 <= '0' WHEN std_match(tm(1),'-') ELSE 'Z';
```

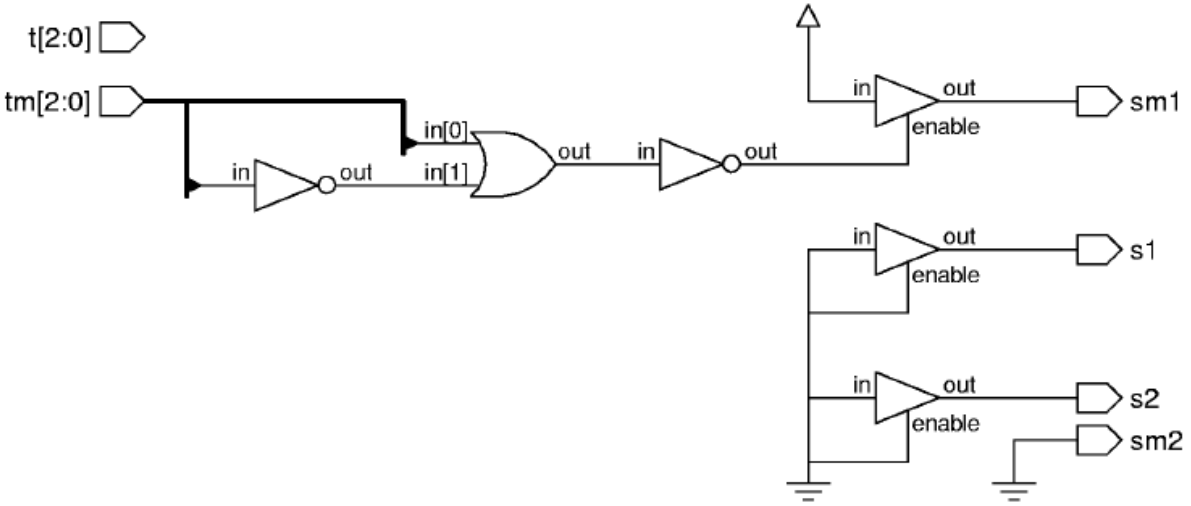
- saídas s1 e s2 → semântica de "-" não considerada por =

[t igual a "0-1"] nunca verdadeiro

```
s1 <= '1' WHEN (t = "0-1") ELSE 'Z';
```

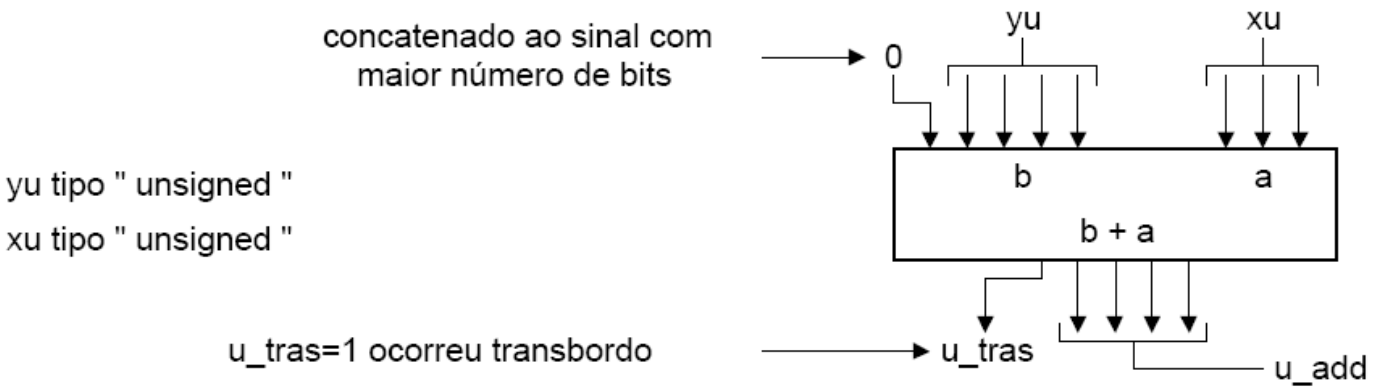
[t(1) igual a '-'] nunca verdadeiro

```
s2 <= '0' WHEN (t(1)= '-') ELSE 'Z';
```



Exemplos de síntese

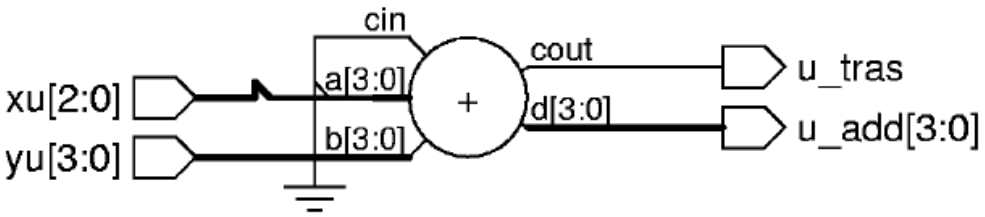
- **Detectando transbordo em operações de soma unsigned**
  - concatenar 0 na posição mais a direita (vetor com maior dimensão)
  - bit extra no resultado da soma indica o transbordo



Exemplos de síntese

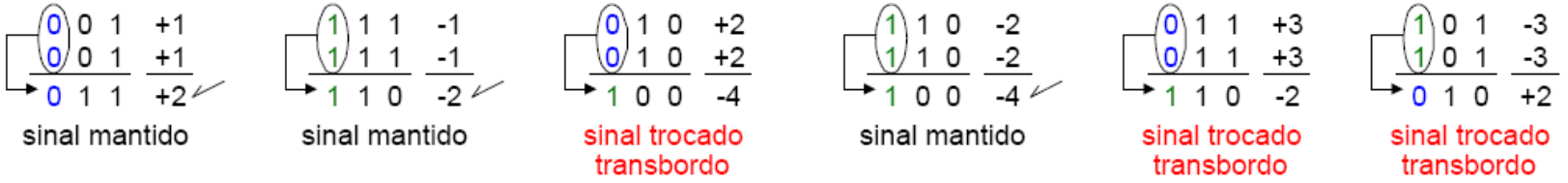
- **Detectando transbordo em operações de soma unsigned**
  - exemplo: xu 3 bits yu 4 bits
  - circuito sintetizado nível RTL: cout da primitiva empregado para detectar transbordo

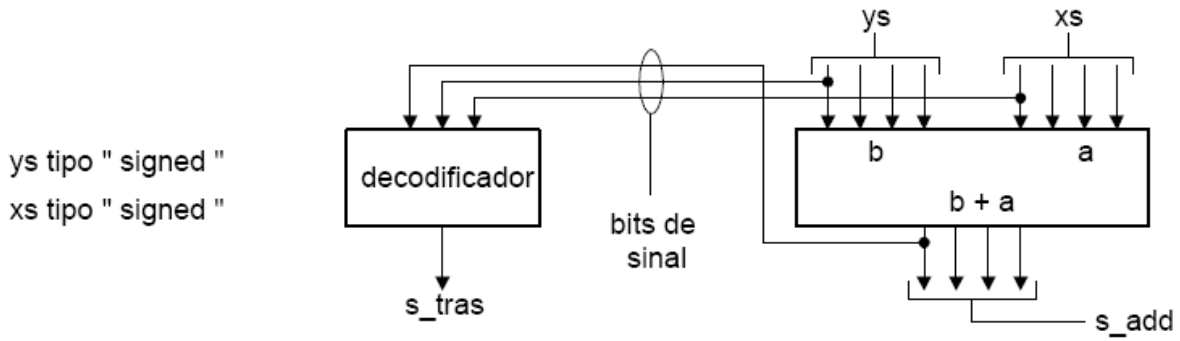
```
 5 ENTITY std1076o IS
 6   PORT (xu     : IN  UNSIGNED(2 DOWNTO 0);
 7         yu     : IN  UNSIGNED(3 DOWNTO 0);
 8         u_add  : OUT UNSIGNED(3 DOWNTO 0); -- u_add=xu+yu
 9         u_tras : OUT STD_LOGIC);          -- u_tras_u =1 se transbordo em xu+yu
10 END std1076o;
11
12 ARCHITECTURE teste OF std1076o IS
13   SIGNAL u_adi : UNSIGNED(4 DOWNTO 0);  -- resultado interno
14 BEGIN
15   u_adi  <= xu + ('0'& yu);       -- 0 concatenado com vetor de maior dimensao
16   u_tras <= u_adi(4);
17   u_add  <= u_adi(3 DOWNTO 0);
18 END teste;
```



Exemplos de síntese

- **Detectando transbordo em operações de soma tipo signed**
  - verificar o sinal do resultado da operação
  - sinais diferentes → resultado sempre correto
  - sinais iguais → resultado correto caso o sinal for mantido





Exemplos de síntese

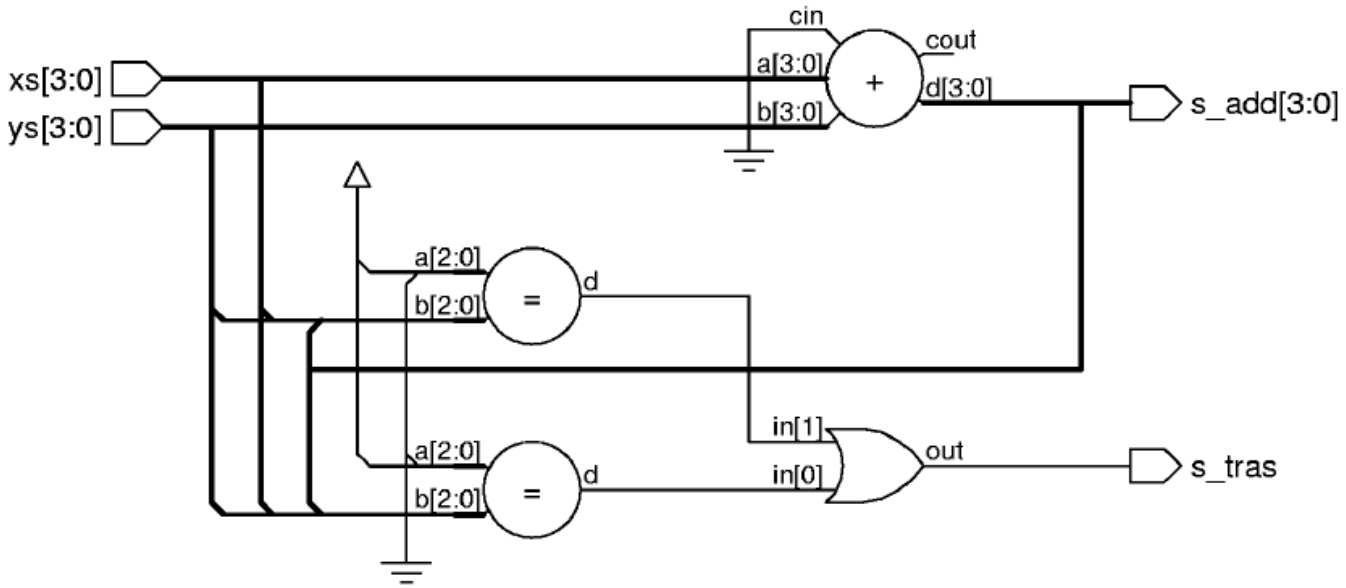
- **Detectando transbordo em operações de soma tipo signed**
  - construção WITH SELECT:

    implementa um decodificador que detecta as duas condições de transbordo

```
 1 LIBRARY ieee;
 2 USE ieee.std_logic_1164.ALL;
 3 USE ieee.numeric_std.ALL;
 4
 5 ENTITY std1076g IS
 6   PORT (xs     : IN     SIGNED (3 DOWNTO 0);
 7         ys     : IN     SIGNED (3 DOWNTO 0);
 8         s_add  : BUFFER SIGNED (3 DOWNTO 0);
 9         s_tras : OUT    STD_LOGIC);
10 END std1076g;
11
12 ARCHITECTURE teste OF std1076g IS
13   SIGNAL teste : STD_LOGIC_VECTOR(2 DOWNTO 0);
14 BEGIN
15   s_add <= xs + ys;
16
17   teste <= xs(3) & ys(3) & s_add(3);
18   WITH teste SELECT                    -- decodificador
19     s_tras <= '1' WHEN "001" | "110",  -- condicoes de transbordo
20               '0' WHEN OTHERS;         -- resultado correto
21 END teste;
```

Exemplos de síntese

- **Detectando transbordo em operações de soma tipo signed**
  - Circuito sintetizado nível RTL
  - s_trans 2 comparadores de 3 bits: (verificar condições de transbordo)
    0,0,1 = [xs(3), ys(3), s_add(3)] ou 1,1,0 = [xs(3), ys(3), s_add(3)]
  - bloco de soma corresponde a um somador de três bits
  - sinal de transbordo do bloco não é utilizado



Exemplos de síntese

- **Circuitos contadores**

  - fácil implementação → operações de soma a cada borda do relógio

  - detalhes no exemplo:

    - linha 17 comparação q_uns /= 10 expressão q_uns <= q_uns +1
      - funções "/=" e "+" permitem um dos operandos seja do tipo integer

    - linhas 15 e 18:
      - incorreto q_uns <= 0
      - valor 0 é do tipo inteiro, e o sinal q_uns é tipo unsigned

```
13  abc: PROCESS (ck, rst)
14   BEGIN
15     IF rst = 1 THEN q_uns <= "0000";
16     ELSIF rising_edge(ck) THEN
17       IF q_uns /= 10 THEN q_uns <= q_uns +1;    -- unsigned + integer possivel
18       ELSE                q_uns <= "0000";
19       END IF;
20     END IF;
21   END PROCESS abc;
```

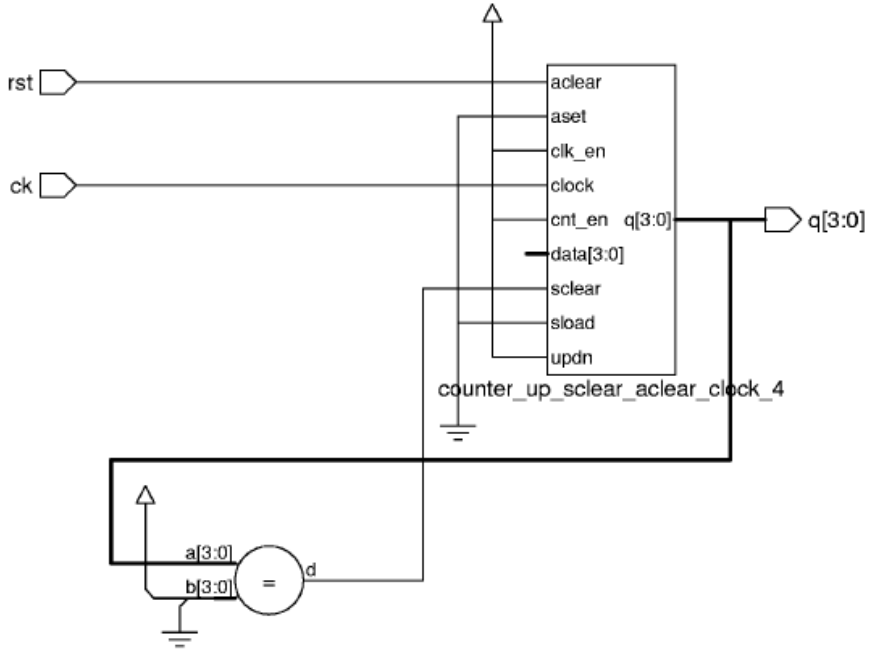
Exemplos de síntese

- **Circuitos contadores** - exemplo completo
  - note rotina de conversão - linhas 23 a 25
  - sinal q tipo std_logic_vector, sinal q_uns tipo unsigned

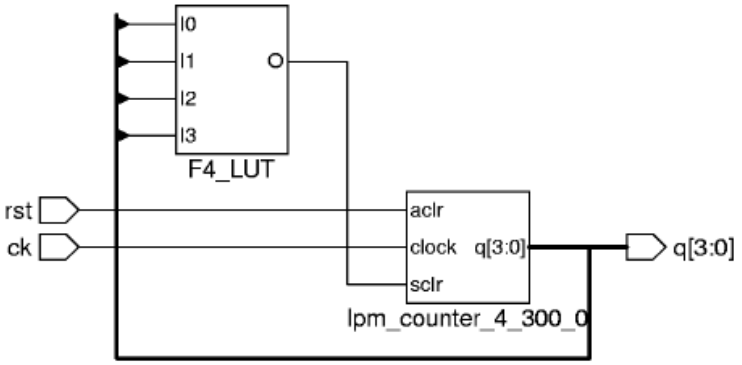
```
 5 ENTITY std1076w IS
 6   PORT (ck, rst  : IN  STD_LOGIC;
 7         q        : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
 8 END std1076w;
 9
10 ARCHITECTURE teste OF std1076w IS
11   SIGNAL q_uns : UNSIGNED(3 DOWNTO 0);
12 BEGIN
13   abc: PROCESS (ck, rst)
14   BEGIN
15     IF rst = 1 THEN q_uns <= "0000";
16     ELSIF rising_edge(ck) THEN
17       IF q_uns /= 10 THEN q_uns <= q_uns +1; -- unsigned + integer possivel
18       ELSE                q_uns <= "0000";
19       END IF;
20     END IF;
21   END PROCESS abc;
22
23   converte: FOR i IN 0 TO 3 GENERATE
24     q(i) <= q_uns(i);
25   END GENERATE converte;
26 END teste;
```

Exemplos de síntese

- **Circuitos contadores** - exemplo completo
  - circuito sintetizado:



nível RTL



nível porta lógica