Projeto com Circuitos Reconfiguráveis

Projeto lógico combinacional
Vetores e Somador combinacional

Prof. Daniel M. Muñoz Arboleda

FGA - UnB

Descrição VHDL porta AND2 de 8 bits

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3
4  entity and_vector is port(
5      a, b: in bit_vector (7 downto 0);
6      s: out bit_vector (7 downto 0));
7  end and_vector;
8
9  architecture comportamental of and_vector is
10 begin
11
12     process (a,b)
13     begin
14       for i in 7 downto 0 LOOP
15         s(i) <= a(i) and b(i);
16       end LOOP;
17     end process;
18
19 end comportamental;
20
```

Descrição VHDL somador 8 bits (exemplo 1)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4
5  entity somador is port(
6      a, b: in unsigned (7 downto 0);
7      s: out std_logic_vector(7 downto 0));
8  end somador;
9
10 architecture comportamental of somador is
11 signal result : integer;
12 begin
13
14    result <= CONV_INTEGER(a) + CONV_INTEGER(b);
15    s <= CONV_STD_LOGIC_VECTOR(result, 8);
16 end comportamental;
17
```

Descrição VHDL somador 8 bits (exemplo 2)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  USE ieee.std_logic_signed.all;
5  USE work.math.all;
6
7  entity somadorB is port(
8      a, b: in  std_logic_vector(15 downto 0);
9      s: out std_logic_vector(15 downto 0));
10 end somadorB;
11
12 architecture comportamental of somadorB is
13 signal result1: integer;  -- signed
14 signal result2: integer;  -- signed
15 signal result3: integer;  -- signed
16 begin
17
18    result1 <= vect_to_int(a);
19    result2 <= vect_to_int(b);
20    result3 <= result1 + result2;
21    s <= int_to_st16 (result3);
22 end comportamental;
23
```

Descrição VHDL somador 8 bits (exemplo 3)

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  USE ieee.std_logic_signed.all;
5  USE work.math.all;
6
7  entity somador is port(
8      a, b: in std_logic_vector(15 downto 0);
9      s: out  std_logic_vector(15 downto 0));
10 end somador;
11
12 architecture comportamental of somador is
13 signal result1: signed(15 downto 0);  -- signed
14 signal result2: signed(15 downto 0);  -- signed
15 signal result3: signed(15 downto 0);  -- signed
16 begin
17
18    result1 <= signed(a);
19    result2 <= signed(b);
20    result3 <= result1 + result2;
21    s <= std_logic_vector( result3);
22 end comportamental;
23
```

Como evitar o *overflow* do somador?