Laboratório de Lógica Digital

Prática IV

Para projetos grandes ou projetos em que estão trabalhando mais de uma pessoa, é viável a utilização de componentes. Um componente interliga entidades de modo a criar uma hierarquia entre elas. Outra utilidade da utilização de componentes é a não repetição de comandos que serão muito utilizados. Basicamente, um componente é dividido em declaração e solicitação.

A declaração de um componente é semelhante à declaração de uma entidade, utilizando o comando COMPONENT no lugar do comando ENTITY. Esta declaração deve ser feita dentro da arquitetura, no mesmo local onde são declaradas as classes.

Exemplo:

Somador completo de 1 bit.

```
1  entity somador is
2  port(a, b, ve: in bit;
3          s, va: out bit);
4  end somador;
5
6  architecture comportamento of somador is
7  begin
8     s <= a XOR b XOR ve;                          -- soma 1
9    va <= (a AND b) OR (a AND ve) OR (b AND ve); -- vai 1
10
11 end comportamento;
```

A solicitação de um componente é feita através de um rótulo, um nome e um mapeamento de portas para o componente. O mapa de portas consiste em uma lista que faz a ligação entre os sinais do componente com os sinais da entidade que solicita o componente. A lista de sinais do mapa pode seguir uma ordem posicional (o primeiro sinal declarado no componente será ligado ao primeiro sinal declarado do mapa de portas) ou nomeada, onde o programador deve definir qual sinal recebe qual.

Somador de 4 bits.

```
1  entity componente is
2  port(x, y: in bit_vector(3 downto 0);
3         ze: in bit;
4         za: out bit;
5          s: out bit_Vector(3 downto 0);
6  end componente;
7
8  architecture comportamento of componente is
9
10  COMPONENT somador
11    PORT(a, b, ve: in bit; s, va: out bit);
12  END COMPONENT;
13
14 signal v: bit_vector(3 downto 1); -- vai 1 interno
15
16 begin
17    x0: somador PORT MAP(x(0), y(0),   ze, s(0), v(1));
18    x1: somador PORT MAP(x(1), y(1), v(1), s(1), v(2));
19    x2: somador PORT MAP(x(2), y(2), v(2), s(2), v(3));
20    x3: somador PORT MAP(x(3), y(3), v(3), s(3), za);
21
22 end comportamento;
```

Exercício:

O código abaixo supostamente apresenta um circuito subtrator. Verifique no software Quartus II sua funcionalidade, e depois faça um desenho do seu diagrama baseado no vhdl (e não no circuito gerado pelo Quartus) além de fazer sua tabela verdade.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity fs1 is
Port ( x : in STD_LOGIC;
y : in STD_LOGIC;
bin : in STD_LOGIC;
d : out STD_LOGIC;
bout : out STD_LOGIC);
end fs1;
architecture Behavioral of fs1 is
begin
d<=x xor y xor bin;
bout<=((not x)and y)or((not x)and bin)or(y and bin);
end Behavioral;
```