UNIVERSIDADE FEDERAL DE MINAS GERAIS
UFMG – CAMPUS PAMPULHA

Disciplina: Laboratório de Sistemas Digitais
Professor: Ricardo de Oliveira Duarte
Estudantes: Igor Braga de Lima, Matheus Vinícius Freitas Oliveira dos Santos e Stéphanie Pereira Barbosa
Turma: PN5
Guia de aula: 04

ATIVIDADE TEÓRICA

1) Como podemos identificar que uma *architecture* em VHDL foi descrita usando o modelo *data-flow*?
Uma *architecture* que usa o modelo *data-flow* pode ser identificada por sua composição: somente declarações *concurrent, conditional* e *selected signal*, ausência do *process.*

2) Qual situação na qual devemos adotar o modelo de representação de *architecture data-flow*?
Em circuitos simples e/ou pequenos.

3) Qual a vantagem de se descrever um circuito VHDL usando *data-flow*?
Obter maior entendimento sobre o caminho que os dados/entradas percorrem, e maior noção do circuito final a ser produzido.

4) Qual a desvantagem de se descrever um circuito VHDL usando *data-flow*?
Em sistemas mais complexos, o *data-flow* perde interpretabilidade, dificultando o entendimento do sistema.

5) Como podemos identificar que uma *architecture* em VHDL foi descrita usando o modelo *behavioral*?
Diferente do *data-flow*, em uma *architecture* que utiliza o modelo *behavioral* é fundamentada o uso do *process statment*, logo, basta identificar essas declarações.

6) Qual situação na qual devemos adota o modelo de representação de *architecture behavioral*?
Quando queremos descrever circuitos mais complexos.

7) Qual a vantagem de se descrever um circuito VHDL usando *behavioral*?
É mais prático para descrever circuitos complexos, pois não precisa se

preocupar com o comportamento de cada elemento que integra o sistema.

8) Qual a desvantagem de se descrever um circuito VHDL usando *behavioral*?
Não é possível obter detalhes de como o circuito será implementado apenas analisando o código VHDL.

9) Como declarar um *process* em VHDL?
```
my_label: process(sensitivity_list) is
        <item_declaration>
begin
        <sequential_statements>
end process my_label;
```

10) Qual característica das declarações contidas dentro do corpo de um *process* em VHDL?
São declarações que são executadas de forma sequencial.

11) O que dispara a execução do conteúdo de um *process* em VHDL?
Quando há alteração nos sinais contidos na lista de sensibilidade.

12) Quais as regras para se definir o que deve ser incluído na lista de sensibilidade de um *process*?
Todos os sinais que são de entradas e que serão utilizados dentro do *process*.

13) Quais as declarações sequenciais em VHDL que você estudou nesse capítulo?
As declarações *if*, *case* e a de atribuição de sinal.

14) Quando é que uma declaração de atribuição de sinais pode ser considerada uma declaração sequencial?
Quando ela estiver dentro de uma declaração *process.*

15) Qual a diferença da declaração sequencial *case* para a declaração sequencial *if*?
A diferença é que a declaração *case* depende de uma expressão de controle.

16) Qual a principal lição que você pode obter da seção 5.5 do livro?
Ao utilizar o *process*, embora se esteja descrevendo o comportamento de um circuito digital de forma algorítmica, deve-se ter em mente de que se está descrevendo o comportamento de um hardware e não um software.

ATIVIDADE PRÁTICA

1) Escolha uma das funções do exercício 1 da pág. 68 da referência principal do curso e implemente-o em VHDL.

A função escolhida foi: e) $F(A, B, C, D) = \sum(1,2)$

CASE:

funcao1E.vhd > ...

```
1   -- Solução do exercício 1e usando case
2   -- 1-e) F(A, B, C, D) = Somatório(1,2)
3
4   library ieee;
5   use ieee.std_logic_1164.all;
6
7   entity exercicio_1 is
8     port(A, B, C, D : in std_logic;
9          F : out std_logic);
10  end exercicio_1;
11
12  architecture funcao1E of exercicio_1 is
13    signal ABCD : std_logic_vector(3 downto 0);
14    begin
15      ABCD <= A & B & C & D;
16      my_proc : process(ABCD)
17      begin
18        case(ABCD) is
19          when "0001" => F <= '1';
20          when "0010" => F <= '1';
21          when others => F <= '0';
22        end case;
23      end process my_proc;
24  end funcao1E;
```

IF:

funcao1E.vhd > ...

```
1   -- Solução do exercício 1e usando if
2   -- 1-e) F(A, B, C, D) = Somatório(1,2)
3
4   library ieee;
5   use ieee.std_logic_1164.all;
6
7   entity exercicio_1 is
8     port(A, B, C, D : in std_logic;
9          F : out std_logic);
10  end exercicio_1;
11
12  architecture funcao1E of exercicio_1 is
13    signal ABCD : std_logic_vector(3 downto 0);
14    begin
15      ABCD <= A & B & C & D;
16      my_proc : process(ABCD)
17      begin
18        if (ABCD = "0001") then F <= '1';
19        elsif (ABCD = "0010") then F <= '1';
20        else F <= '0';
21        end if;
22      end process my_proc;
23  end funcao1E;
```

2) Escreva um *testbench* para a função em VHDL que você implementou.

CASE:

tb_funcao1E.vhd > ...

```vhdl
-- Testbench da função 1E

library ieee;
use ieee.std_logic_1164.all;

entity tb_funcao1E is
end tb_funcao1E;

architecture teste of tb_funcao1E is
  component exercicio_1 is
    port(A, B, C, D : in std_logic;
         F : out std_logic);
  end component;

  signal a, b, c, d, f: std_logic;
  begin
    instancia_somador: exercicio_1 port map(A => a, B => b, C => c, D => d, F => f);
    a <= '0', '1' after 20 ns, '0' after 40 ns, '1' after 60 ns;
    b <= '0', '1' after 10 ns, '1' after 30 ns, '0' after 50 ns;
    c <= '0', '0' after 10 ns, '1' after 30 ns, '1' after 50 ns;
    d <= '0', '1' after 10 ns, '0' after 30 ns, '0' after 50 ns;
end teste;
```

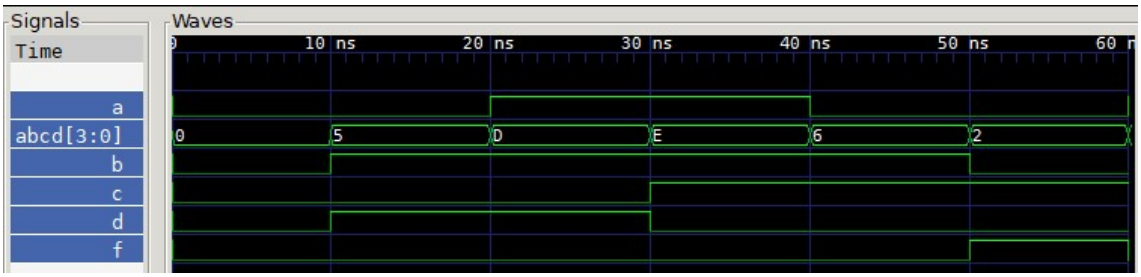
IF:

tb_funcao1E.vhd > ...

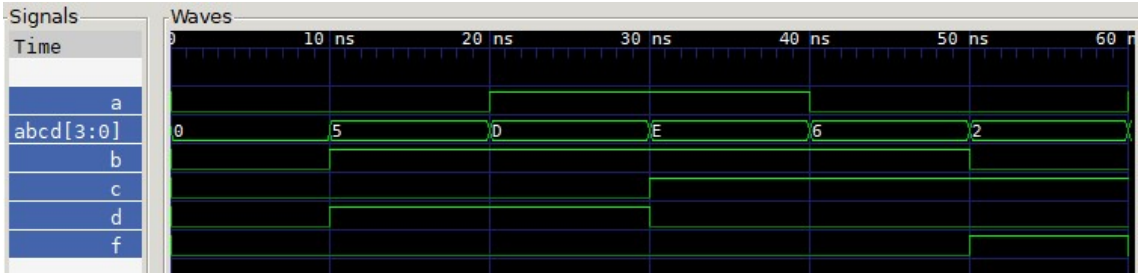
```vhdl
-- Testbench da função 1E

library ieee;
use ieee.std_logic_1164.all;

entity tb_funcao1E is
end tb_funcao1E;

architecture teste of tb_funcao1E is
  component exercicio_1 is
    port(A, B, C, D : in std_logic;
         F : out std_logic);
  end component;

  signal a, b, c, d, f: std_logic;
  begin
    instancia_somador: exercicio_1 port map(A => a, B => b, C => c, D => d, F => f);
    a <= '0', '1' after 20 ns, '0' after 40 ns, '1' after 60 ns;
    b <= '0', '1' after 10 ns, '1' after 30 ns, '0' after 50 ns;
    c <= '0', '0' after 10 ns, '1' after 30 ns, '1' after 50 ns;
    d <= '0', '1' after 10 ns, '0' after 30 ns, '0' after 50 ns;
end teste;
```

3) Compile e simule.

CASE:



IF:



4) Faça o exercício 7 da pág. 69 da referência principal do curso e implemente-o em VHDL.

CASE:

funcao.vhd > ...

```vhdl
-- Solução do exercício 7 usando case
-- decodificador 3x8

library ieee;
use ieee.std_logic_1164.all;

entity exercicio_7 is
  port(SEL : in std_logic_vector(2 downto 0);
       D : out std_logic_vector(7 downto 0));
end exercicio_7;

architecture funcao of exercicio_7 is
  begin
    my_proc : process(SEL)
    begin
      case(SEL) is
        when "000" => D <= "11111110";
        when "001" => D <= "11111101";
        when "010" => D <= "11111011";
        when "011" => D <= "11110111";
        when "100" => D <= "11101111";
        when "101" => D <= "11011111";
        when "110" => D <= "10111111";
        when "111" => D <= "01111111";
        when others => D <= "11111111";
      end case;
    end process my_proc;
end funcao;
```

IF:

funcao.vhd > ...

```vhdl
-- Solução do exercício 7 usando if
-- decodificador 3x8

library ieee;
use ieee.std_logic_1164.all;

entity exercicio_7 is
  port(SEL : in std_logic_vector(2 downto 0);
       D : out std_logic_vector(7 downto 0));
end exercicio_7;

architecture funcao of exercicio_7 is
  begin
    my_proc : process(SEL)
    begin
      if(SEL = "000") then D <= "11111110";
      elsif(SEL = "001") then D <= "11111101";
      elsif(SEL = "010") then D <= "11111011";
      elsif(SEL = "011") then D <= "11110111";
      elsif(SEL = "100") then D <= "11101111";
      elsif(SEL = "101") then D <= "11011111";
      elsif(SEL = "110") then D <= "10111111";
      elsif(SEL = "111") then D <= "01111111";
      else D <= "11111111";
      end if;
    end process my_proc;
end funcao;
```

5) Escreva um *testbench* para a função em VHDL que você implementou.

CASE:

tb_funcao.vhd > ...

```vhdl
-- Testbench do descodificador 3x8

library ieee;
use ieee.std_logic_1164.all;

entity tb_funcao is
end tb_funcao;

architecture teste of tb_funcao is
  component exercicio_7 is
    port(SEL : in std_logic_vector(2 downto 0);
         D : out std_logic_vector(7 downto 0));
  end component;

  signal sel : std_logic_vector(2 downto 0);
  signal d : std_logic_vector(7 downto 0);
  begin
    instancia : exercicio_7 port map(SEL => sel, D => d);
    sel <= "000", "001" after 10 ns, "010" after 20 ns, "011" after 30 ns,
           "100" after 40 ns, "101" after 50 ns, "110" after 60 ns, "111" after 70 ns;
end teste;
```

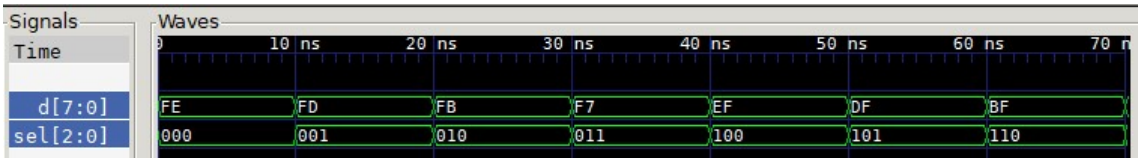
IF:

tb_funcao.vhd > ...

```vhdl
-- Testbench do descodificador 3x8

library ieee;
use ieee.std_logic_1164.all;

entity tb_funcao is
end tb_funcao;

architecture teste of tb_funcao is
  component exercicio_7 is
    port(SEL : in std_logic_vector(2 downto 0);
         D : out std_logic_vector(7 downto 0));
  end component;

  signal sel : std_logic_vector(2 downto 0);
  signal d : std_logic_vector(7 downto 0);
  begin
    instancia : exercicio_7 port map(SEL => sel, D => d);
    sel <= "000", "001" after 10 ns, "010" after 20 ns, "011" after 30 ns,
           "100" after 40 ns, "101" after 50 ns, "110" after 60 ns, "111" after 70 ns;
end teste;
```

6) Compile e simule.

CASE:



IF: