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: 06

ATIVIDADE TEÓRICA

1) Quantos objetos de dados diferentes existem em VHDL e quais são eles?
Signals, variables, constants e *files.*

2) Qual o objeto de dados que é exclusivamente usado em *testbench* se simulações?
Files.

3) Quais os tipos de dados estudados nesse capítulo do livro?
Signals, variables, constants.

4) *Signals* podem ser inicializados? Em quais situações?
A inicialização de *signals* pode ocorrer para situações onde será realizado teste/simulação.

5) Quais as diferenças de *variables* para *signals*?
Variables possuem a limitação de só poderem ser utilizadas dentro de *processes, functions* e *procedures*. Diferente de *signals, variables* podem ser atualizadas durante o *process.*

6) Qual o efeito de se atribuir novos valores a *variables* e *signals* dentro de um *process*?
Ao atribuir valor a *variable* dentro de um *process*, o valor da *variable* muda no mesmo momento, já em um *signal* o valor só muda após a término do *process*.

7) Em quais situações de descrição de um sistema VHDL devemos usar as *variables*?
Variables devem ser usadas dentro de um *process* como um contador de iterações, ou como um armazenador de valor temporário para, por exemplo, a realização de algum tipo de operação.

8) Porque o conteúdo de um *process* deve ser simples e curto?
Deve ser simples e curto porque os *signals* utilizados no *process* só têm seus valores modificados ao término do *process*. Além disso, um *process* simples e curto facilita o processo de síntese.

9) Quais os tipos de dados existentes em VHDL?
Existem os tipos *std_logic, std_logic_vector, unsigned, signed, bit, bit_vector, integer, integer_vector, natural, positive, character, string, boolean, boolean_vector.*

10) VHDL te permite criar um novo tipo de dado?
Sim.

11) Quais os tipos de dados mais comumente em encontrados em descrições de sistemas sintetizáveis em VHDL?
Std logic, std logic vectors e *enumerated*.

12) Quais são as palavras reservadas da linguagem VHDL normalmente encontradas na definição de tipos de dados?
type, is, range e to.

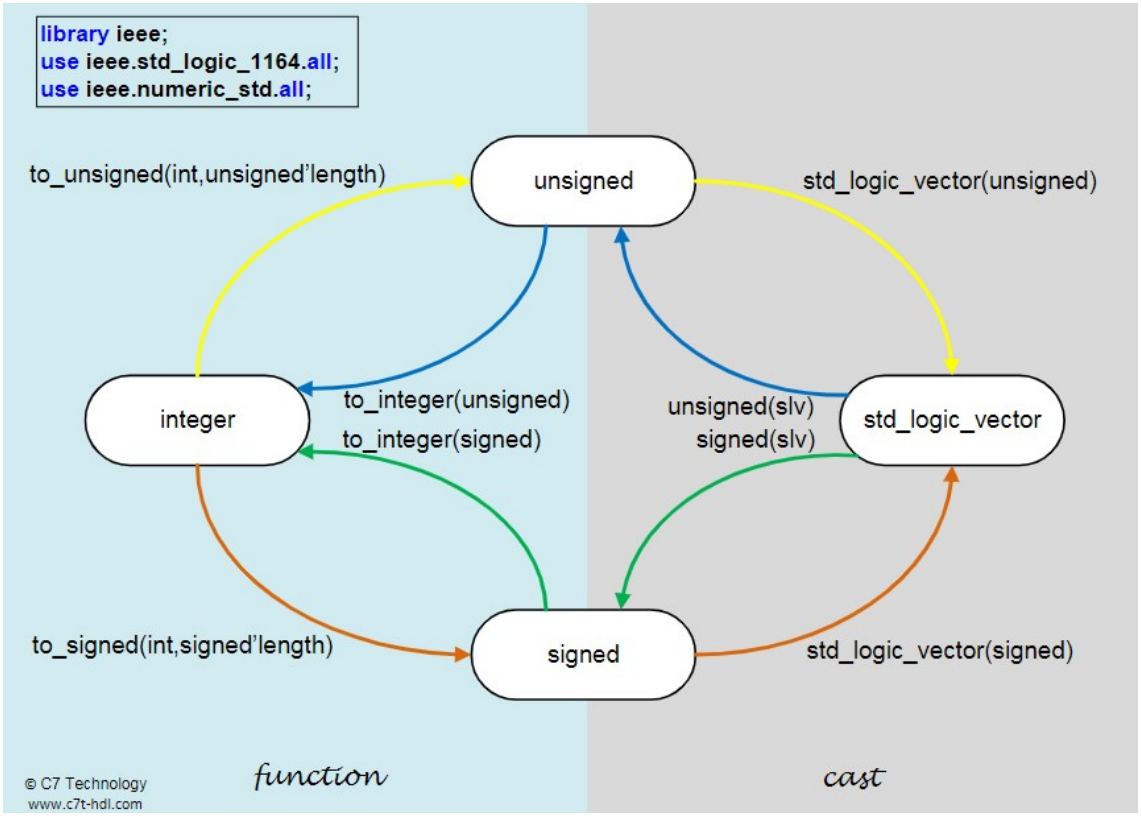
13) Quais *packages* são padrões da biblioteca IEEE e quais *packages* não são padrões?
Os packages padrões são ieee.numeric_std, ieee.std_logic 1164 e os não padrões são ieee.numeric_signed, ieee.numeric_unsigned, ieee.numeric_arithmetic.

14) Por que eu preciso usar *signals* do tipo *signed* ou *unsigned* ao invés de um tipo *std_logic*?
Quando eu preciso usar operações aritméticas dentro do sistema em VHDL

15) Por que devemos usar *std_logic* para habitualmente declarar nossos sinais em uma *entity*?
Porque é um tipo de dado que permite combinar com outros componentes que tenham a mesma interface.

16) Quais valores o tipo de dados *std_logic* pode assumir?
U, X, 0, 1, Z, W, L, H, -

17) Como devo proceder para converter um tipo para o outro tipo?
Para cada tipo de conversão usa-se um comando diferente:



ATIVIDADE PRÁTICA

1) Projete em VHDL uma ULA de 8 bits(usando o modelo comportamental) que realize as seguintes funções: soma com sinal; soma sem sinal; subtração; or lógico; and lógico; xor lógico.

funcao.vhd > ...

```vhdl
-- ULA

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ULA is
  port(A, B : in std_logic_vector(7 downto 0);
       ULA_SEL : in std_logic_vector(2 downto 0);
       ULA_OUT : out std_logic_vector(7 downto 0));
end ULA;

architecture funcao of ULA is
  begin
    my_proc : process(A, B, ULA_SEL)
    begin
      case(ULA_SEL) is
        when "001" => ULA_OUT <= std_logic_vector(signed(A) + signed(B)); -- Soma com sinal
        when "010" => ULA_OUT <= std_logic_vector(unsigned(A) + unsigned(B)); -- Soma sem sinal
        when "011" => ULA_OUT <= std_logic_vector(unsigned(A) - unsigned(B)); -- Subtração
        when "100" => ULA_OUT <= A or B;  -- OR lógico
        when "101" => ULA_OUT <= A and B; -- AND lógico
        when "110" => ULA_OUT <= A xor B; -- XOR lógico
        when others => ULA_OUT <= "00000000";
      end case;
    end process my_proc;
end funcao;
```

2) Escreva um *testbench* para o sistema em VHDL que você implementou.

tb_funcao.vhd > ...

```vhdl
-- Testbench da ULA

library ieee;
use ieee.std_logic_1164.all;

entity tb_funcao is
end tb_funcao;

architecture teste of tb_funcao is
  component ULA is
    port(A, B : in std_logic_vector(7 downto 0);
         ULA_SEL : in std_logic_vector(2 downto 0);
         ULA_OUT : out std_logic_vector(7 downto 0));
  end component;

  signal a, b : std_logic_vector(7 downto 0);
  signal ula_sel : std_logic_vector(2 downto 0);
  signal ula_out : std_logic_vector(7 downto 0);

  begin
    instancia : ULA port map(A => a, B => b, ULA_SEL => ula_sel, ULA_OUT => ula_out);
    a <= "00000000", "11111111" after 35 ns;
    b <= "11111111", "00000000" after 35 ns;
    ula_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;
```

3) Compile, simule e verifique o comportamento do seu circuito.