Computação Digital

ENG 1448

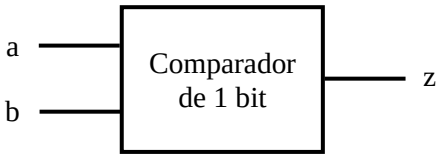
Laboratório 01

Objetivos:

- Primeiro contato com o ISE
- Fazer o roteiro completo, desde o código VHDL até a implementação, passando pela simulação
- Verificar a concorrência da linguagem
- Uso do Kit Spartan 3E
- Primeiro uso do iMPACT
- Instanciar outras entidades como componentes de um projeto maior

**Primeira parte do laboratório.**

1) Faça um código em VHDL de um comparador de 1 bit. Assim, a saída deverá ser verdadeira (‘1’), sempre que as duas palavras (de 1 bit cada) forem iguais. Se necessário, use *signal* para sinais internos.




Após o código escrito, uma vez que não apresente erros no “Check Syntax”, sintetize (synthetize) seu código.

Verifique:

a) No *Summary*, verifique
   número de LUTs utilizadas e disponíveis
   número de IOBs utilizadas e disponíveis
b) Navegue em *Detailed Report*, veja o *delay* do circuito.
c) Em *Synthetize*, analise tanto o *RTL Schematic* quanto o *Technology Schematic*. Investigue as diferenças e relate aqui. Pesquise.

2) Crie um arquivo de teste (*testbench*) com o estímulo abaixo e observe no *ISim Simulator*. Relate sua observação.

```
-- Stimulus process
stim_proc: process
begin
    -- insert stimulus here
    a <= '0';
    b <= '0';
    wait for clock_period*10;

    a <= '0';
    b <= '1';
    wait for clock_period*10;

    a <= '1';
    b <= '0';
    wait for clock_period*10;

    a <= '1';
    b <= '1';
    wait for clock_period*10;

    -- wait forever..
    wait;
end process;
```

3) Troque as posições das linhas de atribuição dentro da arquitetura e teste novamente. Se você não utilizou *signals* internos, tente implementar com *signals* como mostrado abaixo. Que diferenças você encontrou na simulação?

```
primeira_comp <= a AND b;
segunda_comp  <= not(a) AND not(b);
z             <= primeira_comp OR segunda_comp;
```

```
z             <= primeira_comp OR segunda_comp;
primeira_comp <= a AND b;
segunda_comp  <= not(a) AND not(b);
```

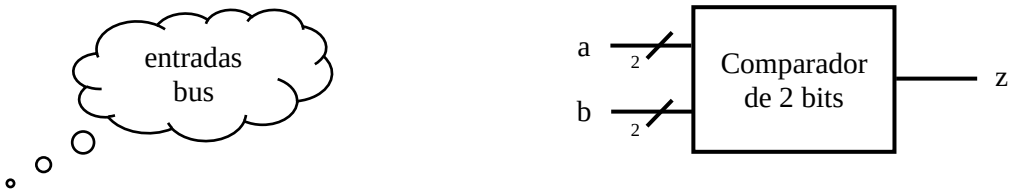
4) Estando funcionando, é hora de implementar! Use o seguinte conteúdo, confronte com o manual da placa, e implemente seu arquivo. Use o iMPACT para configurar o kit.

Conteúdo do *constraints file*:

```
NET "b" LOC = "L13" | IOSTANDARD=LVTTL | PULLUP ;
NET "a" LOC = "L14" | IOSTANDARD=LVTTL | PULLUP ;
NET "z" LOC = "F12" | IOSTANDARD=LVTTL;
```

**Segunda parte do laboratório.**

5) Agora você iniciará um outro projeto do zero e fará um comparador de 2 bits, com a seguinte estrutura de portas:




Só que você instanciará o Comparador de 1 bit criado na 1ª parte usando a seguinte estrutura:

```
unit_label : entity lib_name.entity_name(arch_name)
    port map (
        port_signal => actual_signal,
        port_signal => actual_signal,
        . . .
        port_signal => actual_signal
    );
```

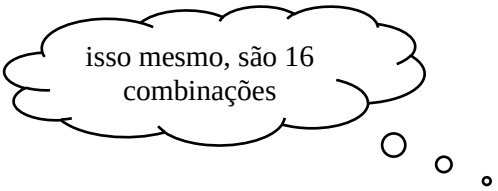
A primeira parte dá um “apelido” para o uso dessa entidade/componente que está sendo evocada. A biblioteca default é ***work***.

No mapeamento de portas, é sempre da porta do componente para (=>) a porta/sinal da entidade corrente, no caso o Comparador de 2 bits.

Pequena cola...

```
CompBit0 : entity work.comp_1bit(Behavioral)
    port map(
        a => a(0),
        b => b(0),
        z => comp(0)
    );
```

6) Modifique o *testbench* e faça as adaptações necessárias para que ele sirva de bancada de testes para seu novo comparador de 2 bits.




7) E por último, implemente este comparador usando as 4 chavinhas para as 2 palavras de 2 bits cada, e um led para a saída (aceso = palavras iguais).

P.S.: Os nomes dos pinos estão no manual.

Relatório na próxima aula, ok?

Detalhe o quanto puder. Enviar relatório com fotos dos *testbench*’s, código, etc.

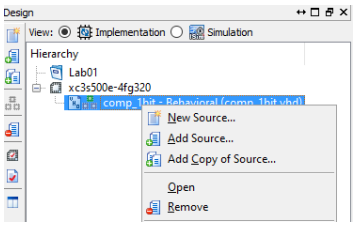
**Criando um arquivo de *testbench*.**

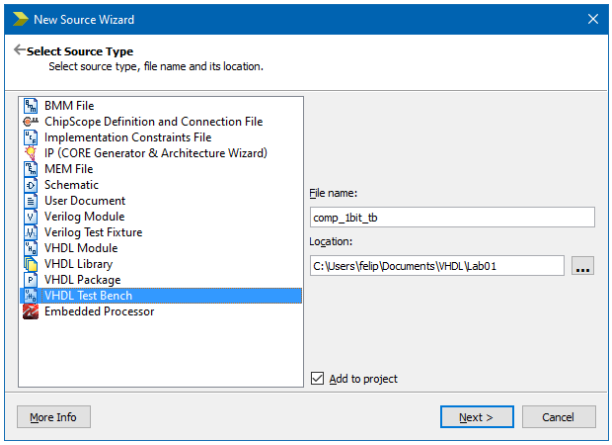
a) Em *Implementation* clique com o botão direito e vá em *New Source*. Escolha a opção *VHDL Test Bench*.

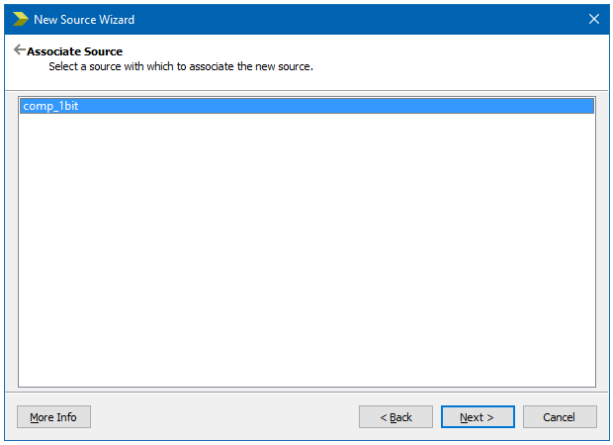
b) Escolha um nome adequado ao arquivo de testes (*testbench*), por exemplo: [nome do arquivo]_tb.vhd

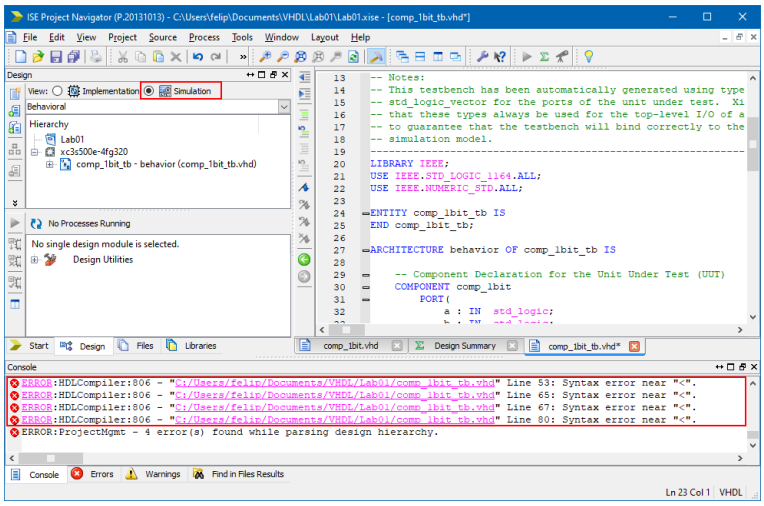
Desta forma é fácil saber a que entidade aquele *testbench* pertence.

c) Escolha a entidade que deseja simular.



d) Na aba de simulação, você deverá ver alguns erros de sintaxe pois o *Project Navigator* espera um circuito síncrono (com sinal de clock).
Nosso circuito é assíncrono (não possui sinal de clock), podemos, entretanto, declarar um sinal de clock apenas para fins de simulação. Aperte *CTRL+H* e substitua "<*clock*>" por "*clock*". Após clicar em "*Replace All*" devemos criar um signal.

Exemplo de *testbench*.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY example_tb IS
END example_tb;

ARCHITECTURE behavior OF example_tb IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT example
        PORT(
            A : IN   std_logic_vector(3 downto 0);
            B : IN   std_logic_vector(3 downto 0);
            C : OUT  std_logic
        );
    END COMPONENT;

    --Inputs
    signal A : std_logic_vector(3 downto 0) := (others => '0');
    signal B : std_logic_vector(3 downto 0) := (others => '0');

    --Outputs
    signal C : std_logic;

    signal clock : std_logic := '0';
    constant clock_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: example PORT MAP (
        a => a,
        b => b,
        z => z
    );

    -- Clock process definitions
    clock_process : process
    begin
        clock <= '0';
        wait for clock_period/2;
        clock <= '1';
        wait for clock_period/2;
    end process;

    -- Stimulus process
    stim_proc : process
    begin
        -- insert stimulus here
        a <= std_logic_vector(to_unsigned(0, 4));
        b <= std_logic_vector(to_unsigned(0, 4));

        for I_A in 0 to 15 loop
            a <= std_logic_vector(to_unsigned(I_A, 4));
            for I_B in 0 to 15 loop
                b <= std_logic_vector(to_unsigned(I_B, 4));
                wait for clock_period*10;
            end loop;
        end loop;

        wait;
    end process;
END;
```