CI INOVADOR RESIDÊNCIA EM MICROELETRÔNICA	**Programa CI Inovador** **Disciplina: Introdução à Verificação** **Profs.: Joseana Macêdo Fechine Régis de Araújo,** **Marcos Morais, Elmar Melcher - UASC/UAEE/UFCG** **Atividade Prática 01** **(Verificação Funcional - Parte I)**

OBJETIVO: Implementação de Testbench.

ATIVIDADE 1 (Testbench - Simples): Considerar a descrição de um circuito simples (*design.sv*) e o testbench utilizado para verificar o funcionamento do referido circuito (*testbench.sv*).

$$y = \overline{b}\,\overline{c} + a\overline{b}$$

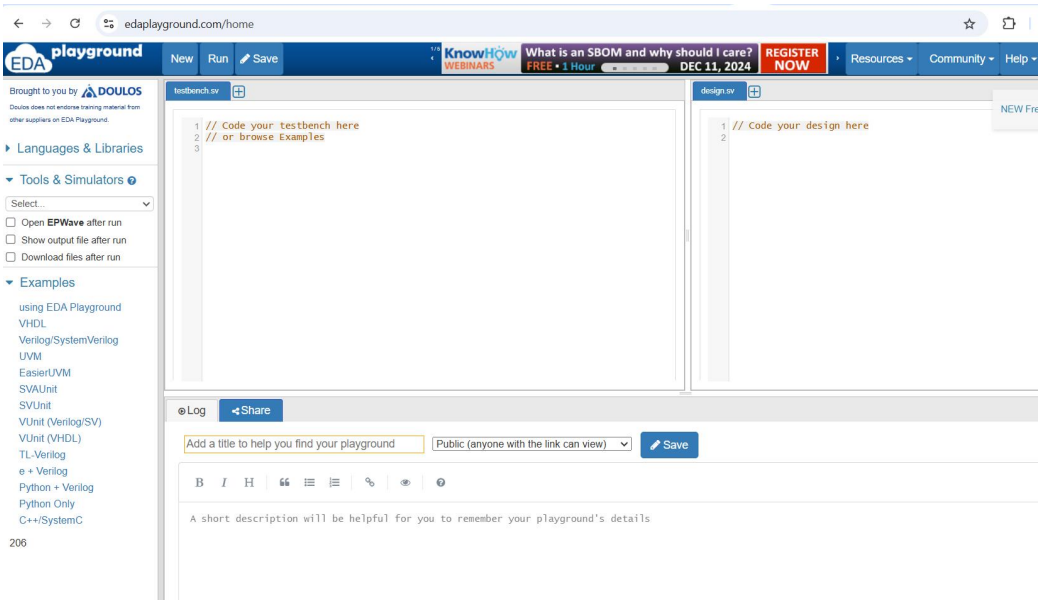
design.sv

```
module sillyfunction (input logic a, b , c,
                      output logic y);
    assign y = ~b & ~c | a & ~b;
endmodule
```

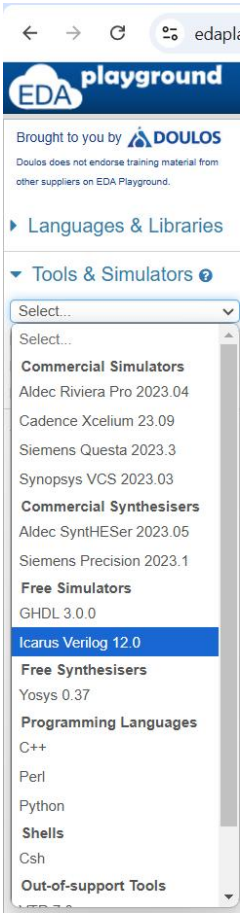
testbench.sv

```
module sillyfunction_tb();
    logic a, b , c;
    logic y;
    // instanciar o "device under test" dut"
    sillyfunction dut (a, b , c , y);
    // estimular o dispositivo (aplicar as entradas) uma de cada vez
    initial                    // podemos utilizar o bloco procedural initial
       begin
          a = 0; b = 0; c = 0; #10;
          c = 1; #10;
          b = 1; c = 0; #10;
          c = 1; #10;
          a = 1; b = 0; c = 0; #10;
          c = 1; #10;
          b = 1; c = 0; #10;
          c = 1; #10;
       end
endmodule
```

- **Passos para implementação**
    - Utilizar a ferramenta [EDAplayground]



    - Selecionar Tools & Simulators -> Free Simulators -> Icarus Verilog 12.0.



- Copiar os códigos do ***design.sv*** e ***testbench.sv*** para as "janelas" correspondentes.
- Executar a simulação (Run).
- Descrever o resultado obtido (apresentar "prints" e comentar a resposta).

ATIVIDADE 2 (Testbench - Checagem automática (*self checking*)): Considere a descrição de um circuito simples (*design.sv*) e o testbench utilizado para verificar o funcionamento do referido circuito (*testbench.sv*).

design.sv

```
module sillyfunction (input logic a, b , c,
                      output logic y);
    assign y = ~b & ~c | a & ~b;
endmodule
```

testbench.sv

```
module sillyfunction_tb();
    logic a, b , c;
    logic y;
    sillyfunction dut(a, b , c , y ); // instanciar o dut
      initial                         // podemos utilizar o bloco procedural initial
        begin
          a = 0; b = 0; c = 0; #10;
          if (y !== 1) $display("000 failed."); // verificando se a saída está correta
          c = 1; #10;
          if (y !== 0) $display("001 failed.");
          b = 1; c = 0; #10;
          if (y !== 0) $display("010 failed.");
          c = 1; #10;
          if (y !== 0) $display("011 failed.");
          a = 1; b = 0; c = 0; #10;
          if (y !== 1) $display("100 failed.");
          c = 1; #10;
          if (y !== 1) $display("101 failed.");
          b = 1; c = 0; #10;
          if (y !== 0) $display("110 failed.");
          c = 1; #10;
          if (y !== 0) $display("111 failed.");
        end
endmodule
```

- **Passos para implementação**
    - Copiar os códigos do ***design.sv*** e ***testbench.sv*** para as "janelas" correspondentes.
    - Executar a simulação (Run).
    - Descrever o resultado obtido (apresentar "prints" e comentar a resposta)..

ATIVIDADE 3 (**Testbench - Checagem automática (self checking) with testvectors**): Considere a descrição de um circuito simples (design.sv) e o testbench utilizado para verificar o funcionamento do referido circuito (testbench.sv).

example.tv

• contém os vetores (abc_yexpected)

000_1
001_0
010_0
011_0
100_1
101_1
110_0
111_0

design.sv

```
module sillyfunction (input logic a, b , c,
                      output logic y);
  assign y = ~b & ~c | a & ~b;
endmodule
```

testbench.sv

```
module testbench();
    logic clk, reset;
    logic a, b, c, yexpected;
    logic y;
    logic [31:0] vectornum, errors; // bookkeeping variables
    logic [3:0] testvectors[10000:0]; // array of testvectors
    // instantiate device under test
    sillyfunction dut(a, b, c, y);
```

```
    // generate clock
  always // no sensitivity list, so it always executes
    begin
        clk = 1; #5; clk = 0; #5;
    end
    // at start of test, load vectors and pulse reset
    initial
      begin
        $readmemb("example.tv", testvectors);
        vectornum = 0; errors = 0;
        reset = 1; #27; reset = 0;
      end
    // Note: $readmemh reads testvector files written in
    // hexadecimal
    // apply test vectors on rising edge of clk
    always @(posedge clk)
        begin
          #1; {a, b, c, yexpected} = testvectors[vectornum];
        end
    // check results on falling edge of clk
    always @(negedge clk)
       if (~reset) begin // skip during reset
                    if (y !== yexpected) begin
                    $display("Error: inputs = %b", {a, b, c});
                    $display(" outputs = %b (%b expected)",y,yexpected);
                    errors = errors + 1;
                  end
      // Note: to print in hexadecimal, use %h. For example,
      // $display("Error: inputs = %h", {a, b, c});
      // increment array index and read next testvector
      vectornum = vectornum + 1;
      if (testvectors[vectornum] === 4'bx) begin
          $display("%d tests completed with %d errors",vectornum, errors);
          $stop;
      end
  end
endmodule
// === and !== can compare values that are 1, 0, x, or z.
```

- **Passos para implementação**

  - Copiar os códigos do ***design.sv*** e ***testbench.sv*** (e ***example.tv***) para as "janelas" correspondentes.
  - Executar a simulação (Run).
  - Descrever o resultado obtido (apresentar "prints" e comentar a resposta).

ATIVIDADE 4 (**Testbench - Checagem automática (self checking)**): Considere a descrição de um circuito simples (design.sv) e o testbench utilizado para verificar o funcionamento do referido circuito (testbench.sv).

design.sv

```
module sillyfunction (input logic a, b,
                      output logic y);
    assign y = a ^ b;
endmodule
```

testbench.sv

```
module sillyfunction_tb();
    logic a, b;
    logic y;
    sillyfunction dut(a, b, y);
    initial
      begin
         a = 0; b = 0; #10;
        if (y !== 1) $display("00 failed.");
        b = 1; #10;
       if (y !== 0) $display("01 failed.");
       a = 1; b = 0; #10;
       if (y !== 0) $display("10 failed.");
       b = 1; #10;
       if (y !== 1) $display("11 failed.");
      end
endmodule
```

- **Passos para implementação**

  - Copiar os códigos do ***design.sv*** e ***testbench.sv*** para as "janelas" correspondentes.
  - Executar a simulação (Run).
  - Descrever o resultado obtido (apresentar "prints" e comentar a resposta)

- Enviar as respostas (descrições dos resultados) de cada atividade em um aruivo (nome do arquivo: "**IntroducaoVerificacao_AtividadePratica01_seunome**").