Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

CSE 412: Selected Topics in Computer Engineering

Assignment 2

Submitted by:

Name: Mohamed Ahmed Mohamed Ahmed
Id: 1701138
Section: 4

Submitted to:

Prof Dr. Ayman Wahba
Eng. Rizk Tawfik

Cairo 2022

Interface Code:

```
interface intf(input clk);
  parameter n = 4;
  //---------- Define inputs and outputs with types -------
  logic winner, loser;
  reg rst, init, gameover;
  reg[1: 0] ctrl, who;
  reg[n-1: 0] val, count;
  reg[3:0] countwin, countlose;

  //------------------ Clocking Block ----------------------
  clocking cb @(posedge clk);
    output gameover, who, ctrl, rst, val, init;
    input winner, loser, count;
  endclocking

  //--------------------- Modports -------------------------
  modport ctr(input clk, rst, ctrl, val, init, output winner, loser, count);
  modport gm(input clk, winner, loser, rst, output gameover, who, countwin,
             countlose);
  modport tb(input clk, countwin, countlose, count, winner, loser, output rst,
             init, gameover, ctrl, who, val);
endinterface
```

Counter Module:

```
module counter(intf.ctr a);
  parameter n = 4;

  //------- Give initial value to winner & loser Signals -----------
  initial begin
    a.winner = 0;
    a.loser = 0;
  end

  //------- Always block with positive edge for clk -------------
  always @(posedge a.clk) begin
    //------ Reset => return counter to initial state -----------
    if (~a.rst) a.count = 0;
    //------------ Init => load a value to counter --------------
    else if (a.init == 1'b1)
      a.count = a.val;

    else begin
      //----------------- CONTROL[1] == 0 (UP) ------------------
      if (a.ctrl[1] == 1'b0)
        a.count = (a.count + a.ctrl[0] + 1) % (1 << n);
      //---------------- CONTROL[1] == 1 (DOWN) -----------------
      else
        a.count = (a.count - a.ctrl[0] - 1) % (1 << n);
    end

    //--------- Disable winner & loser Signals after 1 clk ------
    a.winner = 1'b0;
    a.loser ='b0;
    //---------------- if all ones, winner = 1 ------------------
    if (a.count == (1 << n)-1)
      a.winner = 1'b1;
    //---------------- if all zeros, loser = 1 ------------------
    if (a.count == 0)
      a.loser = 1'b1;
  end
endmodule
```

Game Module:

```
module game(intf.gm b);
  //------------ Give initial values for Signals --------------
  initial begin
    b.who = 2'b00;
    b.gameover = 0;
    b.countwin = 0;
    b.countlose = 0;
  end

  always @(posedge b.clk) begin
    //--------- Disable who & gameover signal after 1 clk -------
    b.who = 2'b00;
    b.gameover = 1'b0;
     //------ Reset => return countwin & countlose to 0 ----------
    if (~b.rst) begin
      b.countwin = 0;
      b.countlose = 0;
    end
    else begin
      //--------------------- winner signal ----------------------
      if (b.winner == 1'b1)
        b.countwin = b.countwin + 1;
      //--------------------- loser signal ------------------------
      else if (b.loser == 1'b1)
        b.countlose = b.countlose + 1;

      //---------- check countlose reaches 15 counts ------------
      if (b.countlose == 15) begin
        b.who = 2'b10;
        b.gameover = 1'b1;
        @(posedge b.clk)
        b.who = 2'b00;
       b.gameover = 1'b0;
        b.countlose = 0;
        b.countwin = 0;
      end
      //---------- check countwin reaches 15 counts ------------
      if (b.countwin == 15) begin
        b.who = 2'b01;
        b.gameover = 1'b1;
        @(posedge b.clk)
        b.who = 2'b00;
       b.gameover = 1'b0;
        b.countlose = 0;
        b.countwin = 0;
      end
    end
  end
endmodule
```

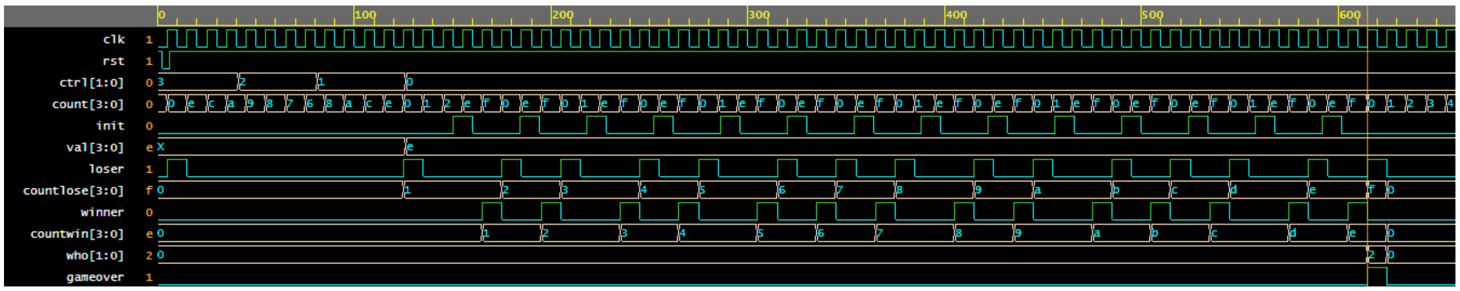
Test Bench Code:

```
program tb(intf.tb c);
  parameter n = 4;

  initial begin
    c.init <= 0;
    c.rst <= 1;
    //----------- Start Counter down with step = 2 ----------
    c.ctrl <= 2'b11;
    //----------- Reset to start counter with zero ----------
    #2 c.rst <= 0;
    #4 c.rst <= 1;
    //---------- change mode to down with step = 1 ----------
    #35 c.ctrl <= 2'b10;
    //----------- change mode to up with step = 2 -----------
    #40 c.ctrl <= 2'b01;
    //----------- change mode to up with step = 1 -----------
    #45 c.ctrl <= 2'b00;
        c.val <= 4'he;

    //-- Repeat loading 14 to fasten reaching the 15 value --
    repeat(14) begin
      #24 c.init <= 1;
      #10 c.init <= 0;
    end
  end

  initial begin
    $dumpfile("waves.vcd");
    $dumpvars;
    #660 $finish;
  end

  property check_15;
    @(posedge c.clk) disable iff (~c.rst)
    ((c.countlose == 15) or (c.countwin == 15)) |->
          ((c.gameover == 1) and ((c.who == 2'b10) or (c.who == 2'b01)));
  endproperty

  property check_15_after;
    @(posedge c.clk) disable iff (~c.rst)
    ((c.countlose == 15) or (c.countwin == 15)) |->
          ##1 ((c.countwin == 0) and (c.countlose == 0) and (c.who == 2'b00));
  endproperty

  property check_loading_value;
    @(posedge c.clk) disable iff (~c.rst)
    (c.init == 1) |-> ##1 (c.count == c.val);
  endproperty

  property valid_who;
    @(posedge c.clk) disable iff (~c.rst)
    c.who != 2'b11;
  endproperty
  property check_rst;
    @(posedge c.clk)
    ~c.rst |-> ##1 (c.count == 0 and c.loser == 1);
  endproperty

  a1: assert property(check_15);
  a2: assert property(check_15_after);
  a3: assert property(check_loading_value);
  a4: assert property(valid_who);
  a5: assert property(check_rst);
endprogram
```

Top Module:

```
module top;
  //--------------- Generating Clock -----------------------
  bit clk;
  parameter CLOCK = 10;
  always #(CLOCK/2) clk = ~clk;

  intf f1(clk);
  counter c(f1.ctr);
  game g(f1.gm);
  tb t(f1.tb);
endmodule
```

Simulation Output:



Eda Playground:

1) Code
2) Screenshot