Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

CSE 412: Selected Topics in Computer Engineering

Assignment 1

Submitted by:

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

Submitted to:

Prof Dr. Ayman Wahba
Eng. Rizk Tawfik

Cairo 2022

Design Code:

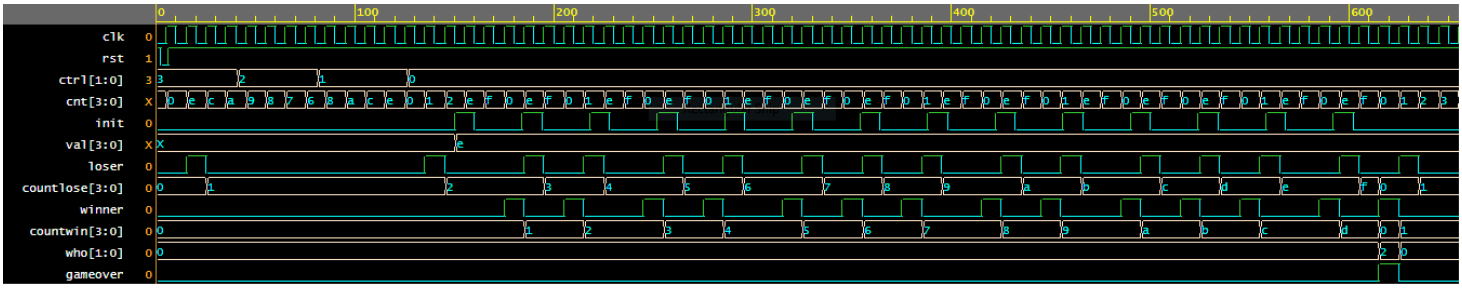
```
module counter(clk, rst, count, ctrl, init, val, winner, loser);
  //------------- Define Parameter for counter size ---------------
  parameter n = 4;
  //-------------- Define inputs and outputs -------------------
  input clk, rst, ctrl, val, init;
  output count, winner, loser;
  //-------------- Define types of Signals --------------------
  wire clk, rst;
  reg init, winner, loser;
  bit[1: 0] ctrl;
  reg[n-1: 0] count, val;

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

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

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

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

```
module game(clk, rst, gameover, who, winner, loser);
  //---------------- Define inputs and outputs ---------------
  output gameover, who;
  input wire winner, loser, clk, rst;
  //---------------- Define types of Signals -----------------
  reg gameover;
  reg[1:0] who;
  reg[3:0] countwin, countlose;
  //------------ Give initial values for Signals -------------
  initial begin
    who = 2'b00;
    gameover = 0;
    countwin = 0;
    countlose = 0;
  end

  always @(posedge clk) begin
    //--------- Disable who & gameover signal after 1 clk -------
    who <= 2'b00;
    gameover <= 1'b0;
    //------ Reset => return countwin & countlose to 0 ----------
    if (~rst) begin
      countwin <= 0;
      countlose <= 0;
    end
    else begin
      //---------------------- winner signal ----------------------
      if (winner == 1'b1)
        countwin <= countwin + 1;
      //---------------------- loser signal -----------------------
      else if (loser == 1'b1)
        countlose <= countlose + 1;
      //---------- check countwin reaches 15 counts ------------
      if (countwin == 15) begin
        countlose <= 0;
        countwin <= 0;
        who <= 2'b10;
        gameover <= 1'b1;
      end
      //---------- check countlose reaches 15 counts ------------
      if (countlose == 15) begin
        countlose <= 0;
        countwin <= 0;
        who <= 2'b01;
        gameover <= 1'b1;
      end
    end
  end
endmodule
```

Test Bench Code:

```
module tb(clk, rst, cnt, ctrl, init, val, winner, loser, gameover, who);
  //------- Define Parameter for counter & clock size ---------
  parameter CLOCK = 10;
  parameter n = 4;
  //---------- Define inputs and outputs with types ----------
  input wire[n-1: 0] cnt;
  input winner, loser;
  output reg clk, rst, init, gameover;
  output reg[1: 0] ctrl, who;
  output reg[n-1: 0] val;

  //--------------------- Generate Clock ----------------------
  always begin
    #(CLOCK / 2) clk = ~clk;
  end

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

    //-- Repeat loading 14 to fasten reaching the 15 value --
    repeat(14) begin
      #24 val = 4'he;
          init = 1;
      #10 init = 0;
    end
  end

  counter c1(clk, rst, cnt, ctrl, init, val, winner, loser);
  game g1(clk, rst, gameover, who, winner, loser);

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

Simulation Output:



Questions Answers:

1) **Does the design need a clock?**
   - Yes, to change the state of counter depending on the control signal and the previous counter value with each positive edge of clock.

2) **Does the design need an asynchronous reset or synchronous reset?**
   - It needs reset to give initial value for counter.
   - I don't find any reason to prefer synchronous or asynchronous so I choose synchronous reset.

Code on Playground:

- Code
- Simulation Output