Logo

Description automatically generated

**Ain Shams University Faculty of Engineering**

**Course Code: CSE 412**

**Course Name: Digital VerificationAssignment 1**

**Submitted by:**

Ahmed Mohamed Ahmed Abd El-Hamed Hassan

**Submitted to:**

Dr. Ayman Wahba

Code: 1700157

Section: 1

Table of Contents

[Requirements: 3](#_Toc100699098)

[Design and Code: 4](#_Toc100699099)

[Counter Module Code: 4](#_Toc100699100)

[Game Status Module Code: 5](#_Toc100699101)

[Test bench: 6](#_Toc100699102)

[Output scenarios: 9](#_Toc100699103)

[First Scenario: 9](#_Toc100699104)

[Second Scenario: 9](#_Toc100699105)

[Third Scenario: 10](#_Toc100699106)

[Forth Scenario: 10](#_Toc100699107)

[Fifth Scenario: 11](#_Toc100699108)

[Sixth Scenario: 11](#_Toc100699109)

[Seventh Scenario: 12](#_Toc100699110)

[Eighth Scenario: 12](#_Toc100699111)

[Nineth Scenario: 13](#_Toc100699112)

[Tenth Scenario: 13](#_Toc100699113)

[Eleventh Scenario: 14](#_Toc100699114)

[Twelfth Scenario: 14](#_Toc100699115)

[Thirteenth Scenario: 15](#_Toc100699116)

[Fourteenth Scenario: 15](#_Toc100699117)

# Requirements:

You have a multi-mode counter. It can count up and down by ones and by twos

There is a two-bit control bus input indicating which one of the four modes is active.

* 00 count up by 1
* 01 count up by 2
* 10 count down 1
* 11 count down 2

You also have an initial value input and a control signal called INIT. When INIT is

logic 1, parallelly load that initial value into the multi-mode counter.

Whenever the count is equal to all zeros, set a signal called LOSER high. When the

count is all ones, set a signal called WINNER high. In either case, the set signal

should remain high for only one cycle.

With a pair of plain binary counters, count the number of times WINNER and LOSER

goes high. When one of them reaches 15, set an output called GAMEOVER high.

If the game is over because LOSER got to 15 first, set a two-bit output called WHO to

2’b01. If the game is over because WINNER got to 15 first, set WHO to 2’b10. WHO

should start at 2’b00 and return to it after each game over.

Then synchronously clear all the counters and start over.

# Design and Code:

The System consists of two main modules and test bench module.

Main modules:

* Game Status module
* Counter module

## Counter Module Code:

|  |
| --- |
| module counter(      //===============      // Output Ports      //===============      output reg [3:0] count\_reg,      // Counter register      //===============      // Input Ports      //===============      input clk,                      // clock      input reset,                    // reset      input Init,                     // initialize   (1: initialize, 0: normal operation)      input [3:0] load,               // load value  (for counter initialization)      input [1:0] control             // control     (0: count up by 1, 1: count up by 2, 2: count down by 1, 3: count down by 2)      );    always @(posedge clk) begin      if (reset) begin        count\_reg = 0;                            // reset counter      end else begin          //==============================          // Initialization          //==============================          if (Init) begin              count\_reg = load;                    // initialize counter          end          //==============================          // Counting          //==============================          else begin              case(control)                         // Check the Control signal              2'b00: count\_reg = count\_reg + 1;     //  00 count up by 1              2'b01: count\_reg = count\_reg - 1;     //  01 count up by 2              2'b10: count\_reg = count\_reg + 2;     //  10 count down by 1              2'b11: count\_reg = count\_reg - 2;     //  11 count down by 2              endcase          end      end    end  endmodule |

## Game Status Module Code:

|  |
| --- |
| module Game\_State#(      //==============================      // Top level block parameters      //==============================      parameter COUNTER\_SIZE = 4          // number of bits in counter      )(      //===============      // Output Ports      //===============      output reg [1:0] who,               // who is the winner      output reg los,                     // loser signal when counter is all zeros      output reg win,                     // winner signal when counter is all ones      output reg gameover,         // gameover signal when loser or winner counters reaches 15      //===============      // Input Ports      //===============      input clk,                          // clock      input reset,                        // reset      input [1:0] control,                // control signal      input INIT,                         // initialization signal      input [COUNTER\_SIZE-1:0] i\_value    // initialization value      );      //==============================      // Signals      //==============================      wire start\_over = reset | gameover; // start over signal     (1: start over and reset all reg and modules, 0: normal operation)      //==============================      // Local registers      //==============================      reg [COUNTER\_SIZE-1:0] count\_reg;   // counter register (read-only)      reg [3:0]wins, losses;              // winner and loser counters      //==============================      // Instantiate Counter module      //==============================      counter c1(.clk(clk), .reset(start\_over), .Init(INIT), .load(i\_value), .control(control), .count\_reg(count\_reg));      always@(posedge clk) begin          // Reset Block          if (start\_over) begin              who = 0;                    // reset Who register              los = 0;                    // release Loser signal              win = 0;                    // release Winner signal              gameover <= 0;              // release Gameover signal              wins = 0;                   // reset Winner counter              losses = 0;                 // reset Loser counter          end          //==============================          // Initialization          //==============================          else if(INIT) begin              who = 0;                    // reset Who register              los = 0;                    // release Loser signal              win = 0;                    // release Winner signal              wins = 0;                   // reset Winner counter              losses = 0;                 // reset Loser counter  gameover <= 0;              // release Gameover signal          end          // Normal Operation          else begin              if (count\_reg == 15) begin                  win = 1;                // set Winner signal                  los = 0;                // release Loser signal                  wins = wins + 1;        // increment winner counter              end else if(count\_reg == 0) begin                  win = 0;                // release Winner signal                  los = 1;                // set Loser signal                  losses = losses + 1;    // increment loser counter              end              else begin                  win = 0;                // release Winner signal                  los = 0;                // release Loser signal              end              if (losses == 15) begin                  who = 1;                // Who with 01 to indicates Loser                  gameover <= 1;          // set Gameover signal              end              if (wins == 15) begin                  who = 2;                // Who with 10 to indicates Winner                  gameover <= 1;          // set Gameover signal              end          end      end  endmodule |
|  |

## Test bench:

|  |
| --- |
| module Game\_State\_testbench  #(      parameter CLOCK = 1,         // clock period      parameter COUNTER\_SIZE = 4   // number of bits in counter      )(      //===============      // Output Ports      //===============      output reg clk,                          // clock      output reg rst\_l,                        // reset      output reg [1:0] control,                // control signal      output reg [COUNTER\_SIZE-1:0] i\_value,   // initialization value      output reg INIT,                         // initialization signal      //===============      // Input Ports      //===============      input wire [1:0] who,                    // who is the winner      input wire los,                      // loser signal when counter is all zeros      input wire win,                       // winner signal when counter is all ones      input wire gameover // gameover signal when loser or winner counters reaches 15      );      //==================      // Local Variables      //==================      int Senario\_NUM;                        // number of senarios      //==============================      // Instantiate the game module      //==============================      Game\_State g1(          .clk(clk),          .reset(rst\_l),          .control(control),          .i\_value(i\_value),          .INIT(INIT),          .who(who),          .los(los),          .win(win),          .gameover(gameover)      );      //==============================      // Create Counter      //==============================      always begin          #CLOCK clk = ~clk;     // create clk works forever      end      //==============================      // Initial Block of Testbench      //==============================      initial begin          Senario\_NUM = 0;            // initialize senario number          clk = 1;                    // start the clock          //===========================================          // For Control Signal = 0 (Count up by 1)          // Senario 1: set initial value to 0          // Senario 2: set initial value to 1          // Senario 3: set initial value to 15          //===========================================          // For Control Signal = 2 (Count down by 1)          // Senario 4: set initial value to 0          // Senario 5: set initial value to 1          // Senario 6: set initial value to 15          //===========================================          for (int cont = 0; cont < 3; cont = cont + 2) begin              for (int i\_v = 0; i\_v < 3; i\_v = i\_v + 1) begin                  rst\_l = 1;                  // reset all registers                  control = cont;             // set control signal                  if(i\_v == 2) i\_value = 15;  // set initial value to 15                  else i\_value = i\_v;         // set initial value to 0 or 1                  INIT = 0;                   // release initialization signal                  #1                          // wait for one clock cycle                  rst\_l = 0;                  // release reset                  INIT = 1;                   // set initialization signal                  #2                          // wait for two clock cycles                  INIT = 0;                   // release initialization signal                  #481                        // wait for 481 clock cycles                  rst\_l = 1;                  // reset all registers              end          end          //===========================================          // For Control Signal = 1 (Count up by 2)          // Senario 7: set initial value to 0          // Senario 8: set initial value to 1          // Senario 9: set initial value to 2          // Senario 10: set initial value to 15          //===========================================          // For Control Signal = 3 (Count down by 2)          // Senario 11: set initial value to 0          // Senario 12: set initial value to 1          // Senario 13: set initial value to 2          // Senario 14: set initial value to 15          //===========================================          for (int cont = 1; cont < 4; cont = cont + 2) begin              for (int i\_v = 0; i\_v < 4; i\_v = i\_v + 1) begin                  rst\_l = 1;                      // reset all registers                  control = cont;                 // set control signal                  if(i\_v == 3) i\_value = 15;      // set initial value to 15                  else i\_value = i\_v;             // set initial value to 0, 1, or 2                  INIT = 0;                       // release initialization signal                  #1                              // wait for one clock cycle                  rst\_l = 0;                      // release reset                  INIT = 1;                       // set initialization signal                  #2                              // wait for two clock cycles                  INIT = 0;                       // release initialization signal                  #251                            // wait for 251 clock cycles                  rst\_l = 1;                      // reset all registers              end          end      end      //=============================================      // Dump variables to view them in the waveform      //=============================================      initial begin          $dumpfile("wave.vcd");          $dumpvars;          #5000 $finish;      end      //=============================================      // Print Outputs for Each Senario      //=============================================      always@(posedge gameover)begin          if(who == 2)              $display("Senario Num = %0d -------WINNER", Senario\_NUM);          else              $display("Senario Num = %0d -------LOSER", Senario\_NUM);          Senario\_NUM = Senario\_NUM +1;    end  endmodule |

# Output scenarios:

## First Scenario:

Control Signal = 2’b00 (count up by 1) initial value = 4’b0000

A picture containing graphical user interface

Description automatically generated

Figure 1

Graphical user interface

Description automatically generated

Figure 2

As we started from Zero, loser counter will be ahead from the winner counter by one.

So, the output signal WHO will be 2’b01 indicating that game over happened because of Loser.

As shown in (Figure 2) all signal is cleared to initial value after game-over is signaled.

## Second Scenario:

Control Signal = 2’b00 (count up by 1) initial value = 4’b0001

A picture containing chart

Description automatically generated

Figure 3

Graphical user interface

Description automatically generated

Figure 4

## Third Scenario:

Control Signal = 2’b00 (count up by 1) initial value = 4’b1111

Chart

Description automatically generated

Figure 5

Graphical user interface

Description automatically generated

Figure 6

As we started from num between 1 to 15, winner counter will be ahead from the loser counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Winner.

As shown in (Figures 4,6) all signal is cleared to initial value after game-over is signaled.

## Forth Scenario:

Control Signal = 2’b10 (counting down by 1) initial value = 4’b0000

A picture containing chart

Description automatically generated

Figure 7

A picture containing timeline

Description automatically generated

Figure 8

## Fifth Scenario:

Control Signal = 2’b10 (counting down by 1) initial value = 4’b0001

A picture containing timeline

Description automatically generated

Figure 9

Timeline

Description automatically generated with medium confidence

Figure 10

As we started from num between 0 to 14, loser counter will be ahead from the winner counter by one.

So, the output signal WHO will be 2’b01 indicating that game over happened because of Loser.

As shown in (Figure 8,10) all signal is cleared to initial value after game-over is signaled.

## Sixth Scenario:

Control Signal = 2’b10 (counting down by 1) initial value = 4’b1111

Timeline

Description automatically generated with medium confidence

Figure 11

Graphical user interface

Description automatically generated

Figure 12

As we started from 15, winner counter will be ahead from the loser counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Winner.

As shown in (Figure 12) all signal is cleared to initial value after game-over is signaled.

## Seventh Scenario:

Control Signal = 2’b01 (count up by 2) initial value = 4’b0000

A screenshot of a computer

Description automatically generated with medium confidence

Figure 13

A screenshot of a computer

Description automatically generated with medium confidence

Figure 14

## Eighth Scenario:

Control Signal = 2’b01 (count up by 2) initial value = 4’b0010

Timeline

Description automatically generated

Figure 15

A screenshot of a computer

Description automatically generated with medium confidence

Figure 16

As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Loser.

As shown in (Figures 16,14) all signal is cleared to initial value after game-over is signaled.

## Nineth Scenario:

Control Signal = 2’b01 (count up by 2) initial value = 4’b0001

Graphical user interface

Description automatically generated

Figure 17

Graphical user interface

Description automatically generated

Figure 18

## Tenth Scenario:

Control Signal = 2’b01 (count up by 2) initial value = 4’b1111

Timeline

Description automatically generated

Figure 19

Graphical user interface

Description automatically generated

Figure 20

As we started from 1,15(ODD NUMBER), winner counter will be ahead from the loser counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Winner.

As shown in (Figures 18,20) all signal is cleared to initial value after game-over is signaled.

## Eleventh Scenario:

Control Signal = 2’b11 (count down by 2) initial value = 4’b0000

Timeline

Description automatically generated

Figure 21

Graphical user interface

Description automatically generated

Figure 22

## Twelfth Scenario:

Control Signal = 2’b11 (count down by 2) initial value = 4’b0010

Timeline

Description automatically generated with medium confidence

Figure 23

Graphical user interface

Description automatically generated

Figure 24

As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Loser.

As shown in (Figures 22,24) all signal is cleared to initial value after game-over is signaled.

## Thirteenth Scenario:

Control Signal = 2’b11 (count down by 2) initial value = 4’b0001

Timeline

Description automatically generated

Figure 25

Graphical user interface

Description automatically generated

Figure 26

## Fourteenth Scenario:

Control Signal = 2’b11 (count down by 2) initial value = 4’b1111

A screenshot of a computer

Description automatically generated with medium confidence

Figure 27

Graphical user interface

Description automatically generated

Figure 28

As we started from 1,15(ODD NUMBER), winner counter will be ahead from the loser counter by one.

So, the output signal WHO will be 2’b10 indicating that game over happened because of Winner.

As shown in (Figures 26,28) all signal is cleared to initial value after game-over is signaled.