Logo

Description automatically generated

**Ain Shams University Faculty of Engineering**

**Course Code: CSE 412**

**Course Name: Digital Verification  
Assignment 2**

**Submitted by:**

Ahmed Mohamed Ahmed Abd El-Hamed Hassan

**Submitted to:**

Dr. Ayman Wahba

Code: 1700157

Section: 1

Repository of project:

https://github.com/ahmed192a/Digital-Verification-/tree/main/2.GameModule\_ProfessionalTestbench

Table of Contents

[Requirements: 3](#_Toc103202658)

[Design and Code: 4](#_Toc103202659)

[Counter Module Code: 4](#_Toc103202660)

[Game Status Module Code: 5](#_Toc103202661)

[Interface: 7](#_Toc103202662)

[Top Module: 7](#_Toc103202663)

[Test bench: 8](#_Toc103202664)

[Outputs Of Assertion: 10](#_Toc103202665)

[Output scenarios: 12](#_Toc103202666)

[First Scenario: 12](#_Toc103202667)

[Second Scenario: 12](#_Toc103202668)

[Third Scenario: 13](#_Toc103202669)

[Forth Scenario: 13](#_Toc103202670)

[Fifth Scenario: 14](#_Toc103202671)

[Sixth Scenario: 14](#_Toc103202672)

[Seventh Scenario: 15](#_Toc103202673)

[Eighth Scenario: 15](#_Toc103202674)

[Nineth Scenario: 16](#_Toc103202675)

[Tenth Scenario: 16](#_Toc103202676)

[Eleventh Scenario: 17](#_Toc103202677)

[Twelfth Scenario: 17](#_Toc103202678)

[Thirteenth Scenario: 18](#_Toc103202679)

[Fourteenth Scenario: 18](#_Toc103202680)

# 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
* Interface Module
* Program Module
* Top 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      )(      Game\_Interface.dut Signals      );      //==============================      // Signals      //==============================      wire start\_over = Signals.reset | Signals.gameover; // start over signal      //==============================      // 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(Signals.clk), .reset(start\_over), .Init(Signals.INIT), .load(Signals.i\_value), .control(Signals.control), .count\_reg(count\_reg));      always@(posedge Signals.clk) begin          // Reset Block          //start\_over = Signals.reset | Signals.gameover;          if (start\_over) begin              Signals.who <= 0;                    // reset Who register              Signals.los <= 0;                    // release Loser signal              Signals.win <= 0;                    // release Winner signal              Signals.gameover <= 0;              // release Gameover signal              wins = 0;                   // reset Winner counter              losses = 0;                 // reset Loser counter          end          //==============================          // Initialization          //==============================          else if(Signals.INIT) begin              Signals.who <= 0;                    // reset Who register              Signals.los <= 0;                    // release Loser signal              Signals.win <= 0;                    // release Winner signal              wins = 0;                   // reset Winner counter              losses = 0;                 // reset Loser counter              Signals.gameover <= 0;              // release Gameover signal          end          // Normal Operation          else begin              if (count\_reg == 15) begin                  Signals.win <= 1;                // set Winner signal                  Signals.los <= 0;                // release Loser signal                  wins = wins + 1;        // increment winner counter              end else if(count\_reg == 0) begin                  Signals.win <= 0;                // release Winner signal                  Signals.los <= 1;                // set Loser signal                  losses = losses + 1;    // increment loser counter              end              else begin                  Signals.win <= 0;                // release Winner signal                  Signals.los <= 0;                // release Loser signal              end              if (losses == 15) begin                  Signals.who <= 1;                // Who with 01 to indicates Loser                  Signals.gameover <= 1;          // set Gameover signal              end              if (wins == 15) begin                  Signals.who <= 2;                // Who with 10 to indicates Winner                  Signals.gameover <= 1;          // set Gameover signal              end          end      end  endmodule |

## Interface:

|  |
| --- |
| interface Game\_Interface #(      parameter COUNTER\_SIZE = 4      )(      input bit clk      );      bit [1:0] who, control;      bit los, win, gameover, reset, INIT;      bit [COUNTER\_SIZE-1:0] i\_value;      clocking cb @(posedge clk);     // Clocking block          default input #0ns output #1ns;          output reset, control, INIT, i\_value;          input who, los, win, gameover;      endclocking      modport dut     // Port for Device under the Test      (          output gameover, who, los, win,          input clk, reset, control, INIT, i\_value      );      modport tb      // Port for Testbench      (          clocking cb,          output reset      );  endinterface |

## Top Module:

|  |
| --- |
| module top (output bit clk);      initial clk = 1;      always #1 clk = ~clk;      Game\_Interface inter(.clk(clk));      Game\_State\_testbench u0(.Signals(inter.tb));      Game\_State u1(.Signals(inter.dut));      //=============================================      // Dump variables to view them in the waveform      //=============================================      initial begin          $dumpfile("wave.vcd");          $dumpvars;      end  endmodule |

## Test bench:

|  |
| --- |
| program Game\_State\_testbench (      Game\_Interface.tb Signals      );      // wire who ,gameover;      assign who = Signals.cb.who;      assign los = Signals.cb.los;      assign win = Signals.cb.win;      assign gameover = Signals.cb.gameover;      int cont;      int i\_v ;      //==============================      // Initial Block of Testbench      //==============================      initial begin          //===========================================          // For Control Signal = 0 (Count up by 1)          // Scenario 1: set initial value to 0          // Scenario 2: set initial value to 1          // Scenario 3: set initial value to 15          //===========================================          // For Control Signal = 2 (Count down by 1)          // Scenario 4: set initial value to 0          // Scenario 5: set initial value to 1          // Scenario 6: set initial value to 15          //===========================================          for ( cont = 0; cont < 3; cont = cont + 2) begin              for ( i\_v = 0; i\_v < 3; i\_v = i\_v + 1) begin                  $display("\n\n Senario: Control = %0d, initail value = %0d",cont,i\_v);                  Signals.cb.reset <= 1;                   // reset all registers                  Signals.cb.control <= cont;              // set control signal                  if(i\_v == 2) Signals.cb.i\_value <= 15;   // set initial value to 15                  else Signals.cb.i\_value <= i\_v;          // set initial value to 0 or 1                  Signals.cb.INIT <= 0;                    // release initialization signal                  #2                                      // wait for one clock cycle                  Signals.cb.reset <= 0;                   // release reset                  Signals.cb.INIT <= 1;                    // set initialization signal                  #2                                      // wait for two clock cycles                  Signals.cb.INIT <= 0;                    // release initialization signal                  #482                                    // wait for 481 clock cycles                  Signals.cb.reset <= 1;                   // reset all registers              end          end          //===========================================          // For Control Signal = 1 (Count up by 2)          // Scenario 7: set initial value to 0          // Scenario 8: set initial value to 1          // Scenario 9: set initial value to 2          // Scenario 10: set initial value to 15          //===========================================          // For Control Signal = 3 (Count down by 2)          // Scenario 11: set initial value to 0          // Scenario 12: set initial value to 1          // Scenario 13: set initial value to 2          // Scenario 14: set initial value to 15          //===========================================          for ( cont = 1; cont < 4; cont = cont + 2) begin              for ( i\_v = 0; i\_v < 4; i\_v = i\_v + 1) begin                  $display("\n\n Senario: Control = %0d, initail value = %0d",cont,i\_v);                  Signals.cb.reset <= 1;                       // reset all registers                  Signals.cb.control <= cont;                  // set control signal                  if(i\_v == 3) Signals.cb.i\_value <= 15;       // set initial value to 15                  else Signals.cb.i\_value <= i\_v;              // set initial value to 0, 1, or 2                  Signals.cb.INIT <= 0;                        // release initialization signal                  #2                                           // wait for one clock cycle                  Signals.cb.reset <= 0;                       // release reset                  Signals.cb.INIT <= 1;                        // set initialization signal                  #2                                           // wait for two clock cycles                  Signals.cb.INIT <= 0;                        // release initialization signal                  #252                                         // wait for 251 clock cycles                  Signals.cb.reset <= 1;                       // reset all registers              end          end          #20;      end      //===========================      // Properties      //===========================      property signals\_cleared;        @(Signals.cb) disable iff(!($fell(Signals.reset) )) (who==0 || los ==0 || gameover ==0 || win ==0);      endproperty      property winner\_checker;        @(Signals.cb)        if($fell(Signals.reset)) ##[113:241] gameover ==1;      endproperty      //===========================      // Asserions      //===========================      assert\_winner\_checker: assert property(winner\_checker)$display("[%0t] ----- Assertion GameOver passed", $time);      assert\_signals\_cleared: assert property (signals\_cleared) $display("[%0t] ----- Assertion Reseting\_signals passed", $time);  endprogram |

## Outputs Of Assertion:

Text

Description automatically generated

This is the First Scenarios

Control = 0

1. Initial value = 0
2. Initial value = 1
3. Initial value = 15

Text, letter

Description automatically generated

This is the Second Scenarios

Control = 2

1. Initial value = 0
2. Initial value = 1
3. Initial value = 15

Table

Description automatically generated

This is the Third Scenarios

Control = 1

1. Initial value = 0
2. Initial value = 1
3. Initial value = 2
4. Initial value = 15

Table

Description automatically generated

This is the Fourth Scenarios

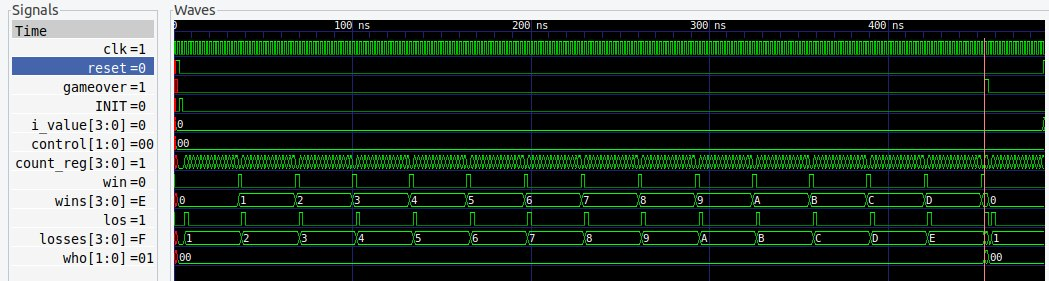
Control = 3

1. Initial value = 0
2. Initial value = 1
3. Initial value = 2
4. Initial value = 15

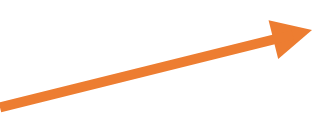
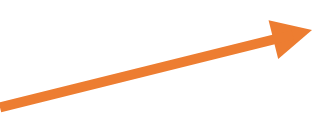
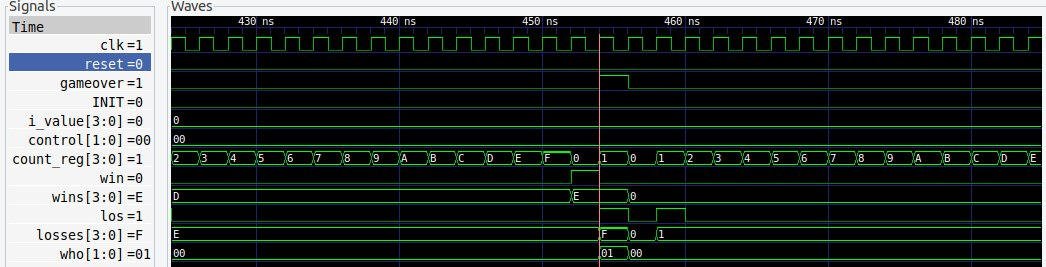
# Output scenarios:

## First Scenario:

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



*Figure 1*



*Figure 2*

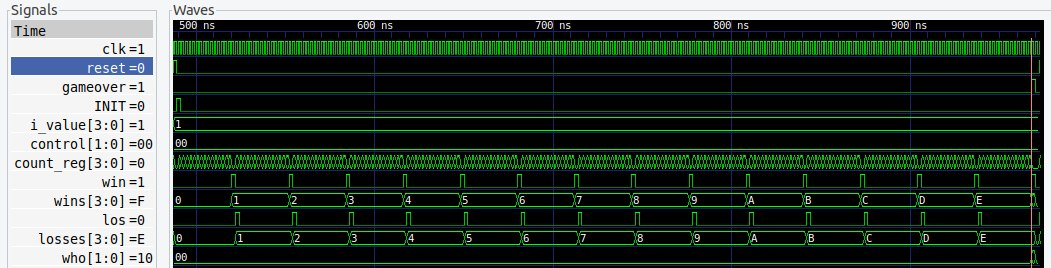
As we started from Zero, loser counter will be ahead from the winner counter by one.

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

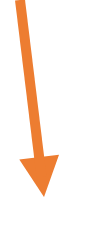
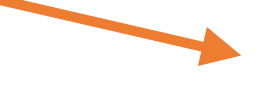
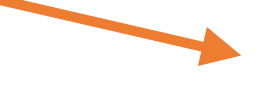
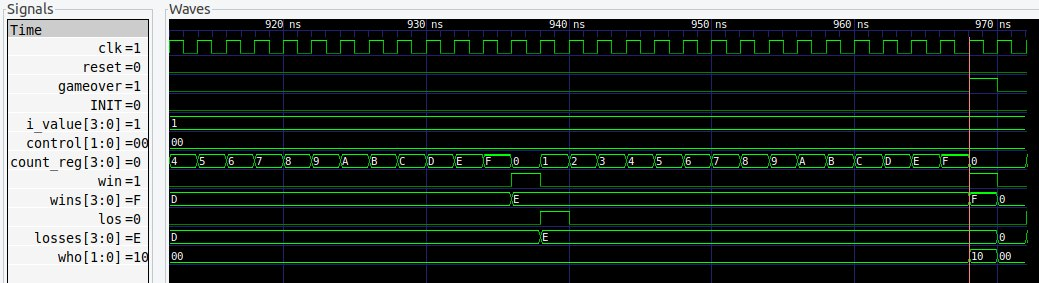
As shown in (Figure 2) all signal is cleared to initial value after game-over is signaled.

## Second Scenario:

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



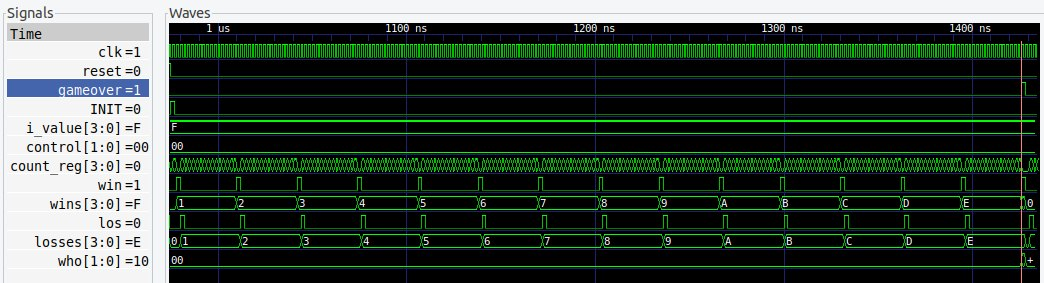
*Figure 3*



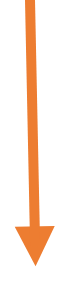
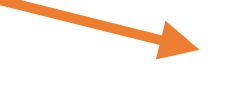
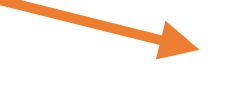
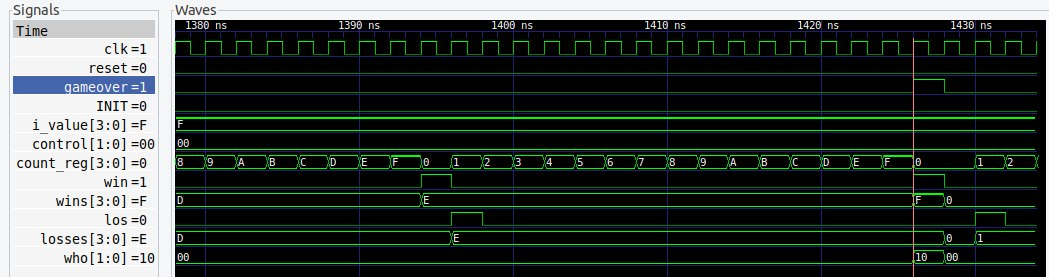
*Figure 4*

## Third Scenario:

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



*Figure 5*



*Figure 6*

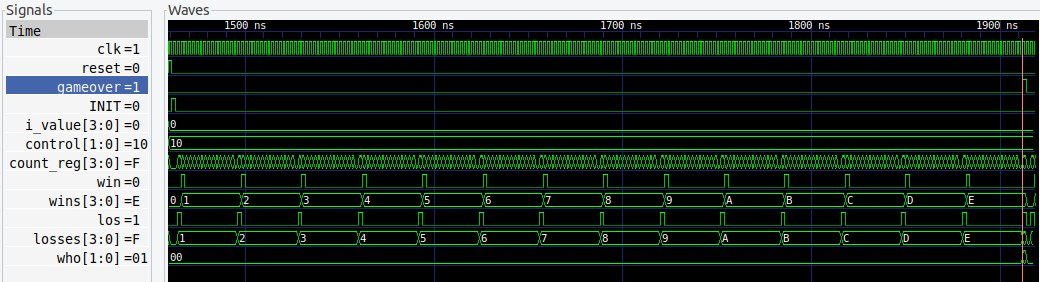
As we started from num between 1 to 15, winner counter will be ahead from the loser counter by one.

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

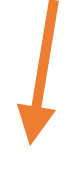
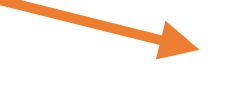
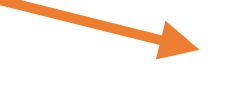
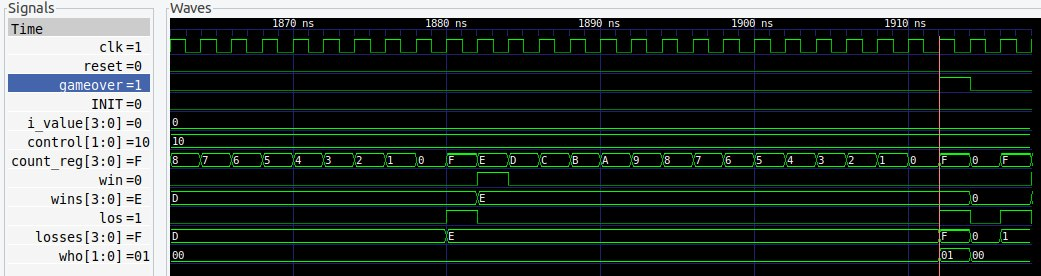
As shown in (Figures 4,6) all signal is cleared to initial value after game-over is signaled.

## Forth Scenario:

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



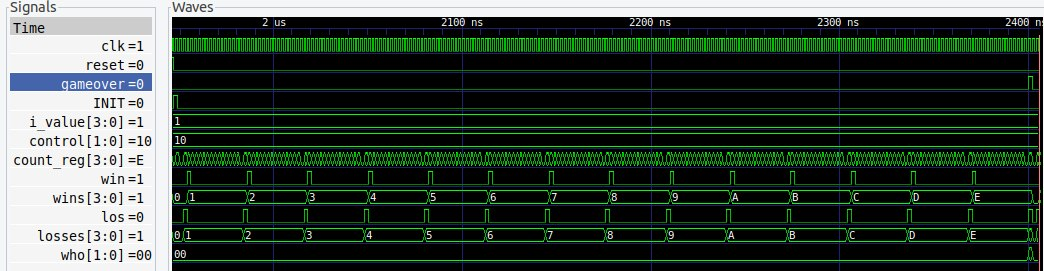
*Figure 7*



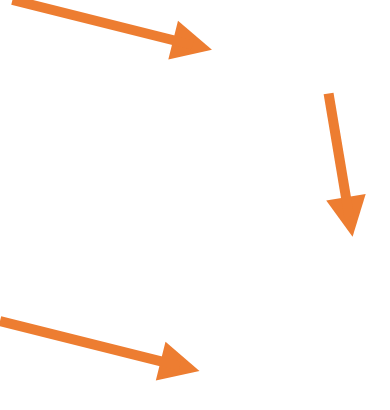
*Figure 8*

## Fifth Scenario:

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



*Figure 9*



*Figure 10*

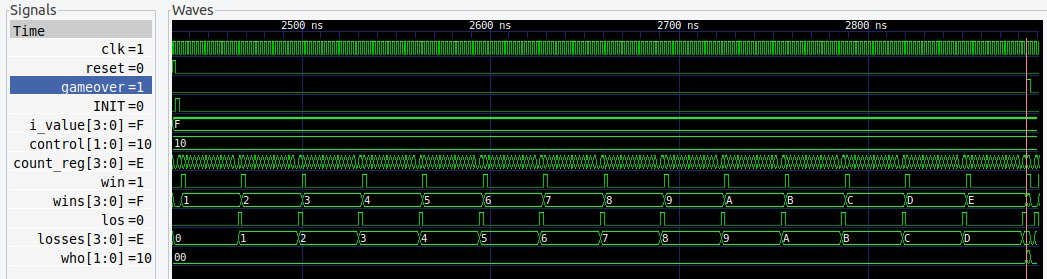
As we started from num between 0 to 14, loser counter will be ahead from the winner counter by one.

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

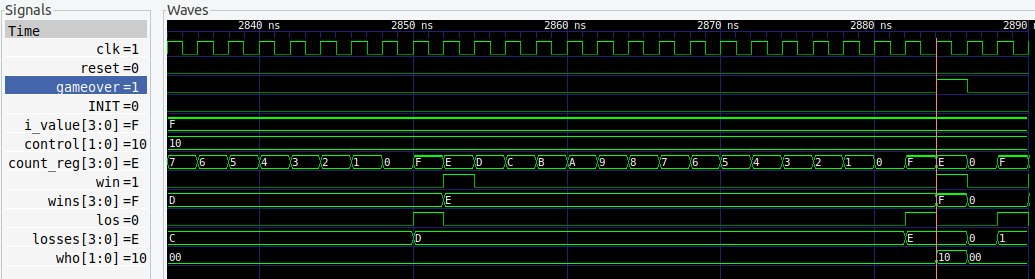
As shown in (Figure 8,10) all signal is cleared to initial value after game-over is signaled.

## Sixth Scenario:

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



*Figure 11*



*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*

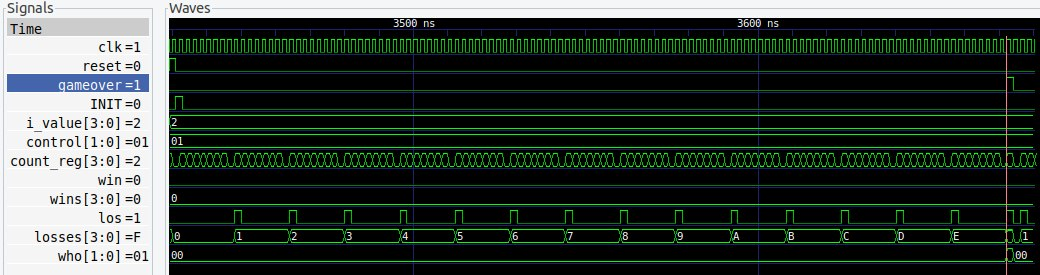
A screenshot of a computer

Description automatically generated with medium confidence

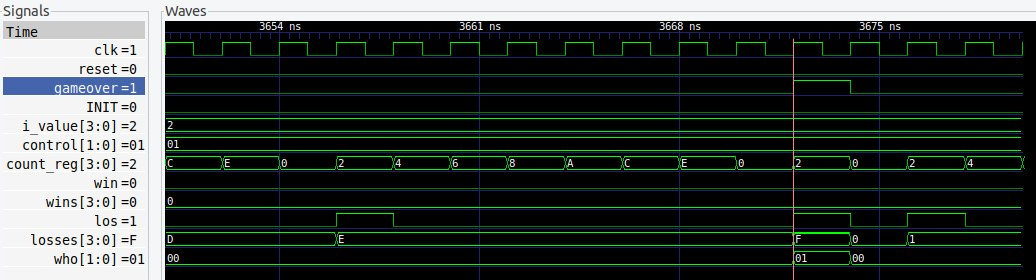
*Figure 14*

## Eighth Scenario:

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



*Figure 15*



*Figure 16*

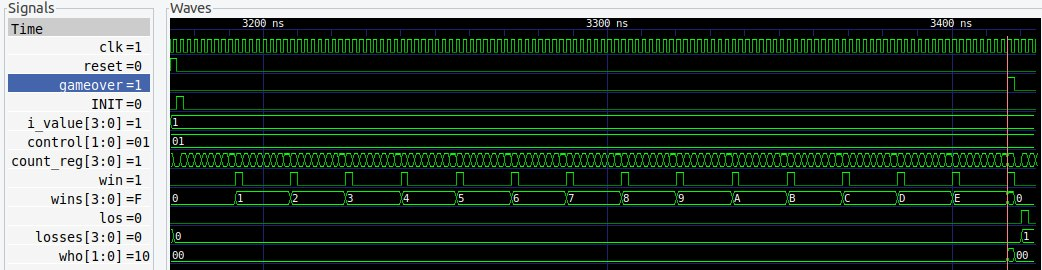
As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

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

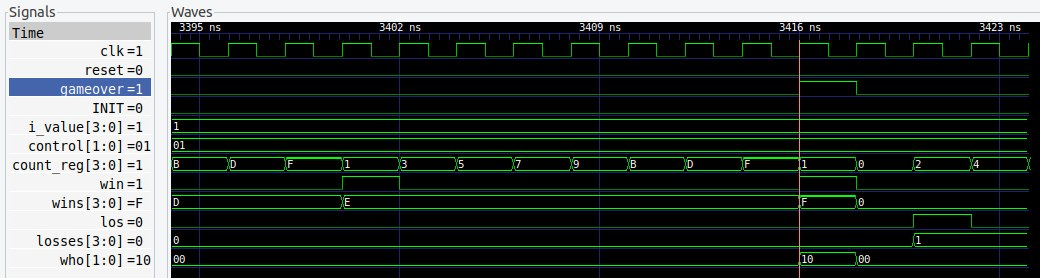
As shown in (Figures 16,14) all signal is cleared to initial value after game-over is signaled.

## Nineth Scenario:

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



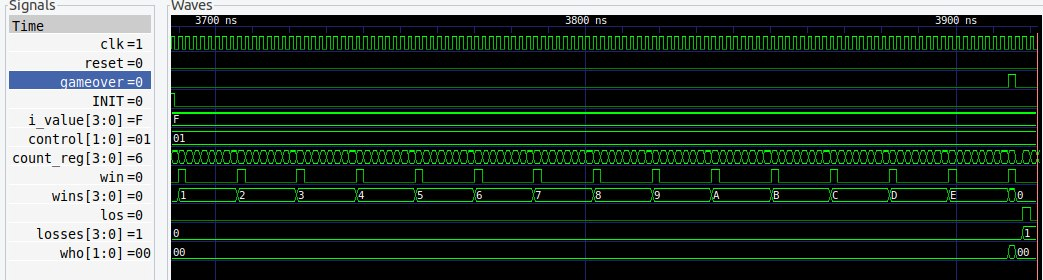
*Figure 17*



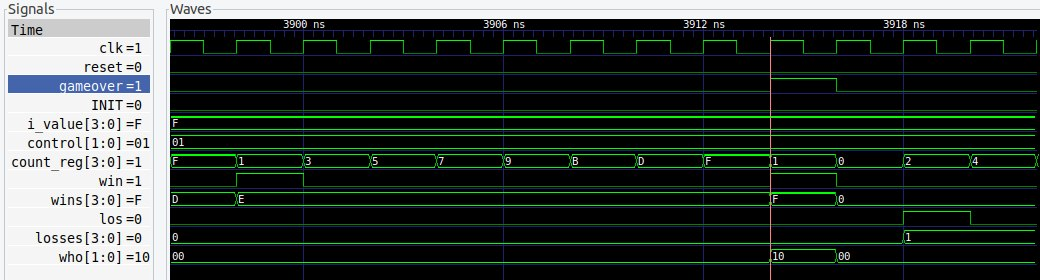
*Figure 18*

## Tenth Scenario:

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



*Figure 19*



*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.

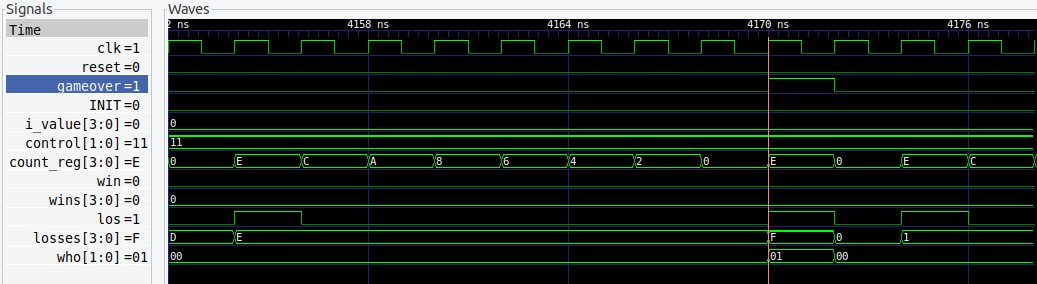
As shown in (Figures 18,20) all signal is cleared to initial value after game-over is signaled.

## Eleventh Scenario:

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



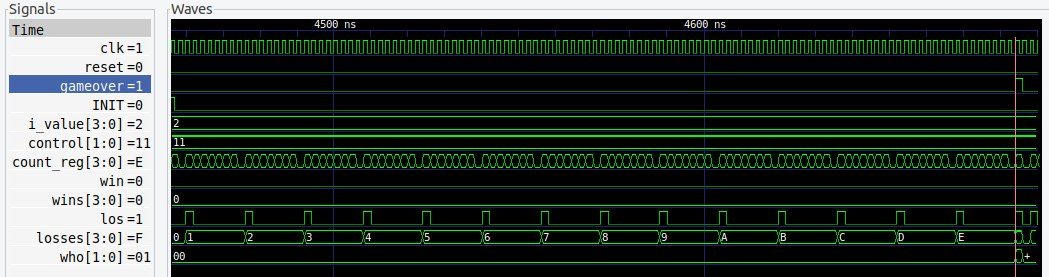
*Figure 21*



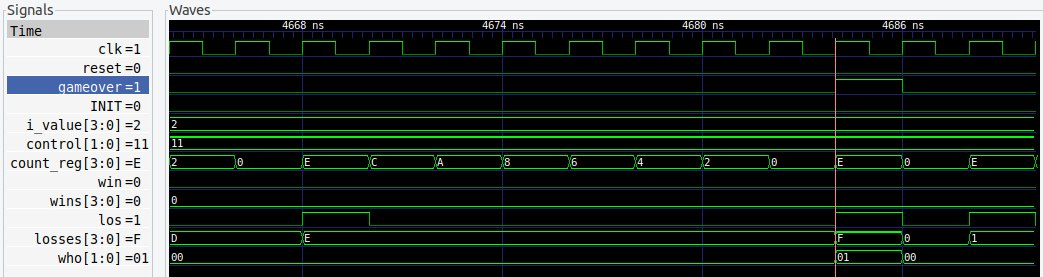
*Figure 22*

## Twelfth Scenario:

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



*Figure 23*



*Figure 24*

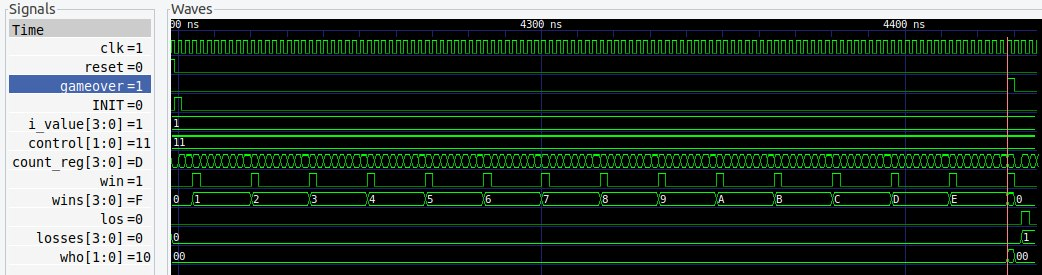
As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

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

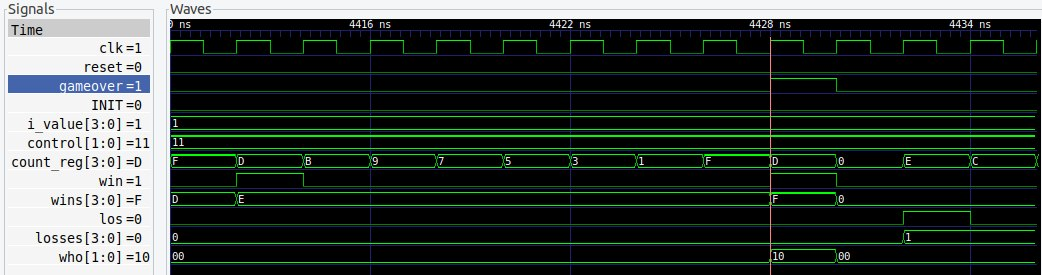
As shown in (Figures 22,24) all signal is cleared to initial value after game-over is signaled.

## Thirteenth Scenario:

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



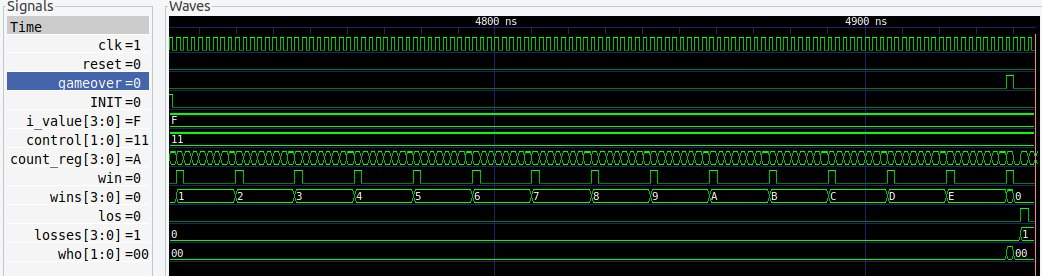
*Figure 25*



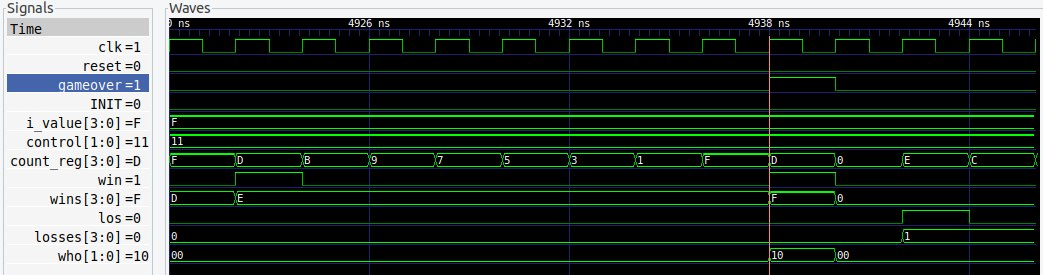
*Figure 26*

## Fourteenth Scenario:

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



*Figure 27*



*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.