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:

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:

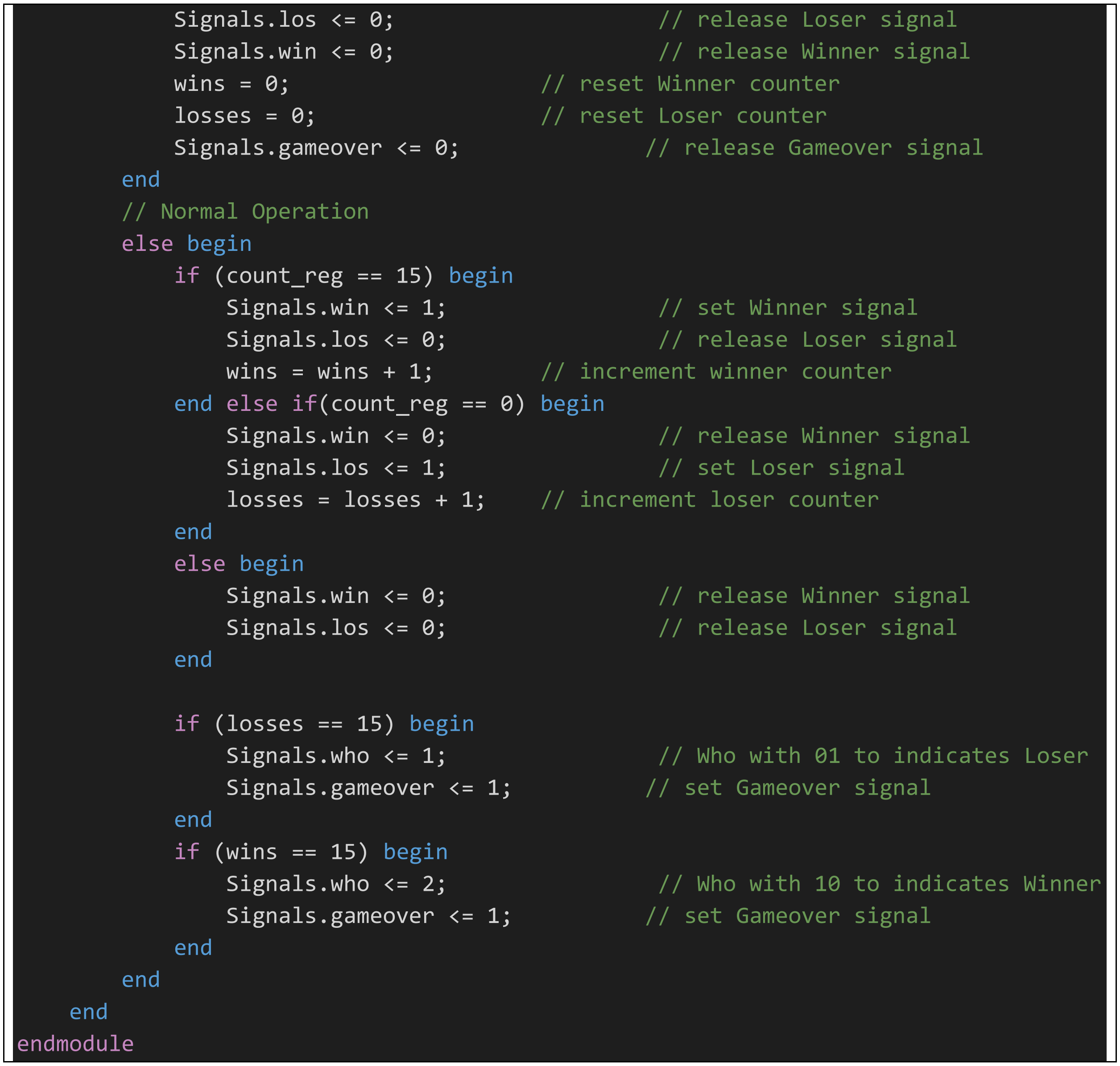
- Game Status Module
- Counter Module
- Interface Module
- Program Module
- Top Module

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:

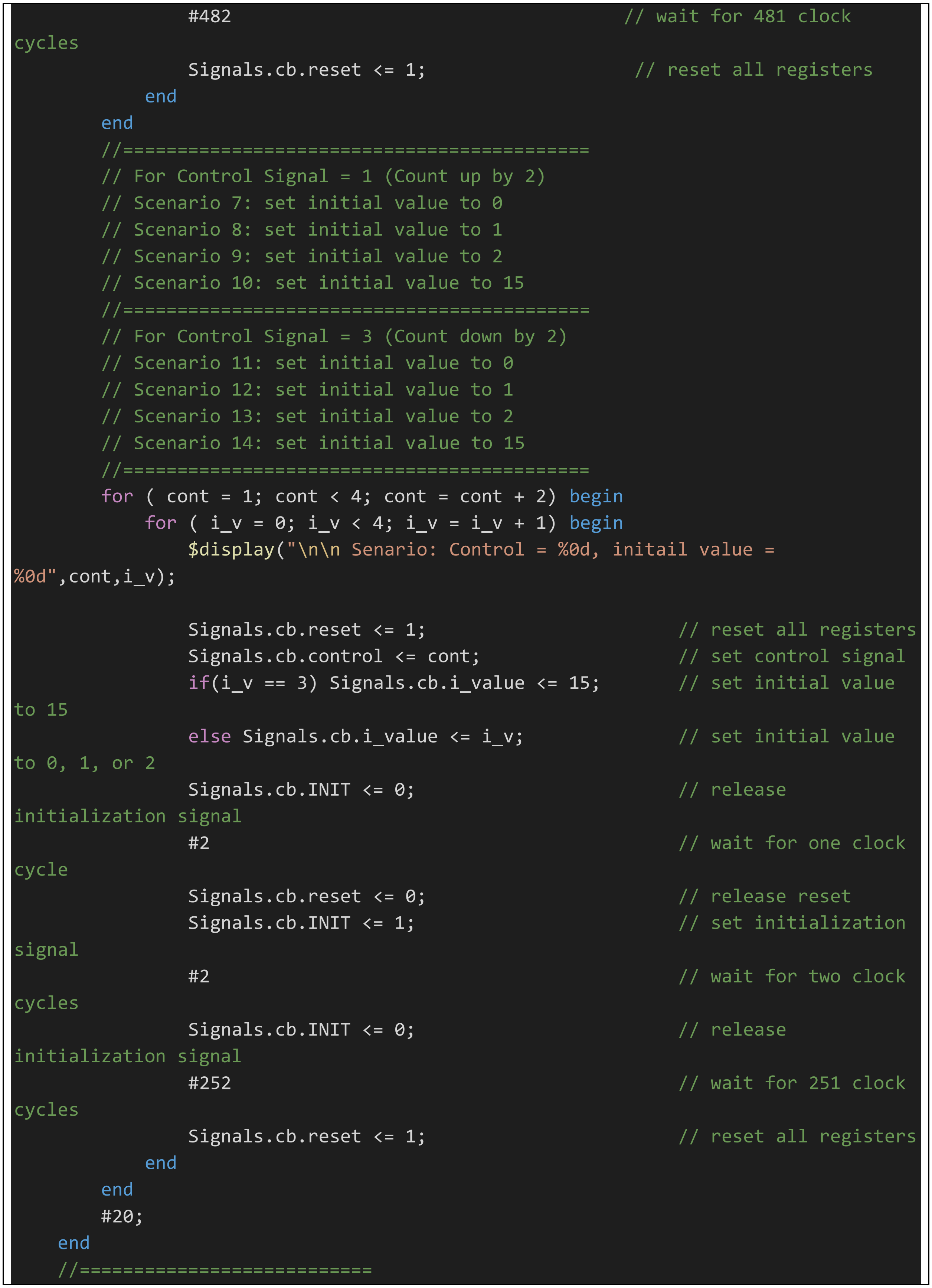
```
interface Game_Interface #(
    parameter COUNTER_SIZE = 4
    )(
    input bit clk
    );
    bit [1:0] who, control;
    bit los, win, gameover, reset, INIT;
    bit [COUNTER_SIZE-1:0] i_value;
    clocking cb @(posedge clk);     // Clocking block
        default input #0ns output #1ns;
        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:

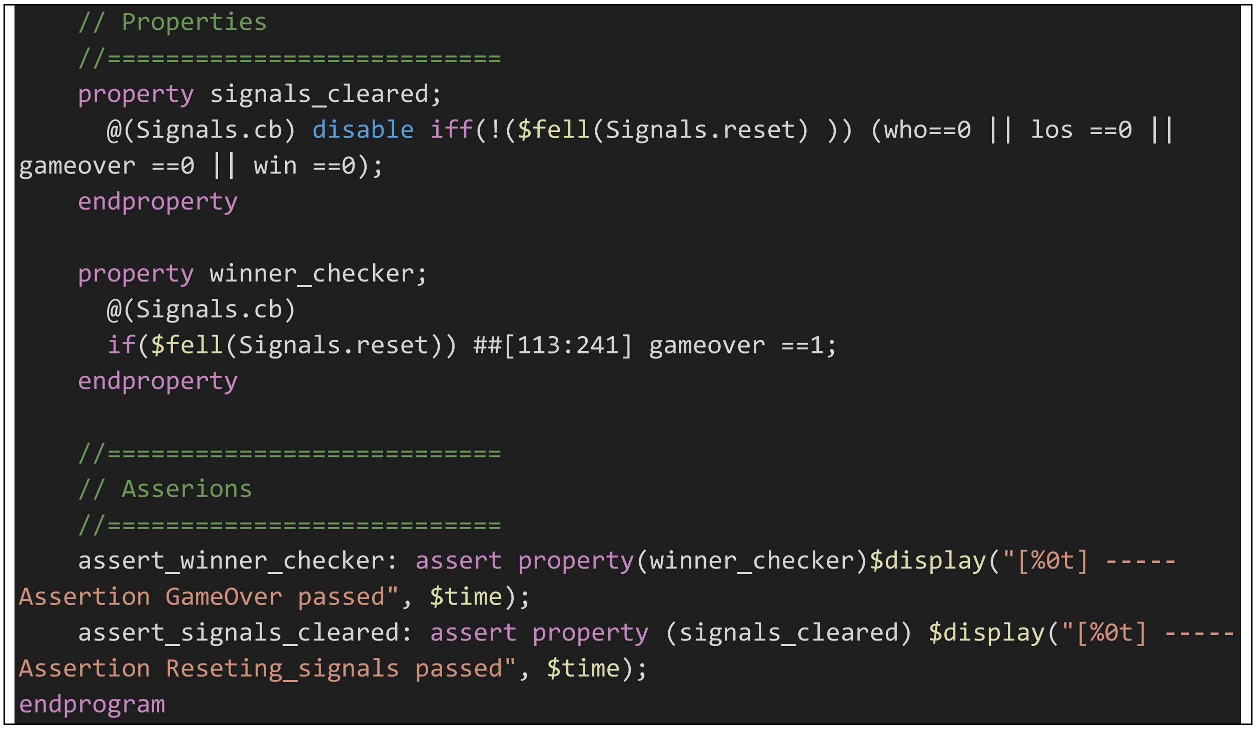
```
module top (output bit clk);
    initial clk = 1;
    always #1 clk = ~clk;
    Game_Interface inter(.clk(clk));
    Game_State_testbench u0(.Signals(inter.tb));
    Game_State u1(.Signals(inter.dut));
    //=============================================
    // Dump variables to view them in the waveform
    //=============================================
    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
    //=============================
```



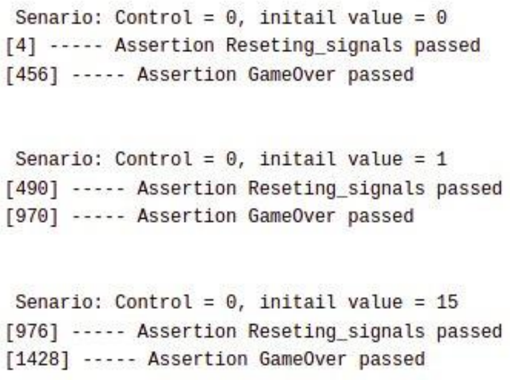
```
    // Properties
    //===========================
    property signals_cleared;
      @(Signals.cb) disable iff(!($fell(Signals.reset) )) (who==0 || los ==0 ||
gameover ==0 || win ==0);
    endproperty

    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:

This is the First Scenarios

Control = 0

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

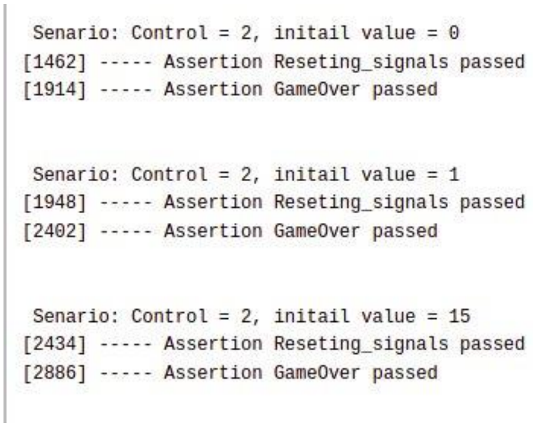


```
 Senario: Control = 0, initail value = 0
[4] ----- Assertion Reseting_signals passed
[456] ----- Assertion GameOver passed

 Senario: Control = 0, initail value = 1
[490] ----- Assertion Reseting_signals passed
[970] ----- Assertion GameOver passed

 Senario: Control = 0, initail value = 15
[976] ----- Assertion Reseting_signals passed
[1428] ----- Assertion GameOver passed
```

This is the Second Scenarios

Control = 2

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

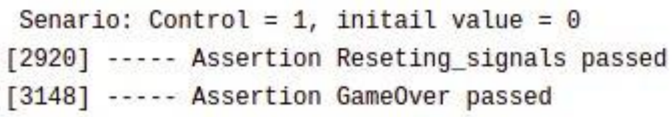


```
 Senario: Control = 2, initail value = 0
[1462] ----- Assertion Reseting_signals passed
[1914] ----- Assertion GameOver passed

 Senario: Control = 2, initail value = 1
[1948] ----- Assertion Reseting_signals passed
[2402] ----- Assertion GameOver passed

 Senario: Control = 2, initail value = 15
[2434] ----- Assertion Reseting_signals passed
[2886] ----- Assertion GameOver passed
```

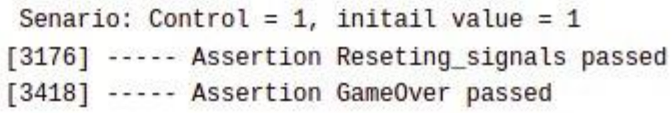
This is the Third Scenarios

Control = 1

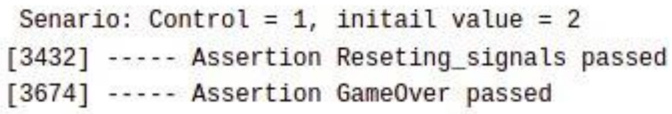
1. Initial value = 0
2. Initial value = 1
3. Initial value = 2
4. Initial value = 15



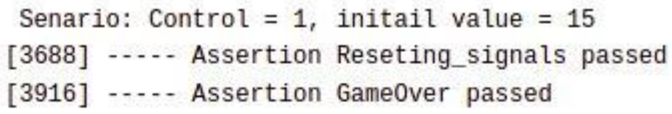
```
 Senario: Control = 1, initail value = 0
[2920] ----- Assertion Reseting_signals passed
[3148] ----- Assertion GameOver passed
```



```
 Senario: Control = 1, initail value = 1
[3176] ----- Assertion Reseting_signals passed
[3418] ----- Assertion GameOver passed
```



```
 Senario: Control = 1, initail value = 2
[3432] ----- Assertion Reseting_signals passed
[3674] ----- Assertion GameOver passed
```

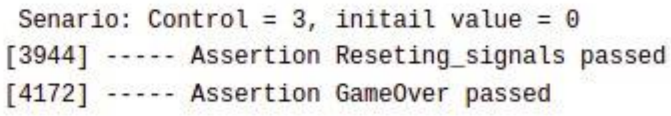


```
 Senario: Control = 1, initail value = 15
[3688] ----- Assertion Reseting_signals passed
[3916] ----- Assertion GameOver passed
```

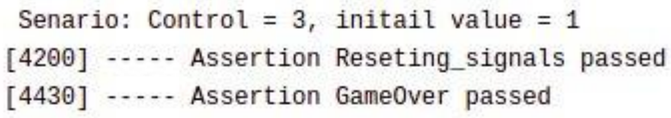
This is the Fourth Scenarios

Control = 3

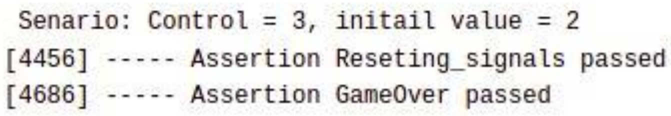
1. Initial value = 0
2. Initial value = 1
3. Initial value = 2
4. Initial value = 15



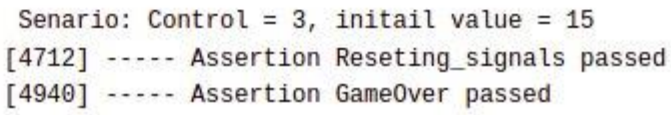
```
 Senario: Control = 3, initail value = 0
[3944] ----- Assertion Reseting_signals passed
[4172] ----- Assertion GameOver passed
```



```
 Senario: Control = 3, initail value = 1
[4200] ----- Assertion Reseting_signals passed
[4430] ----- Assertion GameOver passed
```



```
 Senario: Control = 3, initail value = 2
[4456] ----- Assertion Reseting_signals passed
[4686] ----- Assertion GameOver passed
```

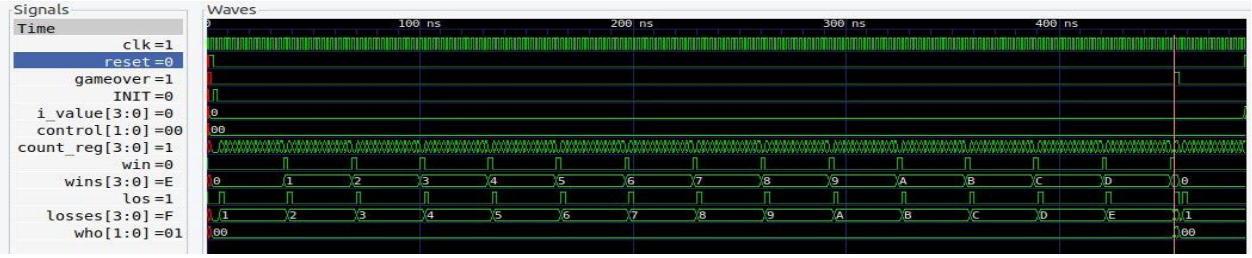


```
 Senario: Control = 3, initail value = 15
[4712] ----- Assertion Reseting_signals passed
[4940] ----- Assertion GameOver passed
```

Output scenarios:

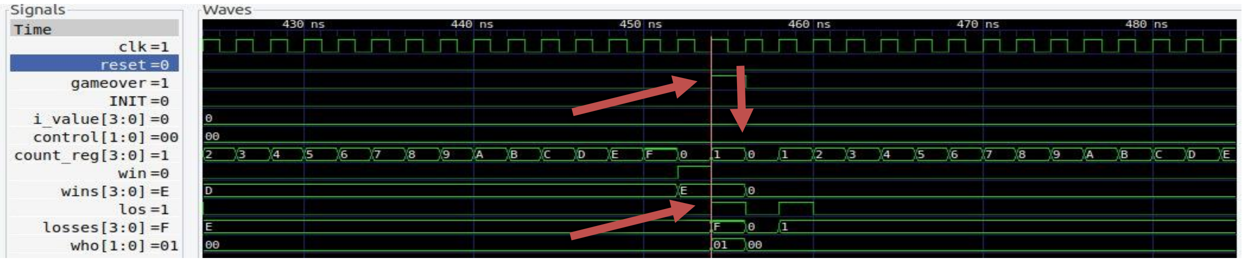
First Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=0
control[1:0]=00
count_reg[3:0]=1
win=0
wins[3:0]=E
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 1



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=0
control[1:0]=00
count_reg[3:0]=1
win=0
wins[3:0]=E
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 2

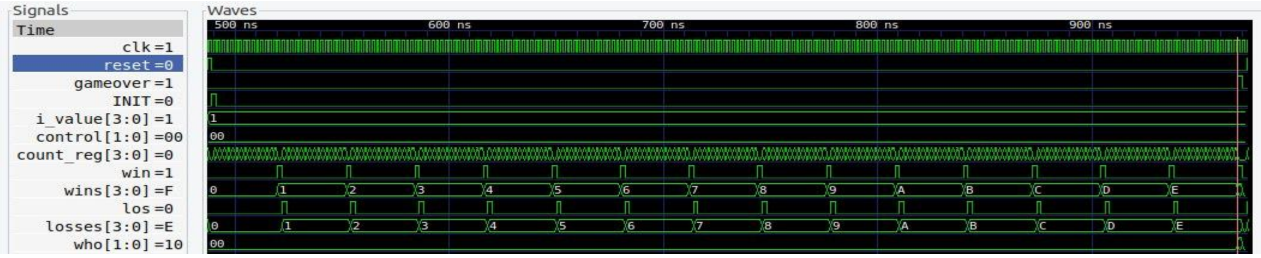
As we started from Zero, loser counter will be ahead from the winner counter by one.

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.

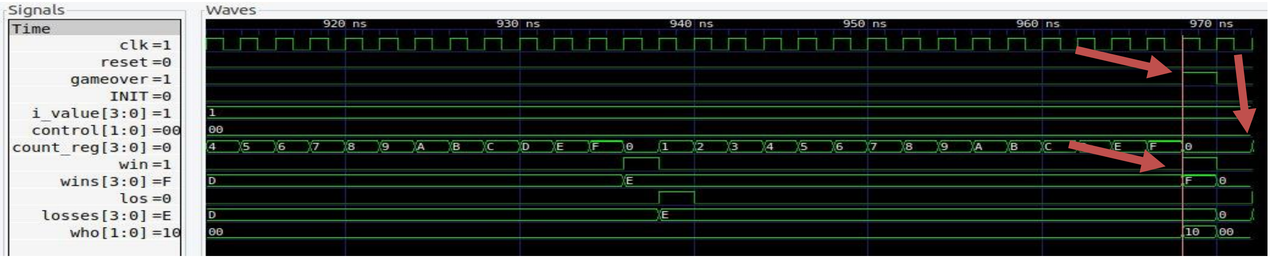
Second Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=1
control[1:0]=00
count_reg[3:0]=0
win=1
wins[3:0]=F
los=0
losses[3:0]=E
who[1:0]=10
```

Figure 3

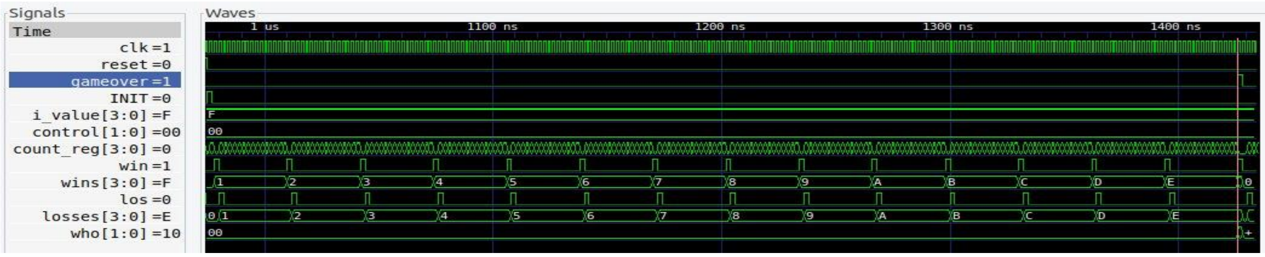


```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=1
control[1:0]=00
count_reg[3:0]=0
win=1
wins[3:0]=F
los=0
losses[3:0]=E
who[1:0]=10
```

Figure 4

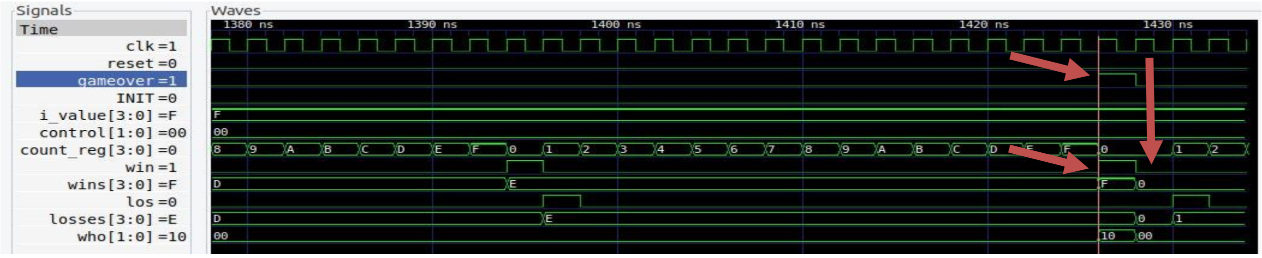
Third Scenario:

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



```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=F
 control[1:0]=00
count_reg[3:0]=0
        win=1
   wins[3:0]=F
        los=0
 losses[3:0]=E
    who[1:0]=10
```

Figure 5



```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=F
 control[1:0]=00
count_reg[3:0]=0
        win=1
   wins[3:0]=F
        los=0
 losses[3:0]=E
    who[1:0]=10
```

Figure 6

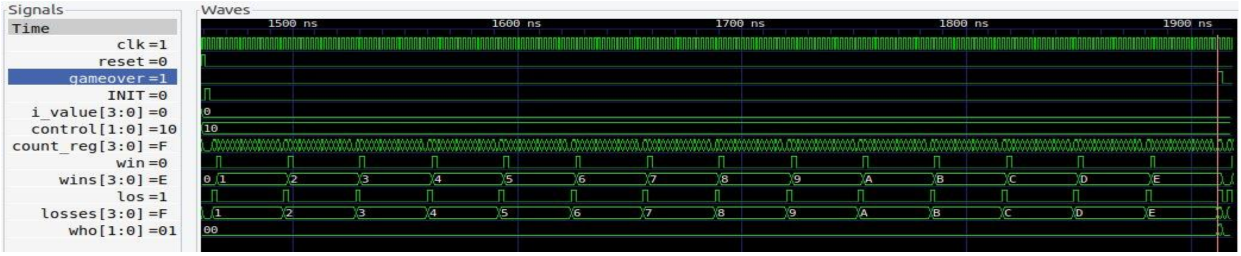
As we started from num between 1 to 15, winner counter will be ahead from the loser counter by one.

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.

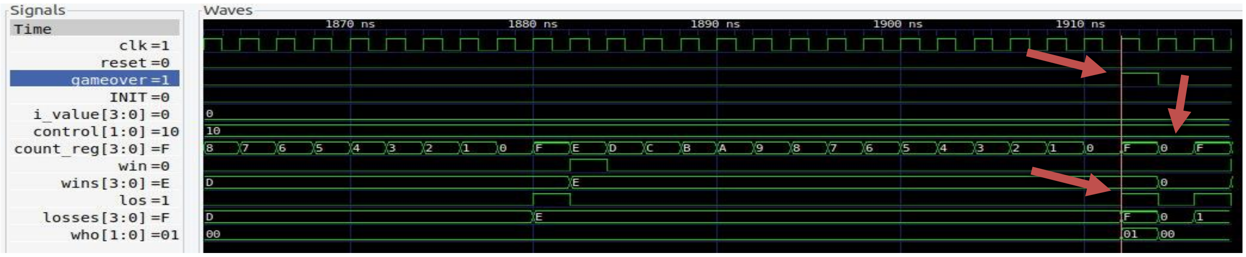
Forth Scenario:

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



```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=0
 control[1:0]=10
count_reg[3:0]=F
        win=0
   wins[3:0]=E
        los=1
 losses[3:0]=F
    who[1:0]=01
```

Figure 7

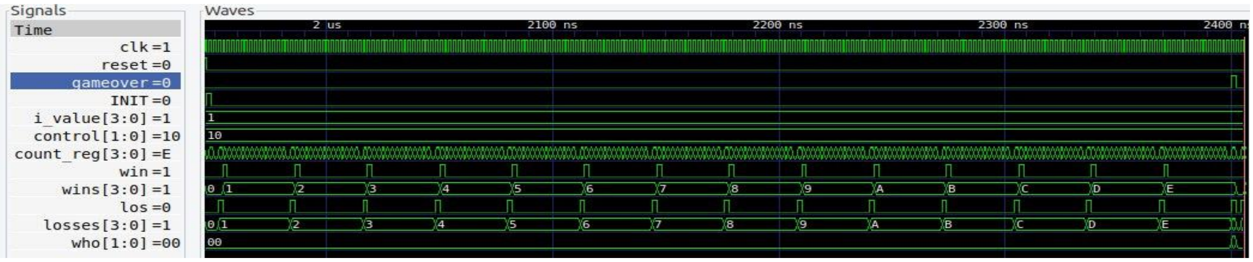


```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=0
 control[1:0]=10
count_reg[3:0]=F
        win=0
   wins[3:0]=E
        los=1
 losses[3:0]=F
    who[1:0]=01
```

Figure 8

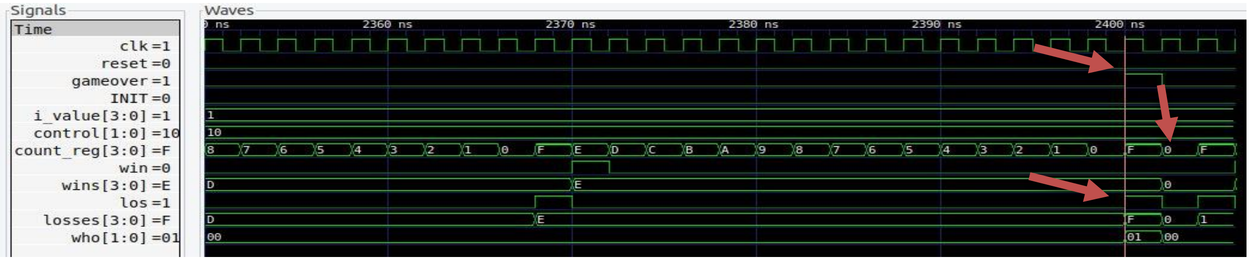
Fifth Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=0
INIT=0
i_value[3:0]=1
control[1:0]=10
count_reg[3:0]=E
win=1
wins[3:0]=1
los=0
losses[3:0]=1
who[1:0]=00
```

Figure 9



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=1
control[1:0]=10
count_reg[3:0]=F
win=0
wins[3:0]=E
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 10

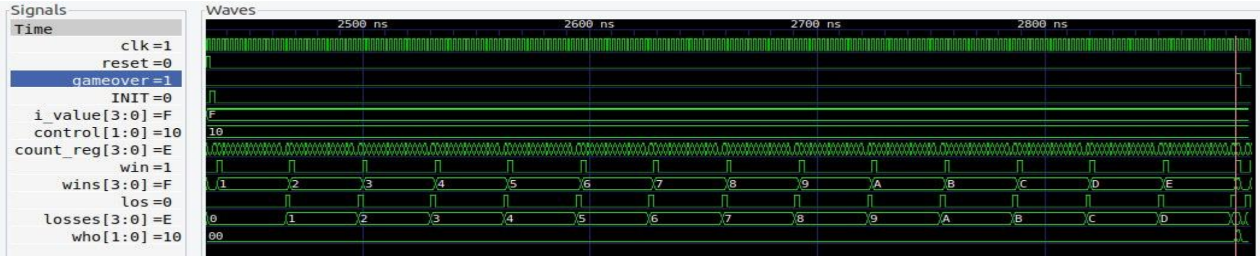
As we started from num between 0 to 14, loser counter will be ahead from the winner counter by one.

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.

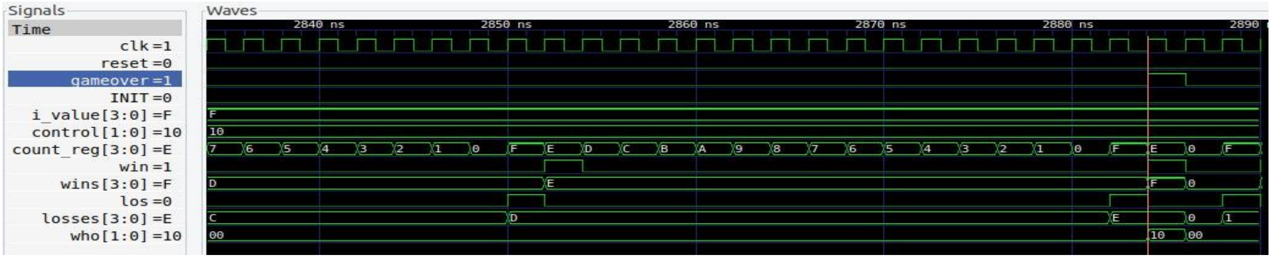
Sixth Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=F
control[1:0]=10
count_reg[3:0]=E
win=1
wins[3:0]=F
los=0
losses[3:0]=E
who[1:0]=10
```

Figure 11



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=F
control[1:0]=10
count_reg[3:0]=E
win=1
wins[3:0]=F
los=0
losses[3:0]=E
who[1:0]=10
```

Figure 12

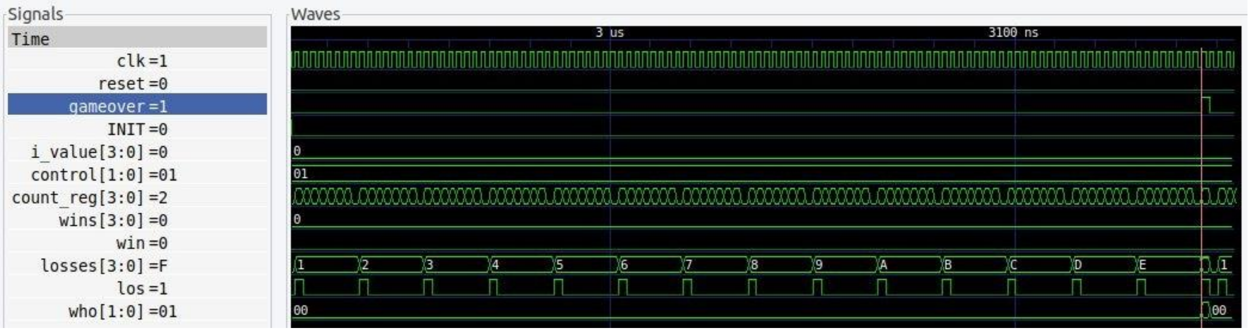
As we started from 15, winner counter will be ahead from the loser counter by one.

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:

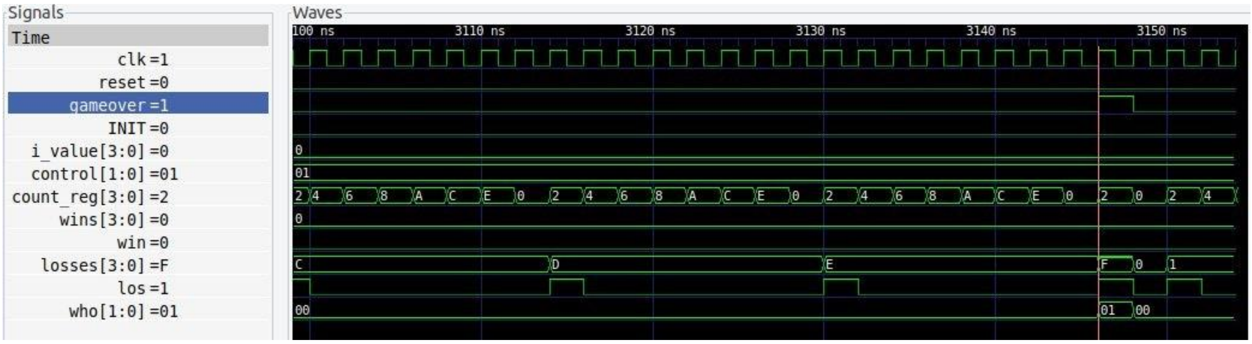
Control Signal = 2’b01 (count up by 2) initial value = 4’b0000



Signals

```
Time
          clk=1
        reset=0
     gameover=1
         INIT=0
  i_value[3:0]=0
  control[1:0]=01
count_reg[3:0]=2
     wins[3:0]=0
          win=0
   losses[3:0]=F
          los=1
     who[1:0]=01
```

Figure 13



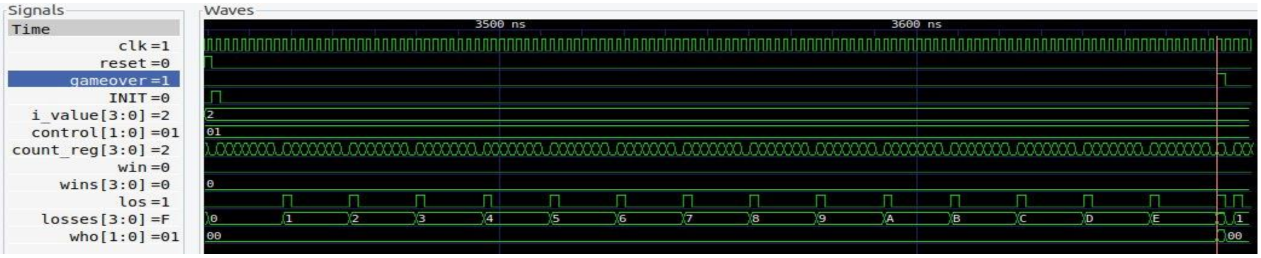
Signals

```
Time
          clk=1
        reset=0
     gameover=1
         INIT=0
  i_value[3:0]=0
  control[1:0]=01
count_reg[3:0]=2
     wins[3:0]=0
          win=0
   losses[3:0]=F
          los=1
     who[1:0]=01
```

Figure 14

Eighth Scenario:

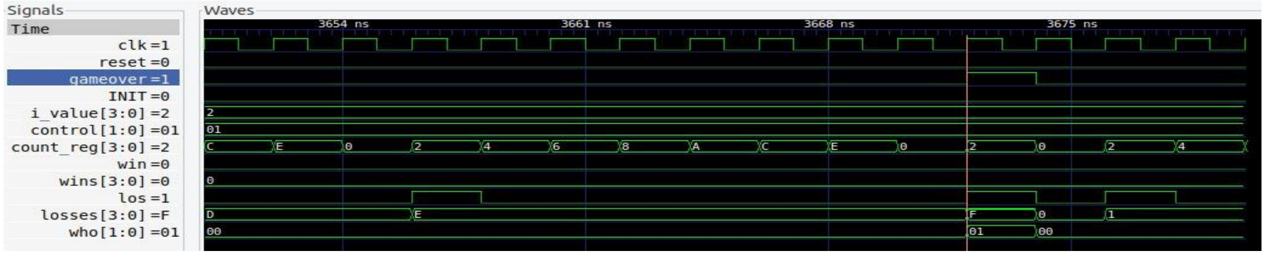
Control Signal = 2’b01 (count up by 2) initial value = 4’b0010



Signals

```
Time
          clk=1
        reset=0
     gameover=1
         INIT=0
  i_value[3:0]=2
  control[1:0]=01
count_reg[3:0]=2
          win=0
     wins[3:0]=0
          los=1
   losses[3:0]=F
     who[1:0]=01
```

Figure 15



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=2
control[1:0]=01
count_reg[3:0]=2
win=0
wins[3:0]=0
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 16

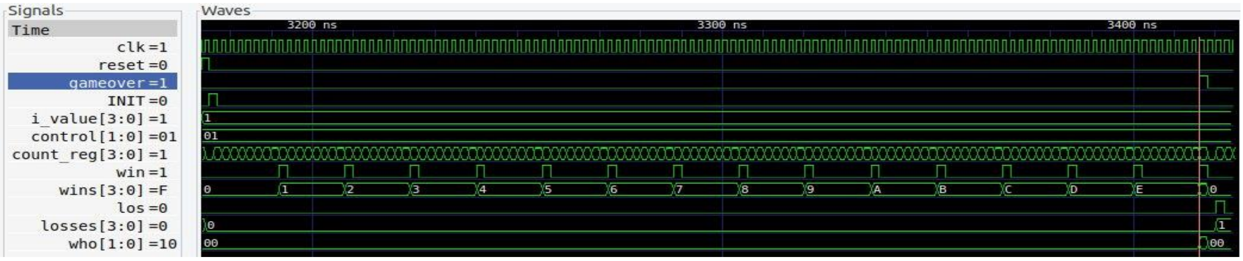
As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

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.

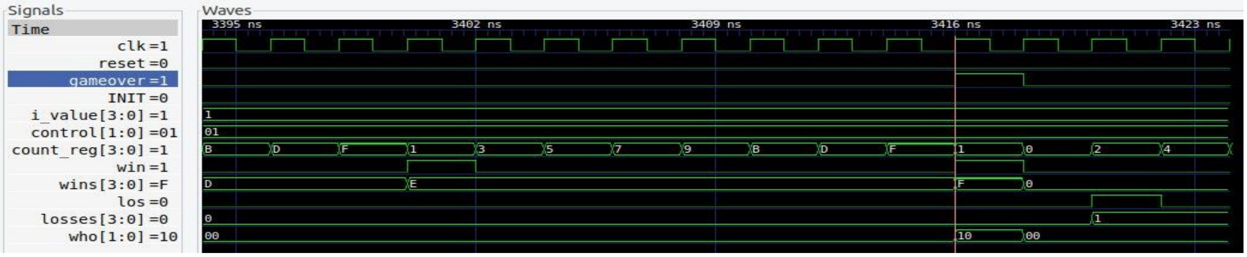
Nineth Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=1
control[1:0]=01
count_reg[3:0]=1
win=1
wins[3:0]=F
los=0
losses[3:0]=0
who[1:0]=10
```

Figure 17

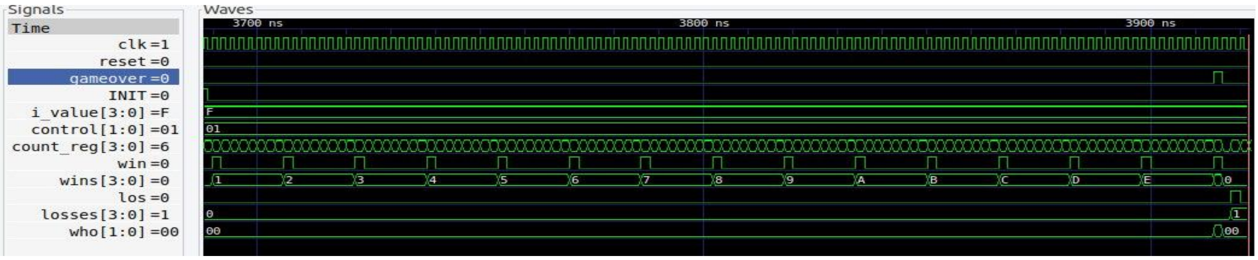


```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=1
control[1:0]=01
count_reg[3:0]=1
win=1
wins[3:0]=F
los=0
losses[3:0]=0
who[1:0]=10
```

Figure 18

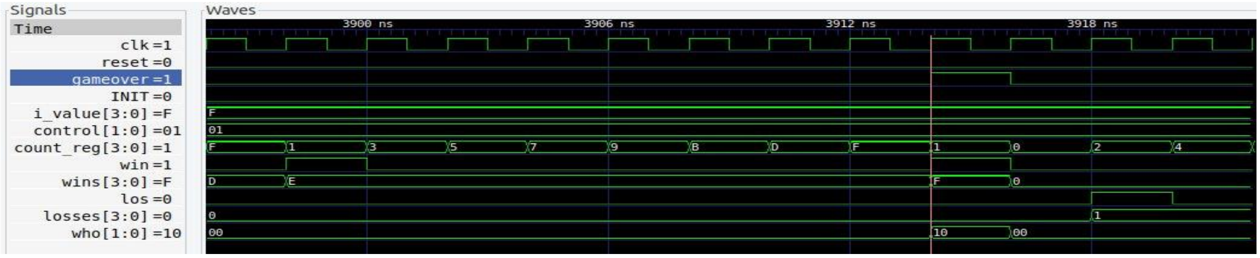
Tenth Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=0
INIT=0
i_value[3:0]=F
control[1:0]=01
count_reg[3:0]=6
win=0
wins[3:0]=0
los=0
losses[3:0]=1
who[1:0]=00
```

Figure 19



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=F
control[1:0]=01
count_reg[3:0]=1
win=1
wins[3:0]=F
los=0
losses[3:0]=0
who[1:0]=10
```

Figure 20

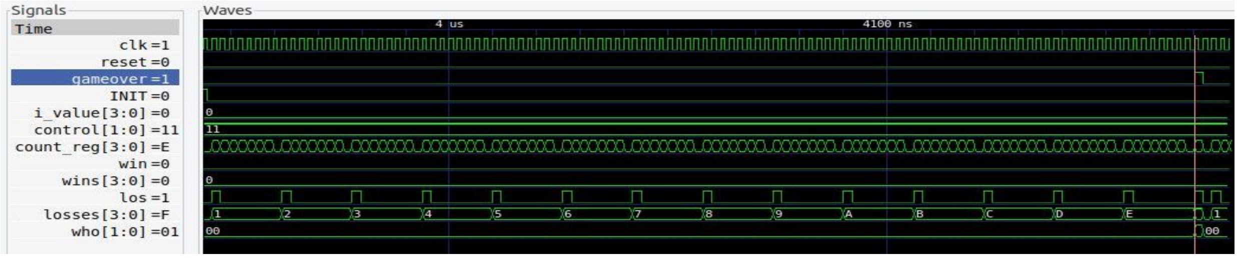
As we started from 1,15(ODD NUMBER), winner counter will be ahead from the loser counter by one.

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.

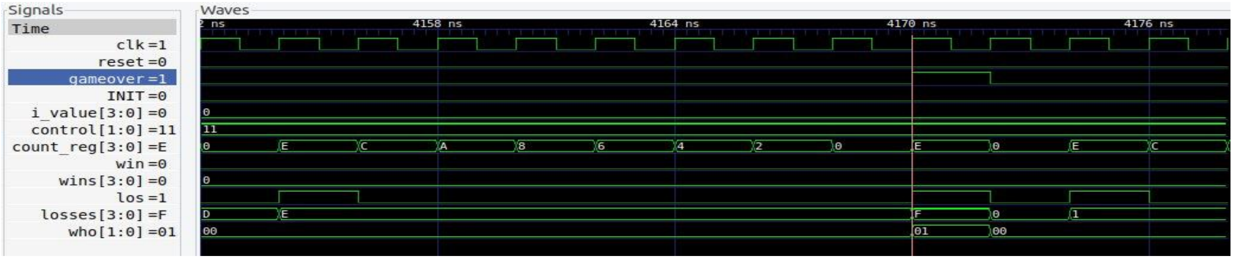
Eleventh Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=0
control[1:0]=11
count_reg[3:0]=E
win=0
wins[3:0]=0
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 21

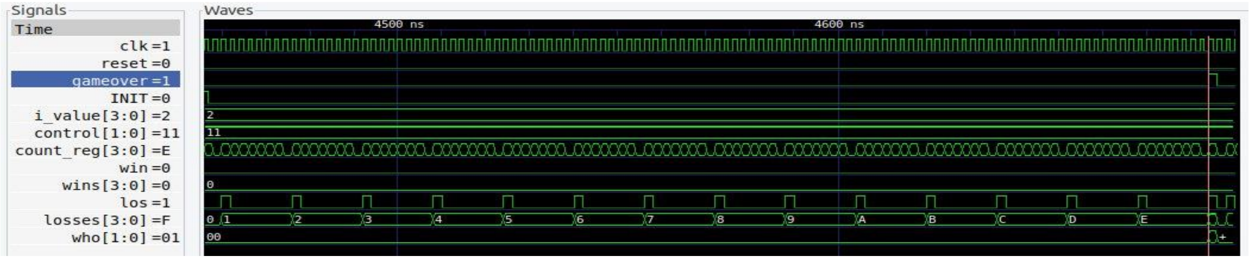


```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=0
control[1:0]=11
count_reg[3:0]=E
win=0
wins[3:0]=0
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 22

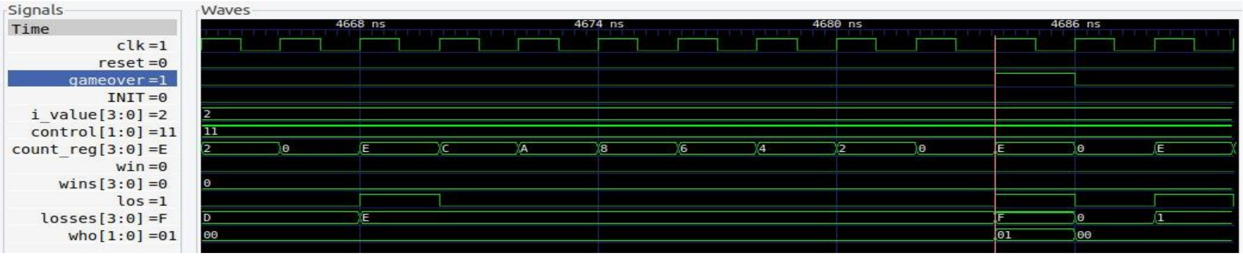
Twelfth Scenario:

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



```
Signals
Time
clk=1
reset=0
gameover=1
INIT=0
i_value[3:0]=2
control[1:0]=11
count_reg[3:0]=E
win=0
wins[3:0]=0
los=1
losses[3:0]=F
who[1:0]=01
```

Figure 23



```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=2
control[1:0]=11
count_reg[3:0]=E
        win=0
  wins[3:0]=0
        los=1
 losses[3:0]=F
   who[1:0]=01
```

Figure 24

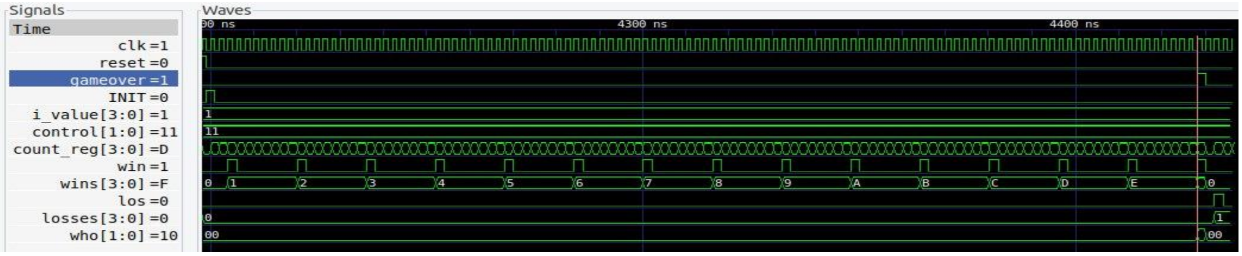
As we started from 0,2(EVEN NUMBER), loser counter will be ahead from the winner counter by one.

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.

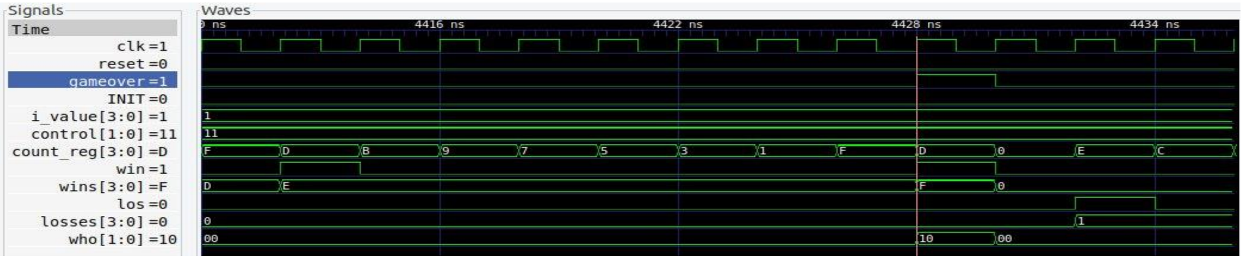
Thirteenth Scenario:

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



```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=1
control[1:0]=11
count_reg[3:0]=D
        win=1
  wins[3:0]=F
        los=0
 losses[3:0]=0
   who[1:0]=10
```

Figure 25

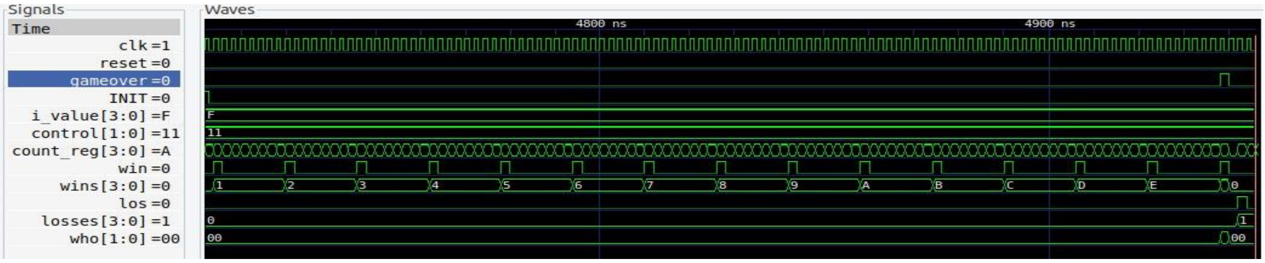


```
Signals
Time
        clk=1
      reset=0
   gameover=1
       INIT=0
 i_value[3:0]=1
control[1:0]=11
count_reg[3:0]=D
        win=1
  wins[3:0]=F
        los=0
 losses[3:0]=0
   who[1:0]=10
```

Figure 26

Fourteenth Scenario:

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



```
Signals
Time
        clk=1
      reset=0
   gameover=0
       INIT=0
 i_value[3:0]=F
control[1:0]=11
count_reg[3:0]=A
        win=0
  wins[3:0]=0
        los=0
 losses[3:0]=1
   who[1:0]=00
```

Figure 27

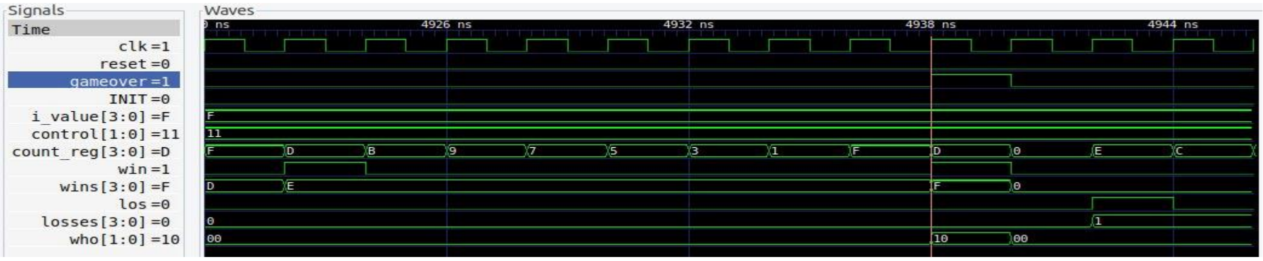


Figure 28

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.