Assignment 2

Program:

Course Code: CSE412

Course Name: Digital Verification

Omar Mohamed Omar AbdelSamea

1700903

Ain Shams University

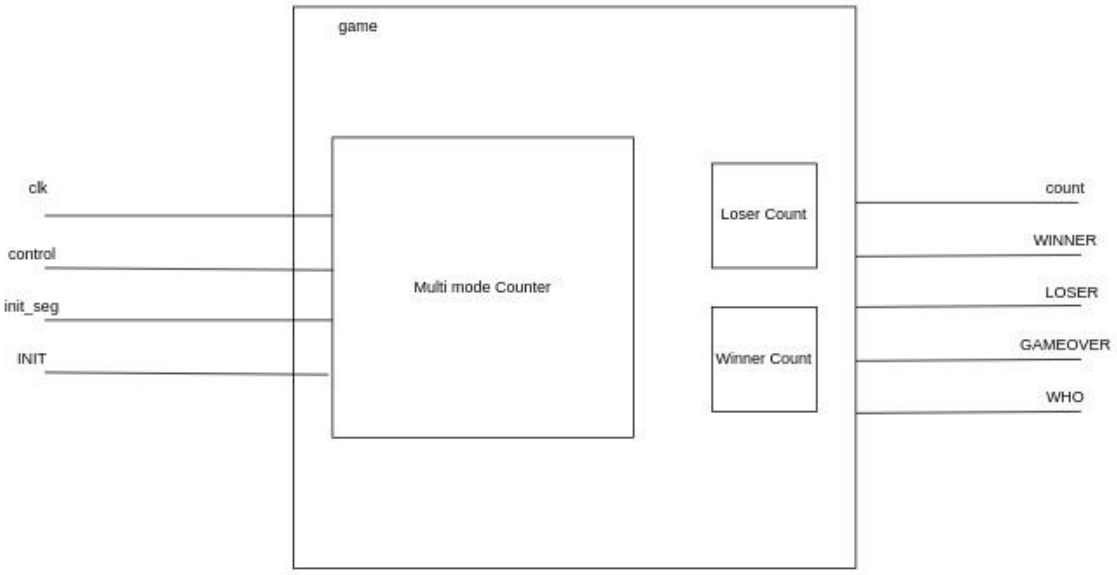
Faculty of Engineering

Spring Semester – 2022

MODULES

There are 2 modules multimode counter and game

1. **Multimode counter**: counter with 2 control bits that selects one of four modes
   - Counting up by 1
   - Counting up by 2
   - Counting down by 1
   - Counting down by 2
2. **Game**: main module responsible for checking the counter value and sets WINNER signal high when the counter value reaches all 1's then it adds one on the WINNER count register, sets LOSER signal high when the counter value reaches all 0's then it adds one on the LOSER count register. When WINNER count or LOSER count reaches value 15 the module sets GAMEOVER signal as high immediately.

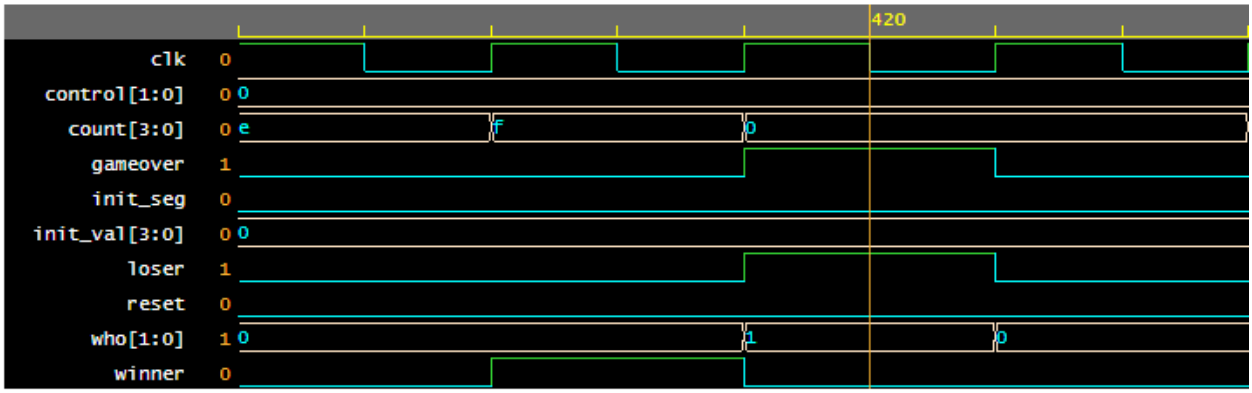


ENHANCED TEST BENCH

1- Normal ports replaced with interface `game_if`.
2- 2 modports for main module and test bench.
   a. Dut modport for game module
   b. Driver modport for game test bench
3- Appropriate clocking implemented by using clocking block in system Verilog.
4- Test bench implemented with program module.
5- Clock generator that eliminates racing conditions.
6- Assertion to validate each test case scenario.

TESTBENCH

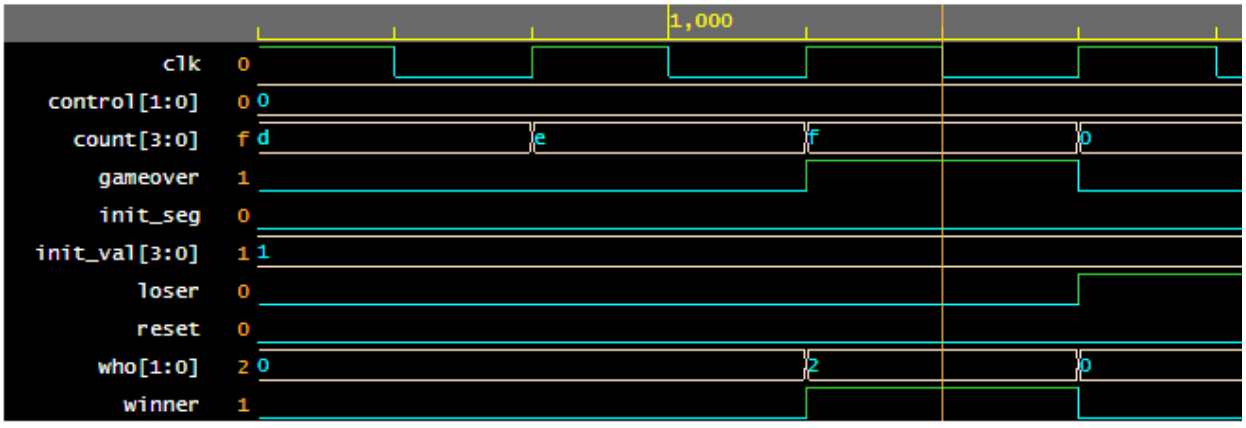
Scenario 1: Set Initial value to 0 to give loser an advantage with ctrl 0



Assertion:

```
#418 // t = 420
assertion_1: assert (gif.who == 2'b01)
  $info("loser finished the game");
else
  errors++;
```

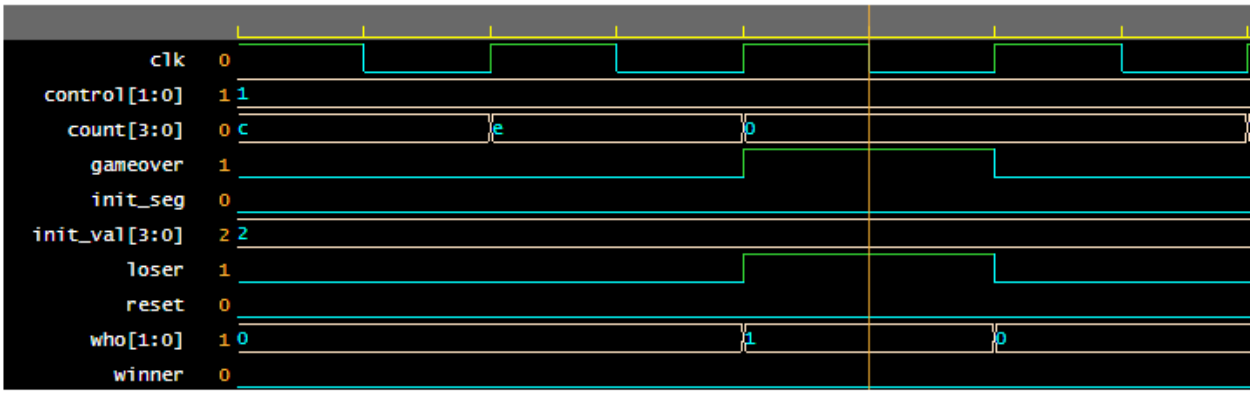
Scenario 2: Set Initial value to 1 to give winner an advantage with ctrl 0



Assertion:

```
#478 // t = 1002
assertion_2: assert (gif.who == 2'b10)
  $info("winner finished the game");
else
  errors++;
```

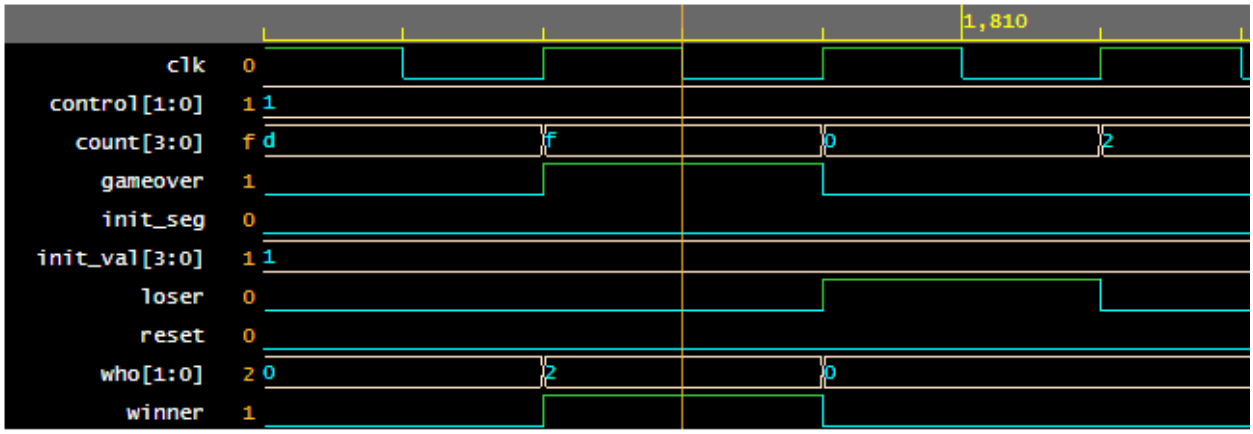
Scenario 3: Set Initial value to 2 to give loser an advantage with ctrl 1



Assertion:

```
#240 // t = 1286
assertion_3: assert (gif.who == 2'b01)
  $info("loser finished the game");
else
  errors++;
```

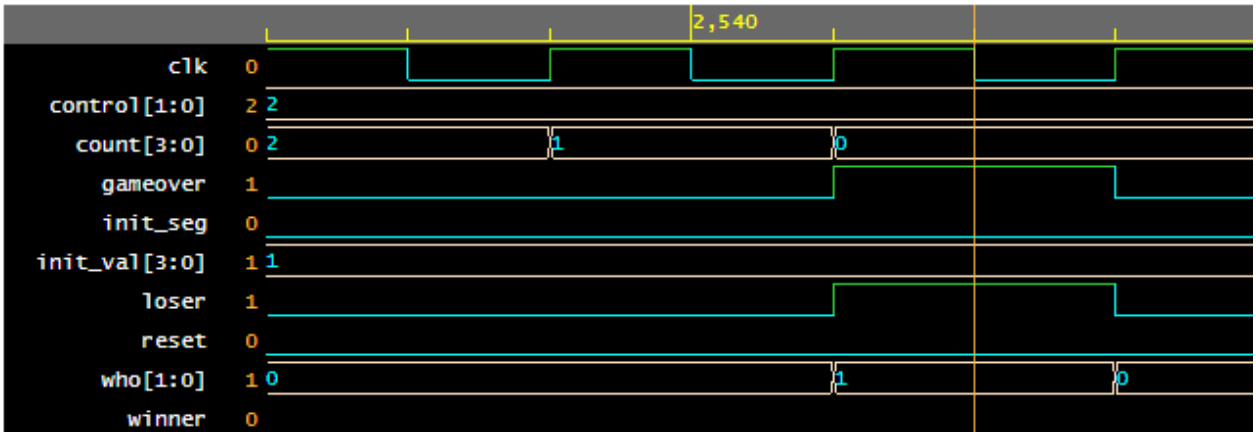
Scenario 4: Set Initial value to 1 to give winner an advantage with ctrl 1



Assertion:

```
#240 // t = 1808
assertion_4: assert (gif.who == 2'b10)
  $info("winner finished the game");
else
  errors++;
```

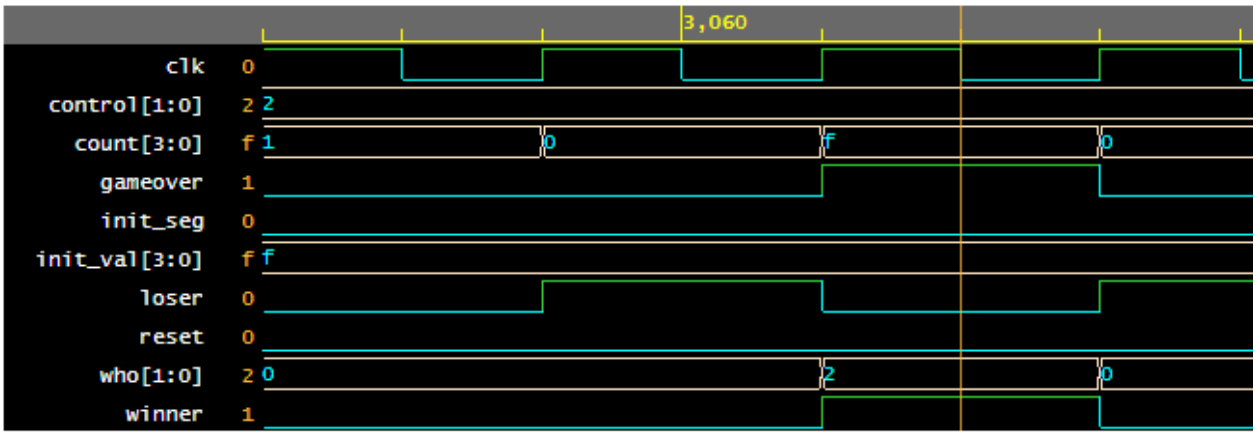
Scenario 5: Set Initial value to 1 to give loser an advantage with ctrl 2



Assertion:

```
 #452 // t = 2542
assertion_5: assert (gif.who == 2'b01)
  $info("loser finished the game");
else
  errors++;
```

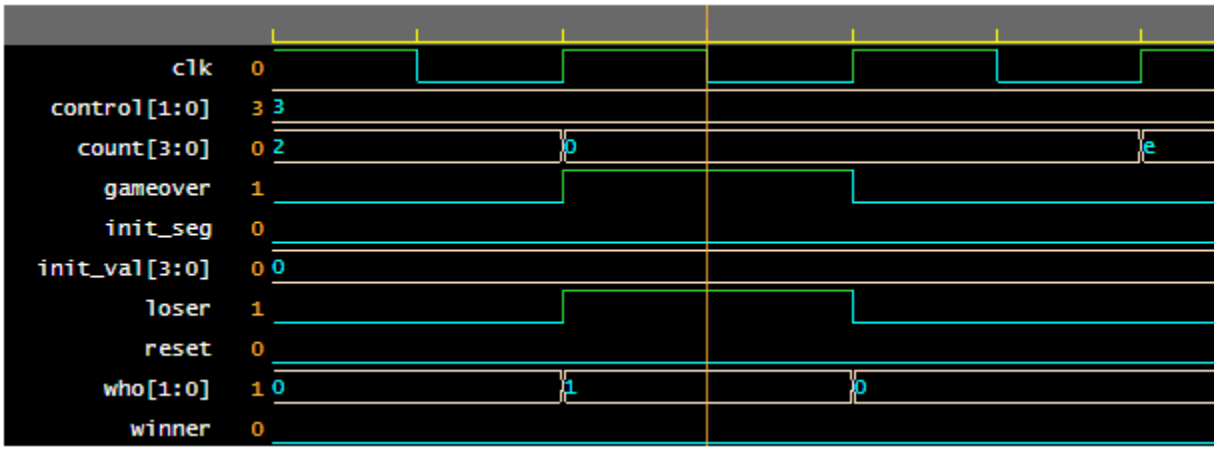
Scenario 6: Set Initial value to 15 to give winner an advantage with ctrl 2



Assertion:

```
#450 // t = 3062
assertion_6: assert (gif.who == 2'b10)
  $info("winner finished the game");
else
  errors++;
```

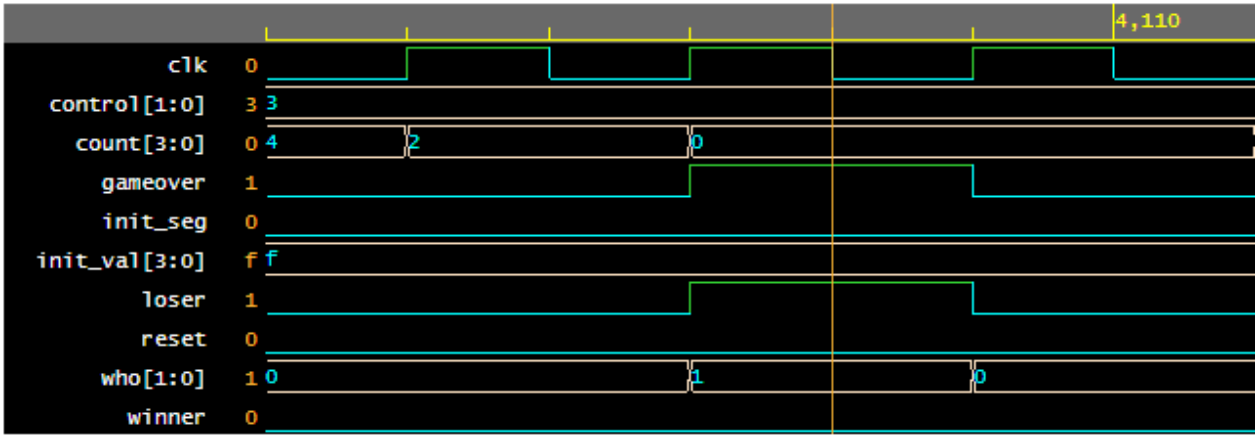
Scenario 7: Set Initial value to 0 to give loser an advantage with ctrl 3



Assertion:

```
#242 // t = 3376
assertion_7: assert (gif.who == 2'b01)
  $info("loser finished the game");
else
  errors++;
```

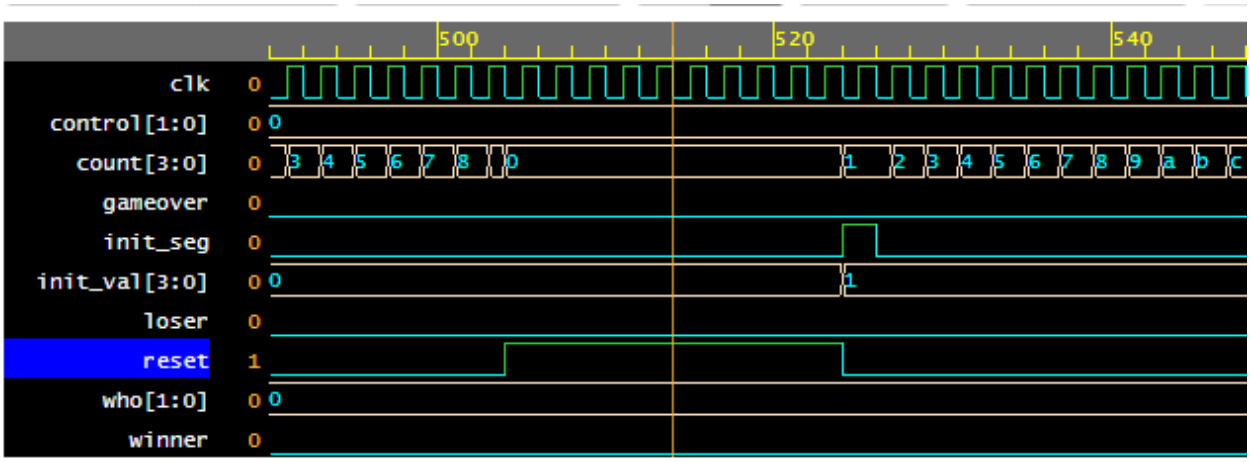
Scenario 8: Set Initial value to 15 to give winner an advantage with ctrl 3



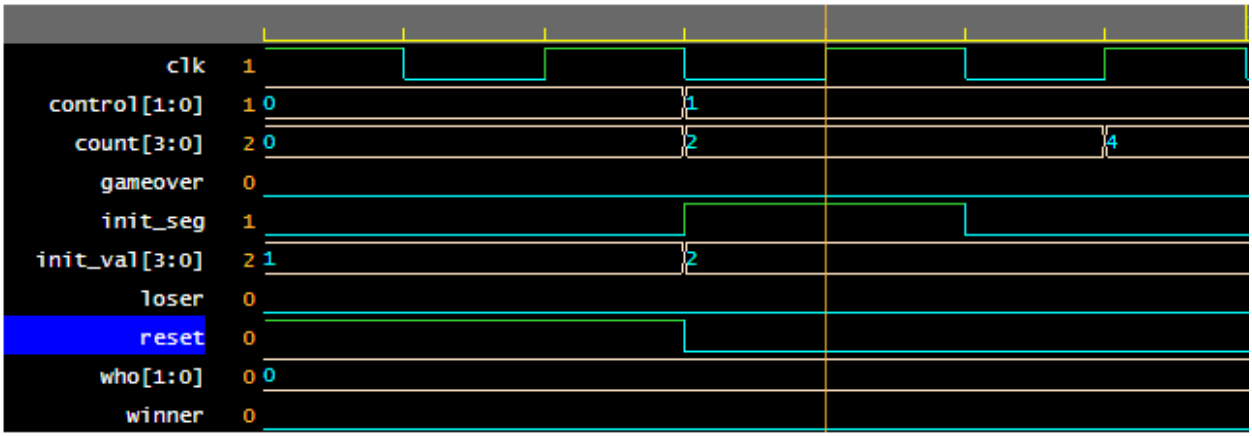
Assertion:

```
#226 // t = 3882
assertion_8: assert (gif.who == 2'b10)
  $info("winner finished the game");
else
  errors++;
```

Reset example



Initial value load example



```
/*
 * multimode counter takes 2 bit control which determens value and up or down
 *
 * @port input clk
 * @port input reset
 * @port input control
 * @port input init_seg
 * @port input init_val
 * @port input gameover
 * @port output count
 */
module multimode_counter#(DATA_WIDTH=4)(clk, reset, control, init_seg, init_val,
gameover, count);
  /* input ports */
  input clk;
  input reset;
  input [1:0] control;
  input init_seg;
  input [DATA_WIDTH - 1:0] init_val;
  input gameover;

  /* output ports */
  output [DATA_WIDTH - 1:0] count;

  bit [DATA_WIDTH - 1:0] value;
  bit [DATA_WIDTH - 1:0] counter;

  assign count = counter;

  typedef enum { UP, DOWN } direction_type;
  direction_type direction;

  always @(posedge clk or posedge reset or posedge init_seg) begin
    if(reset || gameover)
      counter = 0;
    else if (init_seg)
      counter = init_val;
    else begin
      /* mode selection based on control bits */
      case(control)
        2'b00: begin value = 1; direction = UP; end
        2'b01: begin value = 2; direction = UP; end
        2'b10: begin value = 1; direction = DOWN; end
        2'b11: begin value = 2; direction = DOWN; end
```

```
      endcase
      if(direction == UP)
        counter = counter + value;
      else
        counter = counter - value;
    end
  end
endmodule

/*
 * game module holds score board and instance of the multimode counter to run the
game
 *
 * @port input clk
 * @port input reset
 * @port input control
 * @port output count
 * @port input init_seg
 * @port output INIT
 * @port output WINNER
 * @port output LOSER
 * @port output GAMEOVER
 * @port output WHO
 */
module game#(DATA_WIDTH=4)(game_if.dut gif);
    parameter WINS = 15;
    parameter LOSES = 15;

    wire [DATA_WIDTH - 1:0] counter;

    bit winner_seg = 0;
    bit loser_seg = 0;
    bit gameover_seg = 0;
    bit [1:0] who_seg = 0;

    /* register that holds number of winner signal reached high
    and loser signal reached high */
    int winner_count = 0;
    int loser_count = 0;

    /* assign signals to output wires to match the required signal names */
    assign gif.count = counter;
    assign gif.winner = winner_seg;
    assign gif.loser = loser_seg;
```

```
  assign gif.gameover = gameover_seg;
  assign gif.who = who_seg;

  /* checking for counter value to set winner or loser if exist */
  always @(counter) begin
    /* if count equals all 1's set winner signal high */
    if(counter == (2 ** DATA_WIDTH) - 1) begin
      winner_seg = 1;
      winner_count = winner_count + 1;
    end
    /* if count equals all 0's set loser signal high */
    else if(counter == 0 && gif.reset != 1) begin
      loser_seg = 1;
      loser_count = loser_count + 1;
    end
    /* number of winner signal high or number of loser signal high to
    determine who finished the game */
    if(winner_count == WINS || loser_count == LOSES) begin
      gameover_seg = 1;
      /* if winner finished the game set WHO signal = 2 */
      if(winner_count == WINS)
        who_seg = 2'b10;
      /* if loser finished the game set WHO signal = 1 */
      else
        who_seg = 2'b01;
    end
  end

  /* check for clock or asynchornous reset */
always @(posedge gif.clk or posedge gif.reset) begin
      /* reset loser and winner signals every clock cycles */
      loser_seg = 0;
      winner_seg = 0;
      /* reset all values to 0 if signal reset is high or gameover is high */
    if(gameover_seg || gif.reset) begin
        winner_seg = 0;
        loser_seg = 0;
        winner_count = 0;
        loser_count = 0;
        gameover_seg = 0;
        who_seg = 0;
      end
  end
  /* multimode counter instance */
```

```
  multimode_counter mmc(gif.clk, gif.reset, gif.control, gif.init_seg,
gif.init_val, gif.gameover, counter);
endmodule

/*
 * clock generator module
 *
 * @port ouput clk
 */
module clock_generator#(CLK=1)(output bit clk);
  initial
    forever #CLK clk = ~clk;

endmodule

/*
 * top module of the system
 */

module top;
  wire clk;
  clock_generator(clk);
  game_if i1(clk);
  game g1(i1.dut);
  game_tb gtb1(i1.driver);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
    #5000 $finish;
  end
endmodule

/*
 * game interface holds all ports with modport for game and another for testbench
 *
 * @port clk
 * @port reset
 * @port control
 * @port count
 * @port init_seg
 * @port init_val
 * @port winner
 * @port loser
```

```
 * @port gameover
 * @port who
 */
interface game_if (input bit clk);
  parameter DATA_WIDTH = 4;
  bit reset;
  bit [1:0] control = 2'b00;
  bit init_seg = 0;
  bit [DATA_WIDTH - 1:0] init_val;

  wire [DATA_WIDTH - 1:0] count;
  wire loser;
  wire winner;
  wire gameover;
  wire [1:0] who;

  clocking cb @(posedge clk);
    default output #1ns;
    output reset, control, init_seg, init_val;
    input clk, count, winner, loser, gameover, who;
  endclocking

  modport dut(input clk, reset, control, init_seg, init_val, output count,
winner, loser, gameover, who);
  modport driver(clocking cb);

endinterface

/****************************** Testbench ******************************/

program game_tb#(DATA_WIDTH=4)(game_if gif);
  int errors = 0;
  initial begin
    /* Scenario 1: Set Initial value to 0 to give loser an advantage with ctrl
0*/
    gif.cb.control <= 2'b00;
    gif.cb.init_seg <= 1; gif.cb.init_val <= 0;
    #2 gif.cb.init_seg <= 0;

  #418 // t = 420
    assertion_1: assert (gif.who == 2'b01)
      $info("loser finished the game");
    else
      errors++;
```

```
    #82 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;
    /* Scenario 2: Set Initial value to 1 to give winner an advantage with ctrl
0*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 1;
    #2 gif.cb.init_seg <= 0;

    #478 // t = 1002
    assertion_2: assert (gif.who == 2'b10)
      $info("winner finished the game");
    else
      errors++;

    #22 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;

    gif.cb.control <= 2'b01;
    /* Scenario 3: Set Initial value to 2 to give loser an advantage with ctrl
1*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 2;
    #2 gif.cb.init_seg <= 0;

    #240 // t = 1286
    assertion_3: assert (gif.who == 2'b01)
      $info("loser finished the game");
    else
      errors++;

    #260 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;

    /* Scenario 4: Set Initial value to 1 to give winner an advantage with ctrl
1*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 1;
    #2 gif.cb.init_seg <= 0;

    #240 // t = 1808
    assertion_4: assert (gif.who == 2'b10)
      $info("winner finished the game");
    else
      errors++;

    #260 gif.cb.reset <= 1;
```

```
    #20 gif.cb.reset <= 0;

    gif.cb.control <= 2'b10;
    /* Scenario 5: Set Initial value to 1 to give loser an advantage with ctrl
2*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 1;
    #2 gif.cb.init_seg <= 0;

    #452 // t = 2542
    assertion_5: assert (gif.who == 2'b01)
      $info("loser finished the game");
    else
      errors++;


    #48 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;

    /* Scenario 6: Set Initial value to 15 to give winner an advantage with ctrl
2*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 15;
    #2 gif.cb.init_seg <= 0;

    #450 // t = 3062
    assertion_6: assert (gif.who == 2'b10)
      $info("winner finished the game");
    else
      errors++;

    #50 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;

    gif.cb.control <= 2'b11;
    /* Scenario 7: Set Initial value to 0 to give loser an advantage with ctrl
3*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 0;
    #2 gif.cb.init_seg <= 0;

    #242 // t = 3376
    assertion_7: assert (gif.who == 2'b01)
      $info("loser finished the game");
    else
      errors++;
```

```
    #258 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;

    /* Scenario 8: Set Initial value to 15 to give winner an advantage with ctrl
3*/
    gif.cb.init_seg <= 1; gif.cb.init_val <= 15;
    #2 gif.cb.init_seg <= 0;

    #226 // t = 3882
    assertion_8: assert (gif.who == 2'b10)
      $info("winner finished the game");
    else
      errors++;

    #500 gif.cb.reset <= 1;
    #20 gif.cb.reset <= 0;
  end

   final begin
     $display ("Test Ended");
     if (errors > 0)
       $display ("%0d test failed out of 8", errors);
     else
       $display ("all tests passed successfully");
   end
endprogram
```

If all test cases pass:

```
Info: "testbench.sv", 50: top.gtb1.assertion_1: at time 420 ns
loser finished the game
Info: "testbench.sv", 62: top.gtb1.assertion_2: at time 1002 ns
winner finished the game
Info: "testbench.sv", 76: top.gtb1.assertion_3: at time 1286 ns
loser finished the game
Info: "testbench.sv", 89: top.gtb1.assertion_4: at time 1808 ns
winner finished the game
Info: "testbench.sv", 103: top.gtb1.assertion_5: at time 2542 ns
loser finished the game
Info: "testbench.sv", 117: top.gtb1.assertion_6: at time 3062 ns
winner finished the game
Info: "testbench.sv", 131: top.gtb1.assertion_7: at time 3376 ns
loser finished the game
Info: "testbench.sv", 144: top.gtb1.assertion_8: at time 3882 ns
winner finished the game
Test Ended
all tests passed succesfully
```

If error occurs:

```
Info: "testbench.sv", 50: top.gtb1.assertion_1: at time 420 ns
loser finished the game
Info: "testbench.sv", 62: top.gtb1.assertion_2: at time 1002 ns
winner finished the game
Info: "testbench.sv", 76: top.gtb1.assertion_3: at time 1286 ns
loser finished the game
Info: "testbench.sv", 89: top.gtb1.assertion_4: at time 1808 ns
winner finished the game
"testbench.sv", 103: top.gtb1.assertion_5: started at 2542ns failed at 2542ns
        Offending '(gif.who == 2'b11)'
Info: "testbench.sv", 117: top.gtb1.assertion_6: at time 3062 ns
winner finished the game
Info: "testbench.sv", 131: top.gtb1.assertion_7: at time 3376 ns
loser finished the game
Info: "testbench.sv", 144: top.gtb1.assertion_8: at time 3882 ns
winner finished the game
Test Ended
1 test failed out of 8
```