Assignment 2 EXTRA

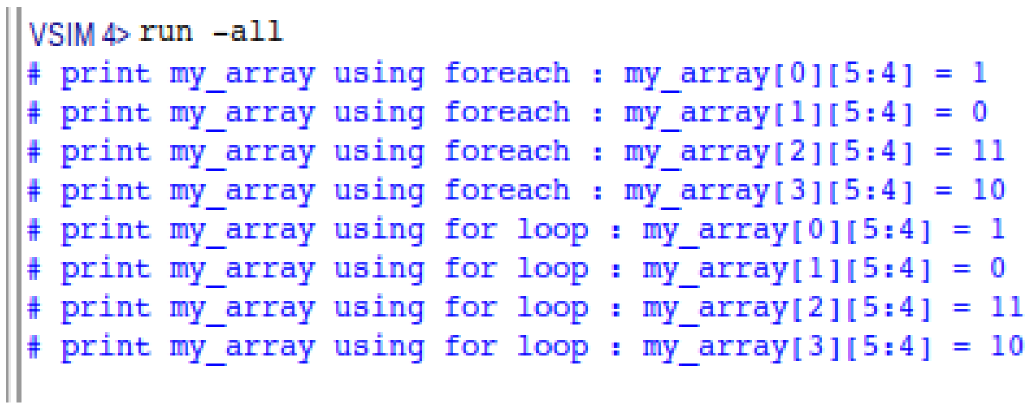
Digital Design Verification

Contents

1 Q1: 2-State Array

1.1 1. Testbench

```
module two_state_array;

   //=================================
   // Declare 2-state array
   //=================================
   bit [11:0] my_array [0:3] ;

initial begin
   //==============================
   // initialize array elements
   //==============================
   my_array[0] = 12'h012;
   my_array[1] = 12'h345;
   my_array[2] = 12'h678;
   my_array[3] = 12'h9AB;

   //==============================
   // print my_array using foreach
   //==============================
   foreach(my_array[i])
      $display("print␣my_array␣using␣foreach␣:␣my_array[%0d][5:4]␣=␣%0b",i,my_array[i][5:4]);

   //==============================
   // print my_array using for loop
   //==============================
   for(int i = 0 ;i < $size(my_array,1) ;i++)
      $display("print␣my_array␣using␣for␣loop␣:␣my_array[%0d][5:4]␣=␣%0b",i,my_array[i][5:4]);
end

endmodule
```



```
VSIM 4> run -all
# print my_array using foreach : my_array[0][5:4] = 1
# print my_array using foreach : my_array[1][5:4] = 0
# print my_array using foreach : my_array[2][5:4] = 11
# print my_array using foreach : my_array[3][5:4] = 10
# print my_array using for loop : my_array[0][5:4] = 1
# print my_array using for loop : my_array[1][5:4] = 0
# print my_array using for loop : my_array[2][5:4] = 11
# print my_array using for loop : my_array[3][5:4] = 10
```

Figure 1: Transcript : all test cases passed

2 Q2: ALU

2.1 1. Testbench code

```
`timescale 1ns/1ps

import ALSU_pkg::*;

module ALSU_tb;

  // -----------------------------------
  // Testbench signals
  // -----------------------------------
  logic clk;
  logic rst;
  logic cin;
  logic red_op_A;
  logic red_op_B;
  logic bypass_A;
  logic bypass_B;
  logic direction;
  logic serial_in;
  logic signed [2:0] A;
  logic signed [2:0] B;
  logic [2:0]        opcode;
  wire [15:0]        leds;
  wire signed [5:0]  out;

  // -----------------------------------
  // DUT instantiation
  // -----------------------------------
  ALSU #(
    .INPUT_PRIORITY("A"),
    .FULL_ADDER("ON")
  ) dut (
    .clk        (clk),
    .rst        (rst),
    .cin        (cin),
    .red_op_A   (red_op_A),
    .red_op_B   (red_op_B),
    .bypass_A   (bypass_A),
    .bypass_B   (bypass_B),
    .direction  (direction),
    .serial_in  (serial_in),
    .A          (A),
    .B          (B),
    .opcode     (opcode),
    .leds       (leds),
    .out        (out)
  );

  // --------------------------------------
  // Create an object for random stimulus
  // --------------------------------------
  alsu_rand_class stim;

  // ----------------------------------
  // Clock & reset generation
  // ----------------------------------
  initial begin
        clk = 0;
        forever begin
           #5 clk = ~clk;
           stim.clk = clk;
        end
   end
  // ------------
  // reset task
  // ------------
  task do_reset();
    rst = 1;
    #10;
    // Check result against a golden model
      golden_model(
        rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
        direction, opcode
      );
    #10;
    rst = 0;
  endtask

  // ----------------------------------
  // Golden model for reference
  // ----------------------------------
task golden_model(
    input logic rst,
    input logic signed [2:0] A, B,
    input logic cin, serial_in, red_op_A, red_op_B,
    input logic bypass_A, bypass_B, direction,
    input logic [2:0] opcode
);

    logic signed [5:0] expected_out;
    logic [15:0] expected_leds;

    logic invalid_red_op, invalid_opcode, invalid;

    // Invalid condition handling
    invalid_red_op = (red_op_A | red_op_B) & (opcode[1] | opcode[2]);
    invalid_opcode = opcode[1] & opcode[2];
    invalid = invalid_red_op | invalid_opcode;

    if(rst) begin
       expected_out = 0;
       expected_leds = 0;
    end else begin
      if (invalid)
        expected_leds = ~expected_leds;
      else
        expected_leds = 0;
    end


    if (bypass_A && bypass_B)
    expected_out = ("A" == "A") ? A : B;  // INPUT_PRIORITY is "A"
    else if (bypass_A)
    expected_out = A;
    else if (bypass_B)
    expected_out = B;
    else if (invalid)
    expected_out = 0;
    else begin
    case (opcode)
        3'h0: begin // OR or Reduction OR
            if (red_op_A && red_op_B)
                expected_out = ("A" == "A") ? |A : |B;
            else if (red_op_A)
                expected_out = |A;
            else if (red_op_B)
                expected_out = |B;
            else
                expected_out = A | B;
        end
        3'h1: begin // XOR or Reduction XOR
            if (red_op_A && red_op_B)
                expected_out = ("A" == "A") ? ^A : ^B;
            else if (red_op_A)
                expected_out = ^A;
            else if (red_op_B)
                expected_out = ^B;
            else
                expected_out = A ^ B;
        end
        3'h2: expected_out = A + B; // ADD
        3'h3: expected_out = A * B; // MUL
        3'h4: begin // SHIFT
```

```
            if (direction)
                expected_out = {expected_out[4:0], serial_in};
            else
                expected_out = {serial_in, expected_out[5:1]};
        end
        3'h5: begin // ROTATE
            if (direction)
                expected_out = {expected_out[4:0], expected_out[5]};
            else
                expected_out = {expected_out[0], expected_out[5:1]};
        end
        default: expected_out = 0;
    endcase
    end

    // Wait another clock so the output is stable
    @(posedge clk);
    #1;

    if ( (out != expected_out) && (leds != expected_leds)) begin
        $error("[ALSU]␣Mismatch␣with␣golden␣model:␣opcode=%0b.␣␣out=%0d,␣expected_out=%0d",
                opcode, out, expected_out);
    end
endtask


  // -----------------------------------
  // Test/Stimulus
  // -----------------------------------
  initial begin

    stim = new();

    do_reset();  // start in reset


    for (int i = 0; i < 200; i++) begin

      // Randomize with constraints
      if(!(stim.randomize())) begin
         $error("Randomization␣failed!");
         $finish;
      end

      // Drive signals
      rst         = stim.rst;
      cin         = stim.cin;
      red_op_A    = stim.red_op_A;
      red_op_B    = stim.red_op_B;
      bypass_A    = stim.bypass_A;
      bypass_B    = stim.bypass_B;
      direction   = stim.direction;
      serial_in   = stim.serial_in;
      opcode      = stim.opcode;
      A           = stim.A;
      B           = stim.B;

      // Wait a clock for inputs to be sampled
      @(posedge clk);

      // Check result against a golden model
      golden_model(
        rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
        direction, opcode
      );

    end

    $display("ALSU␣test␣completed");
    $finish;
  end

endmodule
```

2.2 2. Package code

```
package ALSU_pkg;
  // -----------------------------------------
  // 1) Define the opcodes (including invalids)
  // -----------------------------------------
  typedef enum logic [2:0] {
    OR_0       = 3'h0,  // 000
    XOR_1      = 3'h1,  // 001
    ADD_2      = 3'h2,  // 010
    MUL_3      = 3'h3,  // 011
    SHIFT_4    = 3'h4,  // 100
    ROTATE_5   = 3'h5,  // 101
    INVALID_6  = 3'h6,  // 110
    INVALID_7  = 3'h7   // 111
  } opcode_e;

  // -----------------------------------------
  // 2) 3-bit signed range is -4 .. +3
  // -----------------------------------------
  localparam logic signed [2:0] MAXNEG = -4;  // 3'b100
  localparam logic signed [2:0] ZERO   =  0;
  localparam logic signed [2:0] MAXPOS =  3;  // 3'b011

    class alsu_rand_class;
      // ------------------------------
      // Randomizable DUT inputs
      // ------------------------------
      bit                     clk;
      rand bit                rst;
      rand bit                cin;
      rand bit                red_op_A;
      rand bit                red_op_B;
      rand bit                bypass_A;
      rand bit                bypass_B;
      rand bit                direction;
      rand bit                serial_in;
      rand opcode_e           opcode;
      rand logic signed [2:0] A;
      rand logic signed [2:0] B;


      // ------------------------------
      // Constraints from specification
      // ------------------------------

      // (a) Make RESET happen with a low probability
      constraint c_reset_low_prob {
        rst dist { 0 := 95, 1 := 5 };
      }

      // (b) For ADD or MUL, pick corner values of A,B more often
      //     (MAXNEG, ZERO, MAXPOS) than the other possibilities.
      //     Weighted distribution is used here.
      constraint c_adder_mult_corner {
         if (opcode inside {ADD_2, MUL_3}) {
             A dist { MAXNEG := 5, ZERO := 5, MAXPOS := 5, [-3:-1] := 1, [1:2] := 1 };
             B dist { MAXNEG := 5, ZERO := 5, MAXPOS := 5, [-3:-1] := 1, [1:2] := 1 };
         }
      }


      // (c) If opcode=OR or XOR and red_op_A=1, then A has exactly one bit set
      //     and B is 0 .
      constraint c_red_opA_onebit {
        if ((opcode==OR_0 || opcode==XOR_1) && red_op_A==1) {
          // Force B to be 0 or near 0
          B == 0;
          // A has exactly 1 bit set in its 3 bits:
          A dist {1:=5, 2:=5, MAXNEG:=5 , MAXPOS := 1 , [-3:0] := 1};
        }
      }

      // (d) Similarly, if opcode=OR or XOR and red_op_B=1, then B has exactly one bit set
      //     and A is 0.
      constraint c_red_opB_onebit {
        if ((opcode==OR_0 || opcode==XOR_1) && red_op_B==1) {
          A == 0;
          B dist {1:=5, 2:=5, MAXNEG:=5 , MAXPOS := 1 , [-3:0]:= 1 };
        }
      }

      // (e) Invalid cases (opcode=6 or 7, or red_op_X=1 for non-OR/XOR)
      //     should occur *less* frequently.
      //     Weighted distribution on opcode:
      constraint c_opcode_distribution {
        opcode dist {
          INVALID_6  := 1,
          INVALID_7  := 1,
          OR_0       := 5,
          XOR_1      := 5,
          ADD_2      := 5,
          MUL_3      := 5,
          SHIFT_4    := 5,
          ROTATE_5   := 5
        };
      }

      // (f) For red_op_A/B, require them to be 0 if opcode in {ADD_2, MUL_3, SHIFT_4, ROTATE_5}
      // except for a small chance to produce the invalid scenario:
      constraint c_red_op_non_orxor {
        if (opcode inside {ADD_2, MUL_3, SHIFT_4, ROTATE_5}) {
          (red_op_A == 0) dist {0:=95, 1:=5};
          (red_op_B == 0) dist {0:=95, 1:=5};
        }
      }

      constraint c_red_op_orxor {
        if (opcode inside {XOR_1, OR_0}) {
              {red_op_A, red_op_B} dist {2'b00 :/ 5, 2'b01 :/ 10, 2'b10 :/ 10 ,2'b11 :/ 75};
        }
      }

      // (g) bypass_A and bypass_B should be disabled most of the time
      constraint c_bypass_dist {
          bypass_A dist {0:=3,1:=1};
          bypass_B dist {0:=3,1:=1};
      }

      // (h) If SHIFT or ROTATE, do not constrain A,B.
      //     (No explicit constraint needed => they can be anything.)




      // -----------------------------------
```

```
      // Coverage points
      // -----------------------------------
      covergroup cg @(posedge clk);
        coverpoint rst;
        coverpoint cin;
        coverpoint red_op_A;
        coverpoint red_op_B;
        coverpoint bypass_A;
        coverpoint bypass_B;
        coverpoint opcode;
            coverpoint direction;
            coverpoint serial_in;
            coverpoint A;
            coverpoint B;
      endgroup

          // constructor
      function new();
        cg = new();
      endfunction


    endclass


endpackage
```

2.3 3. Design code

```
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input [2:0] opcode;
input signed [2:0] A, B;
output reg [15:0] leds;
output reg signed [5:0] out;

reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg signed cin_reg;
reg [2:0] opcode_reg;
reg signed [2:0] A_reg, B_reg;

wire invalid_red_op, invalid_opcode, invalid;

//Invalid handling
assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
assign invalid = invalid_red_op | invalid_opcode;

//Registering input signals
always @(posedge clk or posedge rst) begin
  if(rst) begin
     cin_reg <= 0;
     red_op_B_reg <= 0;
     red_op_A_reg <= 0;
     bypass_B_reg <= 0;
     bypass_A_reg <= 0;
     direction_reg <= 0;
     serial_in_reg <= 0;
     opcode_reg <= 0;
     A_reg <= 0;
     B_reg <= 0;
  end else begin
     cin_reg <= cin;
     red_op_B_reg <= red_op_B;
     red_op_A_reg <= red_op_A;
     bypass_B_reg <= bypass_B;
     bypass_A_reg <= bypass_A;
     direction_reg <= direction;
     serial_in_reg <= serial_in;
     opcode_reg <= opcode;
     A_reg <= A;
     B_reg <= B;
  end
end

//leds output blinking
always @(posedge clk or posedge rst) begin
  if(rst) begin
     leds <= 0;
  end else begin
      if (invalid)
        leds <= ~leds;
      else
        leds <= 0;
  end
end

//ALSU output processing
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
    if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
```

```
      out <= B_reg;
    else if (invalid)
        out <= 0;
    else begin
        case (opcode)
          3'h0: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
            else if (red_op_A_reg)
              out <= |A_reg;
            else if (red_op_B_reg)
              out <= |B_reg;
            else
              out <= A_reg | B_reg;
          end
          3'h1: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
            else if (red_op_A_reg)
              out <= ^A_reg;
            else if (red_op_B_reg)
              out <= ^B_reg;
            else
              out <= A_reg ^ B_reg;
          end
          3'h2: out <= A_reg + B_reg;
          3'h3: out <= A_reg * B_reg;
          3'h4: begin
            if (direction_reg)
              out <= {out[4:0], serial_in_reg};
            else
              out <= {serial_in_reg, out[5:1]};
          end
          3'h5: begin
            if (direction_reg)
              out <= {out[4:0], out[5]};
            else
              out <= {out[0], out[5:1]};
          end
        endcase
    end
  end
end

endmodule
```

2.4 4. Bug Fixes

no bugs except cin_reg is one bit not two bits

2.5 5. Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
ALSU_1	When the reset is asserted, the outputs should be low.	- Directed reset at the start of simulation via do_reset().- Randomized afterward using constraint c_reset_low_prob (5% chance of reset=1).	- Covergroup in alsu_rand_class tracks rst transitions. - Ensures we observe enough reset assertions for coverage.	- The testbench checks that when rst is high, out is driven to 0 and leds is 0 (or blinking is suppressed). - Invokes golden_model() to confirm.
ALSU_2	In the absence of invalid cases, when opcode is ADD, the output should perform addition on ports A and B, taking cin if FULL_ADDER is on.	- Randomize opcode under c_opcode_distribution so that ADD (3'h2) appears frequently. - Corner-case inputs for A and B under c_adder_mult_corner (favor -4,0,3). - red_op_A and red_op_B are mostly 0 to avoid invalid conditions.	- The coverage group tracks opcode, A, B, and cin to ensure corner cases (MAXNEG, ZERO, MAXPOS) are exercised. - Also measures how often ADD vs. other operations appear.	- After each random transaction, the testbench calls golden_model() to confirm out == A + B (plus cin if relevant). - Errors are flagged if out differs from the expected sum.

Table 1: Verification Plan

2.6 6. Do File

```
vlib work
vlog ALSU_pkg.sv ALSU.v ALSU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save ALSU_tb.ucdb -onexit
run -all

# to run do file
-- do run.txt
to execute coverage report
-- vcover report ALSU_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=ALSU
-- vcover report -details -cvg -output ALSU_coverage_report.txt ALSU_tb.ucdb
```

2.7 7. Coverage Report

```
Coverage Report by DU with details

=================================================================================
=== Design Unit: work.ALSU
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        32        32         0   100.00%

================================Branch Details================================

Branch Coverage for Design Unit work.ALSU

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ALSU.v
------------------------------------IF Branch------------------------------------
    24                                       402     Count coming in to IF
    24              1                         28       if(rst) begin
    35              1                        374       end else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    51                                       417     Count coming in to IF
    51              1                         43       if(rst) begin
    53              1                        374       end else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    54                                       374     Count coming in to IF
    54              1                        279           if (invalid)
    56              1                         95           else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                       384     Count coming in to IF
    63              1                         28       if(rst) begin
    66              1                        356       else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                       356     Count coming in to IF
    67              1                         15         if (bypass_A_reg && bypass_B_reg)
    69              1                         77         else if (bypass_A_reg)
    71              1                         46         else if (bypass_B_reg)
    73              1                        156         else if (invalid)
    75              1                         62         else begin
Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    76                                        62     Count coming in to CASE
    77              1                         17                 3'h0: begin
    87              1                         18                 3'h1: begin
    97              1                          6                 3'h2: out <= A_reg + B_reg;
    98              1                          7                 3'h3: out <= A_reg * B_reg;
    99              1                          5                 3'h4: begin
    105             1                          8                 3'h5: begin
                                               1     All False Count
Branch totals: 7 hits of 7 branches = 100.00%

------------------------------------IF Branch------------------------------------
    78                                        17     Count coming in to IF
    78              1                          3                   if (red_op_A_reg && red_op_B_reg)
    80              1                          5                   else if (red_op_A_reg)
    82              1                          2                   else if (red_op_B_reg)
    84              1                          7                   else
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    88                                        18     Count coming in to IF
    88              1                          1                   if (red_op_A_reg && red_op_B_reg)
    90              1                          5                   else if (red_op_A_reg)
    92              1                          5                   else if (red_op_B_reg)
    94              1                          7                   else
```

```
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    100                                          5     Count coming in to IF
    100              1                           1                if (direction_reg)
    102              1                           4                else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    106                                          8     Count coming in to IF
    106              1                           2                if (direction_reg)
    108              1                           6                else
Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----    -------    ------  --------
    Conditions                       6          6         0   100.00%

================================Condition Details================================

Condition Coverage for Design Unit work.ALSU --

  File ALSU.v
----------------Focused Condition View-------------------
Line       67 Item    1  (bypass_A_reg && bypass_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
  bypass_A_reg         Y
  bypass_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  bypass_A_reg_0        -
  Row   2:          1  bypass_A_reg_1        bypass_B_reg
  Row   3:          1  bypass_B_reg_0        bypass_A_reg
  Row   4:          1  bypass_B_reg_1        bypass_A_reg

----------------Focused Condition View-------------------
Line       78 Item    1  (red_op_A_reg && red_op_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
  red_op_A_reg         Y
  red_op_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  red_op_A_reg_0        -
  Row   2:          1  red_op_A_reg_1        red_op_B_reg
  Row   3:          1  red_op_B_reg_0        red_op_A_reg
  Row   4:          1  red_op_B_reg_1        red_op_A_reg

----------------Focused Condition View-------------------
Line       88 Item    1  (red_op_A_reg && red_op_B_reg)
Condition totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
  red_op_A_reg         Y
  red_op_B_reg         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  red_op_A_reg_0        -
  Row   2:          1  red_op_A_reg_1        red_op_B_reg
  Row   3:          1  red_op_B_reg_0        red_op_A_reg
  Row   4:          1  red_op_B_reg_1        red_op_A_reg


Expression Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----    -------    ------  --------
    Expressions                      8          8         0   100.00%

================================Expression Details================================

Expression Coverage for Design Unit work.ALSU --

  File ALSU.v
----------------Focused Expression View-----------------
Line       18 Item    1  ((red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]))
Expression totals: 4 of 4 input terms covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  -----------------------  --------------
   red_op_A_reg         Y
   red_op_B_reg         Y
  opcode_reg[1]         Y
  opcode_reg[2]         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  red_op_A_reg_0        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
  Row   2:          1  red_op_A_reg_1        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
  Row   3:          1  red_op_B_reg_0        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
  Row   4:          1  red_op_B_reg_1        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
  Row   5:          1  opcode_reg[1]_0       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
  Row   6:          1  opcode_reg[1]_1       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
  Row   7:          1  opcode_reg[2]_0       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])
  Row   8:          1  opcode_reg[2]_1       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])

----------------Focused Expression View-----------------
Line       19 Item    1  (opcode_reg[1] & opcode_reg[2])
Expression totals: 2 of 2 input terms covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  -----------------------  --------------
  opcode_reg[1]         Y
  opcode_reg[2]         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  opcode_reg[1]_0       opcode_reg[2]
  Row   2:          1  opcode_reg[1]_1       opcode_reg[2]
  Row   3:          1  opcode_reg[2]_0       opcode_reg[1]
  Row   4:          1  opcode_reg[2]_1       opcode_reg[1]

----------------Focused Expression View-----------------
Line       20 Item    1  (invalid_red_op | invalid_opcode)
Expression totals: 2 of 2 input terms covered = 100.00%

     Input Term   Covered  Reason for no coverage   Hint
    -----------  --------  -----------------------  --------------
  invalid_red_op         Y
  invalid_opcode         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  invalid_red_op_0      ~invalid_opcode
  Row   2:          1  invalid_red_op_1      ~invalid_opcode
  Row   3:          1  invalid_opcode_0      ~invalid_red_op
  Row   4:          1  invalid_opcode_1      ~invalid_red_op


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      48        48         0   100.00%

================================Statement Details================================

Statement Coverage for Design Unit work.ALSU --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ALSU.v
    1                                                module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, c
    2                                                parameter INPUT_PRIORITY = "A";
    3                                                parameter FULL_ADDER = "ON";
    4                                                input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    5                                                input [2:0] opcode;
    6                                                input signed [2:0] A, B;
    7                                                output reg [15:0] leds;
    8                                                output reg signed [5:0] out;
    9
    10                                               reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_i
    11                                               reg signed cin_reg;
    12                                               reg [2:0] opcode_reg;
    13                                               reg signed [2:0] A_reg, B_reg;
    14
    15                                               wire invalid_red_op, invalid_opcode, invalid;
    16
    17                                               //Invalid handling
    18              1                        177     assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg
    19              1                        167     assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    20              1                         89     assign invalid = invalid_red_op | invalid_opcode;
    21
    22                                               //Registering input signals
    23              1                        402     always @(posedge clk or posedge rst) begin
    24                                                 if(rst) begin
    25              1                         28           cin_reg <= 0;
    26              1                         28           red_op_B_reg <= 0;
    27              1                         28           red_op_A_reg <= 0;
    28              1                         28           bypass_B_reg <= 0;
    29              1                         28           bypass_A_reg <= 0;
    30              1                         28           direction_reg <= 0;
    31              1                         28           serial_in_reg <= 0;
    32              1                         28           opcode_reg <= 0;
    33              1                         28           A_reg <= 0;
    34              1                          28          B_reg <= 0;
    35                                                  end else begin
    36              1                         374          cin_reg <= cin;
    37              1                         374          red_op_B_reg <= red_op_B;
    38              1                         374          red_op_A_reg <= red_op_A;
    39              1                         374          bypass_B_reg <= bypass_B;
    40              1                         374          bypass_A_reg <= bypass_A;
    41              1                         374          direction_reg <= direction;
    42              1                         374          serial_in_reg <= serial_in;
    43              1                         374          opcode_reg <= opcode;
    44              1                         374          A_reg <= A;
    45              1                         374          B_reg <= B;
    46                                                  end
    47                                                end
    48
    49                                                //leds output blinking
    50              1                         417     always @(posedge clk or posedge rst) begin
    51                                                  if(rst) begin
    52              1                          43          leds <= 0;
    53                                                  end else begin
    54                                                      if (invalid)
    55              1                         279             leds <= ~leds;
    56                                                      else
    57              1                          95             leds <= 0;
    58                                                  end
    59                                                end
    60
    61                                                //ALSU output processing
    62              1                         384     always @(posedge clk or posedge rst) begin
    63                                                  if(rst) begin
    64              1                          28         out <= 0;
    65                                                  end
    66                                                  else begin
    67                                                    if (bypass_A_reg && bypass_B_reg)
    68              1                          15           out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    69                                                    else if (bypass_A_reg)
    70              1                          77           out <= A_reg;
    71                                                    else if (bypass_B_reg)
    72              1                          46           out <= B_reg;
    73                                                    else if (invalid)
    74              1                         156             out <= 0;
    75                                                    else begin
    76                                                        case (opcode)
    77                                                          3'h0: begin
    78                                                            if (red_op_A_reg && red_op_B_reg)
    79              1                           3                 out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
    80                                                            else if (red_op_A_reg)
    81              1                           5                 out <= |A_reg;
    82                                                            else if (red_op_B_reg)
    83              1                           2                 out <= |B_reg;
    84                                                            else
    85              1                           7                 out <= A_reg | B_reg;
    86                                                          end
    87                                                          3'h1: begin
    88                                                            if (red_op_A_reg && red_op_B_reg)
    89              1                           1                 out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
    90                                                            else if (red_op_A_reg)
    91              1                           5                 out <= ^A_reg;
    92                                                            else if (red_op_B_reg)
    93              1                           5                 out <= ^B_reg;
    94                                                            else
    95              1                           7                 out <= A_reg ^ B_reg;
    96                                                          end
    97              1                           6               3'h2: out <= A_reg + B_reg;
    98              1                           7               3'h3: out <= A_reg * B_reg;
    99                                                          3'h4: begin
    100                                                           if (direction_reg)
    101             1                           1                 out <= {out[4:0], serial_in_reg};
    102                                                           else
    103             1                           4                 out <= {serial_in_reg, out[5:1]};
    104                                                         end
    105                                                         3'h5: begin
    106                                                           if (direction_reg)
    107             1                           2                 out <= {out[4:0], out[5]};
    108                                                           else
    109             1                           6                 out <= {out[0], out[5:1]};

Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        118       118         0   100.00%

================================Toggle Details================================

Toggle Coverage for Design Unit work.ALSU

                                              Node      1H->0L      0L->1H  "Coverage"
                                              ---------------------------------------
                                            A[0-2]           1           1      100.00
                                        A_reg[0-2]           1           1      100.00
                                            B[0-2]           1           1      100.00
                                        B_reg[0-2]           1           1      100.00
```

```
                                              bypass_A          1           1     100.00
                                          bypass_A_reg          1           1     100.00
                                              bypass_B          1           1     100.00
                                          bypass_B_reg          1           1     100.00
                                                   cin          1           1     100.00
                                               cin_reg          1           1     100.00
                                                   clk          1           1     100.00
                                             direction          1           1     100.00
                                         direction_reg          1           1     100.00
                                               invalid          1           1     100.00
                                        invalid_opcode          1           1     100.00
                                        invalid_red_op          1           1     100.00
                                           leds[0-15]           1           1     100.00
                                          opcode[0-2]           1           1     100.00
                                      opcode_reg[0-2]           1           1     100.00
                                             out[0-5]           1           1     100.00
                                              red_op_A          1           1     100.00
                                          red_op_A_reg          1           1     100.00
                                              red_op_B          1           1     100.00
                                          red_op_B_reg          1           1     100.00
                                                   rst          1           1     100.00
                                             serial_in          1           1     100.00
                                         serial_in_reg          1           1     100.00

Total Node Count     =         59
Toggled Node Count   =         59
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (118 of 118 bins)


Total Coverage By Design Unit (filtered view): 100.00%

Coverage Report by instance with details

=================================================================================
=== Instance: /ALSU_pkg
=== Design Unit: work.ALSU_pkg
=================================================================================

Covergroup Coverage:
    Covergroups                      1         na         na    100.00%
        Coverpoints/Crosses         11         na         na         na
            Covergroup Bins         40         40          0    100.00%
----------------------------------------------------------------------------------------------------------
Covergroup                                             Metric          Goal          Bins     Status

----------------------------------------------------------------------------------------------------------
 TYPE /ALSU_pkg/alsu_rand_class/cg                    100.00%           100             -     Covered
    covered/total bins:                                    40            40             -
    missing/total bins:                                     0            40             -
    % Hit:                                            100.00%           100             -
    Coverpoint rst                                    100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                       377             1             -     Covered
        bin auto[1]                                        26             1             -     Covered
    Coverpoint cin                                    100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                       207             1             -     Covered
        bin auto[1]                                       196             1             -     Covered
    Coverpoint red_op_A                               100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                        81             1             -     Covered
        bin auto[1]                                       322             1             -     Covered
    Coverpoint red_op_B                               100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                        71             1             -     Covered
        bin auto[1]                                       332             1             -     Covered
    Coverpoint bypass_A                               100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                       297             1             -     Covered
        bin auto[1]                                       106             1             -     Covered
    Coverpoint bypass_B                               100.00%           100             -     Covered
        covered/total bins:                                 2             2             -
        missing/total bins:                                 0             2             -
        % Hit:                                        100.00%           100             -
        bin auto[0]                                       325             1             -     Covered
        bin auto[1]                                        78             1             -     Covered
    Coverpoint opcode                                 100.00%           100             -     Covered
        covered/total bins:                                 8             8             -
        missing/total bins:                                 0             8             -
        % Hit:                                        100.00%           100             -
        bin auto[OR_0]                          50          1          -     Covered
        bin auto[XOR_1]                         40          1          -     Covered
        bin auto[ADD_2]                         66          1          -     Covered
        bin auto[MUL_3]                         92          1          -     Covered
        bin auto[SHIFT_4]                       62          1          -     Covered
        bin auto[ROTATE_5]                      70          1          -     Covered
        bin auto[INVALID_6]                     10          1          -     Covered
        bin auto[INVALID_7]                     10          1          -     Covered
    Coverpoint direction                   100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            227          1          -     Covered
        bin auto[1]                            176          1          -     Covered
    Coverpoint serial_in                   100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            201          1          -     Covered
        bin auto[1]                            202          1          -     Covered
    Coverpoint A                           100.00%        100          -     Covered
        covered/total bins:                      8          8          -
        missing/total bins:                      0          8          -
        % Hit:                             100.00%        100          -
        bin auto[-4]                            74          1          -     Covered
        bin auto[-3]                            16          1          -     Covered
        bin auto[-2]                            30          1          -     Covered
        bin auto[-1]                            30          1          -     Covered
        bin auto[0]                            110          1          -     Covered
        bin auto[1]                             36          1          -     Covered
        bin auto[2]                             38          1          -     Covered
        bin auto[3]                             66          1          -     Covered
    Coverpoint B                           100.00%        100          -     Covered
        covered/total bins:                      8          8          -
        missing/total bins:                      0          8          -
        % Hit:                             100.00%        100          -
        bin auto[-4]                            68          1          -     Covered
        bin auto[-3]                            26          1          -     Covered
        bin auto[-2]                            32          1          -     Covered
        bin auto[-1]                            24          1          -     Covered
        bin auto[0]                            102          1          -     Covered
        bin auto[1]                             48          1          -     Covered
        bin auto[2]                             34          1          -     Covered
        bin auto[3]                             66          1          -     Covered

COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------
Covergroup                                  Metric       Goal       Bins    Status

----------------------------------------------------------------------------------------------
 TYPE /ALSU_pkg/alsu_rand_class/cg         100.00%        100          -     Covered
    covered/total bins:                         40         40          -
    missing/total bins:                          0         40          -
    % Hit:                                 100.00%        100          -
    Coverpoint rst                         100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            377          1          -     Covered
        bin auto[1]                             26          1          -     Covered
    Coverpoint cin                         100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            207          1          -     Covered
        bin auto[1]                            196          1          -     Covered
    Coverpoint red_op_A                    100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                             81          1          -     Covered
        bin auto[1]                            322          1          -     Covered
    Coverpoint red_op_B                    100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                             71          1          -     Covered
        bin auto[1]                            332          1          -     Covered
    Coverpoint bypass_A                    100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            297          1          -     Covered
        bin auto[1]                            106          1          -     Covered
    Coverpoint bypass_B                    100.00%        100          -     Covered
        covered/total bins:                      2          2          -
        missing/total bins:                      0          2          -
        % Hit:                             100.00%        100          -
        bin auto[0]                            325          1          -     Covered
        bin auto[1]                             78          1          -     Covered
    Coverpoint opcode                      100.00%        100          -     Covered
        covered/total bins:                      8          8          -
        missing/total bins:                         0          8          –
        % Hit:                                100.00%        100          –
        bin auto[OR_0]                             50          1          –    Covered
        bin auto[XOR_1]                            40          1          –    Covered
        bin auto[ADD_2]                            66          1          –    Covered
        bin auto[MUL_3]                            92          1          –    Covered
        bin auto[SHIFT_4]                          62          1          –    Covered
        bin auto[ROTATE_5]                         70          1          –    Covered
        bin auto[INVALID_6]                        10          1          –    Covered
        bin auto[INVALID_7]                        10          1          –    Covered
    Coverpoint direction                      100.00%        100          –    Covered
        covered/total bins:                         2          2          –
        missing/total bins:                         0          2          –
        % Hit:                                100.00%        100          –
        bin auto[0]                               227          1          –    Covered
        bin auto[1]                               176          1          –    Covered
    Coverpoint serial_in                      100.00%        100          –    Covered
        covered/total bins:                         2          2          –
        missing/total bins:                         0          2          –
        % Hit:                                100.00%        100          –
        bin auto[0]                               201          1          –    Covered
        bin auto[1]                               202          1          –    Covered
    Coverpoint A                              100.00%        100          –    Covered
        covered/total bins:                         8          8          –
        missing/total bins:                         0          8          –
        % Hit:                                100.00%        100          –
        bin auto[-4]                               74          1          –    Covered
        bin auto[-3]                               16          1          –    Covered
        bin auto[-2]                               30          1          –    Covered
        bin auto[-1]                               30          1          –    Covered
        bin auto[0]                               110          1          –    Covered
        bin auto[1]                                36          1          –    Covered
        bin auto[2]                                38          1          –    Covered
        bin auto[3]                                66          1          –    Covered
    Coverpoint B                              100.00%        100          –    Covered
        covered/total bins:                         8          8          –
        missing/total bins:                         0          8          –
        % Hit:                                100.00%        100          –
        bin auto[-4]                               68          1          –    Covered
        bin auto[-3]                               26          1          –    Covered
        bin auto[-2]                               32          1          –    Covered
        bin auto[-1]                               24          1          –    Covered
        bin auto[0]                               102          1          –    Covered
        bin auto[1]                                48          1          –    Covered
        bin auto[2]                                34          1          –    Covered
        bin auto[3]                                66          1          –    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

 Total Coverage By Instance (filtered view): 100.00%
```

2.8 8.Waveform

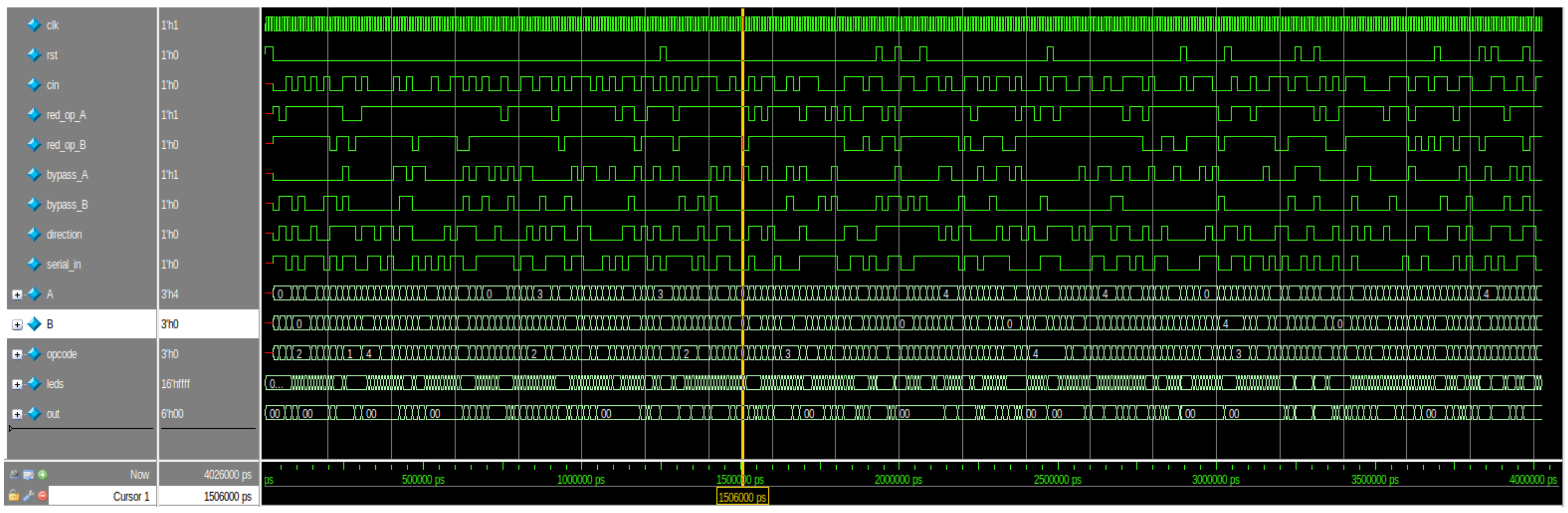
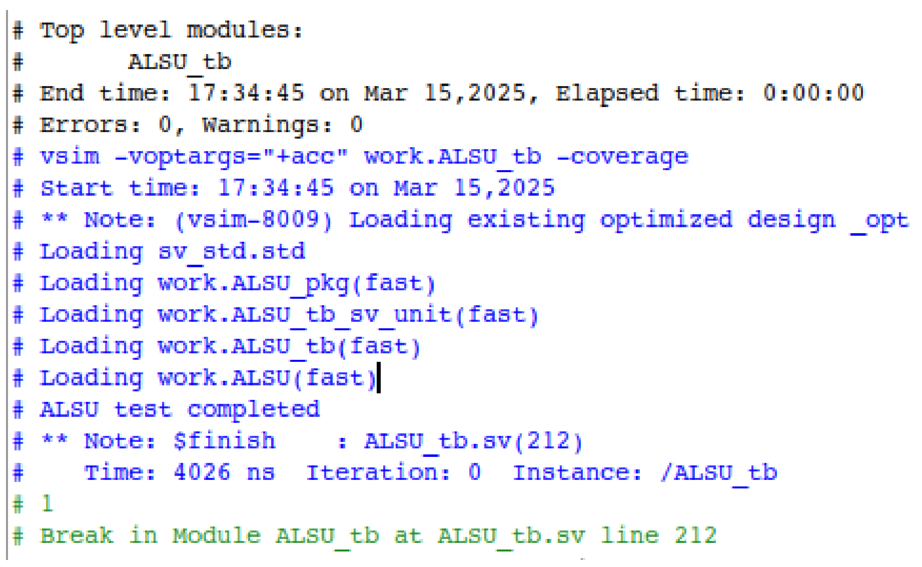


Figure 2: simulation waveform



```
# Top level modules:
#       ALSU_tb
# End time: 17:34:45 on Mar 15,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.ALSU_tb -coverage
# Start time: 17:34:45 on Mar 15,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading sv_std.std
# Loading work.ALSU_pkg(fast)
# Loading work.ALSU_tb_sv_unit(fast)
# Loading work.ALSU_tb(fast)
# Loading work.ALSU(fast)
# ALSU test completed
# ** Note: $finish    : ALSU_tb.sv(212)
#    Time: 4026 ns  Iteration: 0  Instance: /ALSU_tb
# 1
# Break in Module ALSU_tb at ALSU_tb.sv line 212
```

Figure 3: Transcript : all test cases passed

3 Q3: FSM_010

3.1 1. Testbench code

```
`timescale 1ns/1ps

import fsm_package::*;

module fsm_010_tb;
    //===========================
    // declare DUT signals
    //===========================
    logic clk, rst, x;

    logic y;
    logic [1:0] users_count;

    //==============================
    // declare golden model signals
    //==============================
    logic golden_y;
    logic [9:0] golden_users_count;

    //===========================
    // instantiate FSM "DUT"
    //===========================
    FSM_010 DUT (.*);

    //===========================
    // instantiate golden model
    //===========================
    golden_model golden_DUT (.*);

    //======================
    // object from class
    //======================
    fsm_transaction fsm;

    //======================
    // Generate Clock
    //======================
    parameter CLOCK_PERIOD = 10;
    initial begin
         clk = 0 ;
         forever begin
            #(CLOCK_PERIOD/2) clk = ~clk;
            //===========================
            // clock mapping (TB , class)
            //===========================
            fsm.clk = clk;
    end
    end

    //======================
    // reset task
    //======================
    task do_reset();
        rst = 1;
        #(CLOCK_PERIOD*2);
        rst = 0;
    endtask


  //=========================================================================================
  //================================= Method 2 tasks ========================================
  //=========================================================================================
  state_e cs, ns;

  // --------------------------
  // "Golden Model" Task
  // --------------------------
  task golden_model(input bit x_in, input bit rst_in);
    if (rst_in) begin
      cs = IDLE;
      Y = 0;
      fsm.users_count_exp = 0;
    end

    // Next-state logic
    case (cs)
      IDLE:  ns = (x_in) ? IDLE : ZERO;
      ZERO:  ns = (x_in) ? ONE  : ZERO;
      ONE:   ns = (x_in) ? IDLE : STORE;
      STORE: ns = (x_in) ? IDLE : ZERO;
    endcase

    // Update count if in STORE
    if (cs == STORE) begin
      fsm.users_count_exp++;
    end

    // y_exp is 1 in STORE
    fsm.y_exp = (cs == STORE);

    // Move to next state
    cs = ns;
  endtask

  // ---------------------------
  // "Check" Task
  // ---------------------------
  task check_result();
    // Call golden_model with the *same* inputs we just applied
    golden_model(x, rst);

    fsm.y_exp   = y;
    fsm.users_count_exp = users_count;

    // Now compare
    if ((y !== fsm.y_exp) ||
        (users_count !== fsm.users_count_exp)) begin
      $error("Mismatch:␣time=%0t,␣y=%0b␣vs␣%0b,␣users_count=%0d␣vs␣%0d",
              $time, y, fsm.y_exp, users_count, fsm.users_count_exp);
    end
    else begin
      $display("FSM_010␣Match␣golden␣model␣task␣:␣y=%0b,␣user_count=%0d",y ,users_count);
    end
  endtask



    initial begin
        fsm = new();

        do_reset();

        //=============================
        // Method 1 of self checking
        //=============================
        repeat(10) begin
            //=======================
            // assert randomization
            //=======================
            if(!fsm.randomize())begin
               $error("Randomization␣failed!");
               $finish;
            end

           //==================
           // Drive signals
           //==================
           rst = fsm.rst;
           x   = fsm.x;
           fsm.y_exp   = y;
           fsm.users_count_exp = users_count;

           //============================================
           // compare outputs of golden model and design
           //============================================
           assert (y == golden_y || users_count == golden_users_count)
               $display("␣fsm_010␣match␣golden␣model");
           else
               $error("there␣is␣a␣mismatch␣,␣fsm_010␣gets␣y␣:␣%0b␣,users_count␣:␣%0b␣,␣golden␣model␣gets␣y␣:␣%0b␣,users_count␣:␣%0b␣",y
                   ,users_count ,golden_y ,golden_users_count);

           @(posedge clk);
        end

       do_reset();

       //=============================
       // Method 2 of self checking
       //=============================
       repeat(100) begin
            //=======================
            // assert randomization
            //=======================
            if (!fsm.randomize()) begin
               $error("Randomization␣failed!");
               $finish;
            end

            //==================
            // Drive signals
            //==================
            x   = fsm.x;
            rst = fsm.rst;

            //==================
            // check results
            //==================
            check_result();

            @(posedge clk);
        end

        $display("All␣tests␣done.");
        $finish;
    end
endmodule
```

3.2 2. Package code

```
package fsm_package ;

    typedef enum {IDLE ,ZERO ,ONE ,STORE} state_e;

    class fsm_transaction;
        //============================================================
        // declare inputs for randomization
        // declare with them clk signal and connect it to dut clk in TB
        //=============================================================
        bit clk;
        rand bit x;
        rand bit rst;
        bit y_exp;
        bit [1:0] users_count_exp;


        bit [1:0] prev;  // store the last two bits

       //==================================================
       // constrain rst to activate most of the time
       // use probability constrain "dist"
       // estimate 90% for deactivation and 10% activated
       //==================================================
        constraint deactivate_rst_mt {
            rst dist {0:/90 , 1:/10};
        }

       //=======================================================
       // constrain x to be zero for 67% of randomization
       // use probability constrain "dist"
       // If the previous two bits were '01', favor a '0' next
       //======================================================
        constraint x_67_0{
            if (prev != 2'b01) {
                x dist {0:/ 67 , 1:/33};
            }
        }

        constraint c_010_bias {
           if (prev == 2'b01) {
              x dist {0 :/ 100, 1 :/ 0};  // 100% chance of 0
           }
       }

      // Keep track of previous bits
      function void pre_randomize();
          // shift in the new bit as   previous   history
          prev = {prev[0], x};
      endfunction


       //================
       // coverpoints
       //================
       covergroup cg @(posedge clk);
          cp1: coverpoint x;
          cp2: coverpoint rst;
          cp3: coverpoint y_exp;
          cp4: coverpoint users_count_exp;
       endgroup

       //================
       // counstructor
       //================
       function new();
          cg = new();
       endfunction

    endclass
endpackage
```

3.3 3. Design code

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: 010-sequence-detector Design
//
////////////////////////////////////////////////////////////////////////////////
module FSM_010(clk, rst, x, y, users_count);
    parameter IDLE  = 2'b00;
    parameter ZERO  = 2'b01;
    parameter ONE   = 2'b10;
    parameter STORE = 2'b11;

    input clk, rst, x;
    output y;
    output reg [1:0] users_count;

    reg [1:0] cs, ns;

    always @(*) begin
        case (cs)
            IDLE:
                if (x)
                    ns = IDLE;
                else
                    ns = ZERO;
            ZERO:
                if (x)
                    ns = ONE;
                else
                    ns = ZERO;
            ONE:
                if (x)
                    ns = IDLE;
                else
                    ns = STORE;
            STORE:
                if (x)
                    ns = IDLE;
                else
                    ns = ZERO;
            default:    ns = IDLE;
        endcase
    end

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            cs <= IDLE;
        end
        else begin
            cs <= ns;
        end
    end

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            users_count <= 0;
        end
        else begin
            if (cs == STORE)
                users_count <= users_count + 1;
        end
    end

    assign y = (cs == STORE)? 1:0;

endmodule
```

3.4 4. Golden model code

```
//************************************************************************************
// this is golden_model foe design FSM_010
// FSM_010 from moore state machin which mean output logic does not depend on input
// this fsm detect 010 sequence
//***********************************************************************************
module golden_model (
    //=================
    // i/o declaration
    //=================
    input wire clk,
    input wire rst,
    input wire x,
    output reg golden_y,
    output reg [1:0] golden_users_count
);


//======================================
// "next_state ,current_state" Datatype
//======================================
typedef enum logic [1:0] {
    IDLE  = 2'b00,
    ZERO  = 2'b01,
    ONE   = 2'b10,
    STORE = 2'b11
} state_e;

state_e next_state ,current_state ;


//=====================================
// sequential always "present state"
//=====================================
always @(posedge clk or posedge rst) begin
  if(rst) begin
      current_state <= IDLE;
  end else begin
      current_state <= next_state;
  end
```

```
end


//=========================================
// combinational always "next state logic"
//=========================================
always @(*) begin
   case (current_state)
      IDLE : begin
             if (x)
         next_state = IDLE;
          else
         next_state = ZERO;
      end
      ZERO : begin
            if (x)
         next_state = ONE;
        else
         next_state = ZERO;
      end
      ONE : begin
           if (x)
         next_state = IDLE;
       else
         next_state = STORE;
      end
      STORE : begin
             if (x)
         next_state = IDLE;
         else
         next_state = ZERO;
      end
   endcase
end

//=========================================
// combinational always "output logic"
//=========================================
always @(*) begin
   case (current_state)
      IDLE :
            golden_y = 0;
      ZERO :
            golden_y = 0;
      ONE :
            golden_y = 0;
      STORE :
            golden_y = 1;
   endcase
end

//=========================================
// sequential always "counter logic"
// count up every time fsm detect 010
//=========================================
always @(posedge clk or posedge rst) begin
   if(rst) begin
        golden_users_count <= 0;
   end else begin
       if (current_state == STORE)
       golden_users_count <= golden_users_count + 1;
   end
end


endmodule
```

3.5 5. Bug Fixes

```
no bugs
and I change size of user_count to 2 bits to made coverage reach 100% and that not a bug
```

3.6 6. Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FSM_010_1	**Reset Behavior:** When the active-high reset is asserted, the FSM should move to *IDLE*, and `users_count` should be cleared.	• Directed at the start of the simulation (assert `rst`) • Constrained-random with `rst` mostly off, but occasionally on	• Cover reset transitions (`rst=1` leading to *IDLE*) • Check *IDLE* coverage immediately after reset	A checker (or golden model) ensures `users_count = 0` and `y = 0` after reset. Verifies FSM is in *IDLE* state on release of reset.
FSM_010_2	**Pattern Detection:** When the sequence `010` is seen, the FSM outputs `y=1` and increments `users_count`.	• Constrained-random on `x` (67% zeros) • Additional bias if previous bits are `01` (to favor forming `010`)	• State coverage: *IDLE, ZERO, ONE, STORE* • Transition coverage: *IDLE* → *ZERO*, etc.	Compare `y` and `users_count` with a golden model or via task-based checking. Ensure exactly one increment per `010`.

Table 2: Verification Plan for the FSM_010 Design

3.7 7. Do File

```
vlib work
vlog fsm_package.sv FSM_010.v fsm_010_tb.sv golden_model.sv +cover -covercells
vsim -voptargs=+acc work.fsm_010_tb -cover
coverage exclude -src FSM_010.v -line 42
# I exclude default case in fsm design as it never go through this case
add wave *
coverage save fsm_010_tb.ucdb -onexit
run -all

# to run do file
-- do run.txt
to execute coverage report
-- vcover report fsm_010_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=FSM_010
-- vcover report -details -cvg -output fsm_coverage_report.txt fsm_010_tb.ucdb
```

3.8 7. Coverage Report

```
Coverage Report by DU with details

=================================================================================
=== Design Unit: work.FSM_010
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        20        20         0   100.00%

================================Branch Details================================

Branch Coverage for Design Unit work.FSM_010

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FSM_010.v
------------------------------------CASE Branch------------------------------------
    21                                       128     Count coming in to CASE
    22             1                          39                         IDLE:
    27             1                          50                         ZERO:
    32             1                          22                         ONE:
    37             1                          17                         STORE:
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    23                                        39     Count coming in to IF
    23             1                          14                                 if (x)
    25             1                          25                                 else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    28                                        50     Count coming in to IF
    28             1                          19                                 if (x)
    30             1                          31                                 else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                        22     Count coming in to IF
    33             1                           7                                 if (x)
    35             1                          15                                 else
Branch totals: 2 hits of 2 branches = 100.00%
```

```
----------------------------------------IF Branch----------------------------------------
    38                                        17     Count coming in to IF
    38              1                          7                                    if (x)
    40              1                         10                                    else
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------------IF Branch----------------------------------------
    47                                       109     Count coming in to IF
    47              1                         28                if(rst) begin
    50              1                         81                else begin
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------------IF Branch----------------------------------------
    56                                        96     Count coming in to IF
    56              1                         28                if(rst) begin
    59              1                         68                else begin
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------------IF Branch----------------------------------------
    60                                        68     Count coming in to IF
    60              1                         10                           if (cs == STORE)
                                              58     All False Count
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------------IF Branch----------------------------------------
    65                                        67     Count coming in to IF
    65              1                         10        assign y = (cs == STORE)? 1:0;
    65              2                         57        assign y = (cs == STORE)? 1:0;
Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins    Covered     Misses   Coverage
    ----------------              ----    -------     ------   --------
    Conditions                       2          2          0    100.00%

================================Condition Details================================

Condition Coverage for Design Unit work.FSM_010 --

  File FSM_010.v
----------------Focused Condition View--------------------
Line       60 Item    1  (cs == 3)
Condition totals: 1 of 1 input term covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
   (cs == 3)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (cs == 3)_0           -
  Row   2:          1  (cs == 3)_1           -

----------------Focused Condition View--------------------
Line       65 Item    1  (cs == 3)
Condition totals: 1 of 1 input term covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
   (cs == 3)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (cs == 3)_0           -
  Row   2:          1  (cs == 3)_1           -


FSM Coverage:
    Enabled Coverage              Bins      Hits     Misses   Coverage
    ----------------              ----      ----     ------   --------
    FSM States                       4         4          0    100.00%
    FSM Transitions                  7         7          0    100.00%

================================FSM Details================================

FSM Coverage for Design Unit work.FSM_010 --

FSM_ID: cs
    Current State Object : cs
    ----------------------
    State Value MapInfo :
    ---------------------
Line          State Name               Value
----          ----------               -----
  22                IDLE                   0
  27                ZERO                   1
  32                 ONE                   2
  37               STORE                   3
    Covered States :
    ----------------
```

```
                    State            Hit_count
                    -----            ---------
                     IDLE                   38
                     ZERO                   33
                      ONE                   15
                    STORE                   11
    Covered Transitions :
    ---------------------
Line             Trans_ID            Hit_count          Transition
----             --------            ---------          ----------
  26                    0                   18          IDLE -> ZERO
  29                    1                   15          ZERO -> ONE
  48                    2                    7          ZERO -> IDLE
  36                    3                   11          ONE -> STORE
  34                    4                    4          ONE -> IDLE
  41                    5                    4          STORE -> ZERO
  39                    6                    6          STORE -> IDLE


    Summary                            Bins       Hits    Misses   Coverage
    -------                            ----       ----    ------   --------
        FSM States                        4          4         0    100.00%
        FSM Transitions                   7          7         0    100.00%
Statement Coverage:
    Enabled Coverage                   Bins       Hits    Misses   Coverage
    ----------------                   ----       ----    ------   --------
    Statements                           16         16         0    100.00%

================================Statement Details================================

Statement Coverage for Design Unit work.FSM_010 --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FSM_010.v
    8                                                module FSM_010(clk, rst, x, y, users_count);
    9                                                   parameter IDLE  = 2'b00;
    10                                                  parameter ZERO  = 2'b01;
    11                                                  parameter ONE   = 2'b10;
    12                                                  parameter STORE = 2'b11;
    13
    14                                                  input clk, rst, x;
    15                                                  output y;
    16                                                  output reg [1:0] users_count;
    17
    18                                                  reg [1:0] cs, ns;
    19
    20              1                        128        always @(*) begin
    21                                                          case (cs)
    22                                                                  IDLE:
    23                                                                          if (x)
    24              1                         14                                        ns = IDLE;
    25                                                                          else
    26              1                         25                                        ns = ZERO;
    27                                                                  ZERO:
    28                                                                          if (x)
    29              1                         19                                        ns = ONE;
    30                                                                          else
    31              1                         31                                        ns = ZERO;
    32                                                                  ONE:
    33                                                                          if (x)
    34              1                          7                                        ns = IDLE;
    35                                                                          else
    36              1                         15                                        ns = STORE;
    37                                                                  STORE:
    38                                                                          if (x)
    39              1                          7                                        ns = IDLE;
    40                                                                          else
    41              1                         10                                        ns = ZERO;
    42                                                                  default:        ns = IDLE;
    43                                                          endcase
    44                                                  end
    45
    46              1                        109        always @(posedge clk or posedge rst) begin
    47                                                          if(rst) begin
    48              1                         28                        cs <= IDLE;
    49                                                          end
    50                                                          else begin
    51              1                         81                        cs <= ns;
    52                                                          end
    53                                                  end
    54
    55              1                         96        always @(posedge clk or posedge rst) begin
    56                                                          if(rst) begin
    57              1                         28                        users_count <= 0;
    58                                                          end
    59                                                          else begin
    60                                                                  if (cs == STORE)
    61              1                         10                                users_count <= users_count + 1;
    62                                                          end
    63                                                  end
    64
    65              1                       68        assign y = (cs == STORE)? 1:0;

Toggle Coverage:
    Enabled Coverage               Bins       Hits     Misses  Coverage
    ----------------               ----       ----     ------  --------
    Toggles                          20         20          0   100.00%

================================Toggle Details================================

Toggle Coverage for Design Unit work.FSM_010

                                              Node       1H->0L       0L->1H                          "Coverage"
                                              ---------------------------------------------------------------------
                                               clk            1            1                              100.00
                                           cs[0-1]            1            1                              100.00
                                           ns[0-1]            1            1                              100.00
                                               rst            1            1                              100.00
                                  users_count[0-1]            1            1                              100.00
                                                 x            1            1                              100.00
                                                 y            1            1                              100.00

Total Node Count      =         10
Toggled Node Count    =         10
Untoggled Node Count  =          0

Toggle Coverage       =     100.00% (20 of 20 bins)


Total Coverage By Design Unit (filtered view): 100.00%


Coverage Report by instance with details

===============================================================================
=== Instance: /fsm_package
=== Design Unit: work.fsm_package
===============================================================================

Covergroup Coverage:
    Covergroups                       1         na         na    100.00%
        Coverpoints/Crosses           4         na         na         na
            Covergroup Bins          10         10          0    100.00%
----------------------------------------------------------------------------------------------------------
Covergroup                                             Metric        Goal        Bins      Status

----------------------------------------------------------------------------------------------------------
 TYPE /fsm_package/fsm_transaction/cg                 100.00%         100           -      Covered
    covered/total bins:                                    10          10           -
    missing/total bins:                                     0          10           -
    % Hit:                                            100.00%         100           -
    Coverpoint cp1                                    100.00%         100           -      Covered
        covered/total bins:                                 2           2           -
        missing/total bins:                                 0           2           -
        % Hit:                                        100.00%         100           -
        bin auto[0]                                        85           1           -      Covered
        bin auto[1]                                        30           1           -      Covered
    Coverpoint cp2                                    100.00%         100           -      Covered
        covered/total bins:                                 2           2           -
        missing/total bins:                                 0           2           -
        % Hit:                                        100.00%         100           -
        bin auto[0]                                       103           1           -      Covered
        bin auto[1]                                        12           1           -      Covered
    Coverpoint cp3                                    100.00%         100           -      Covered
        covered/total bins:                                 2           2           -
        missing/total bins:                                 0           2           -
        % Hit:                                        100.00%         100           -
        bin auto[0]                                       105           1           -      Covered
        bin auto[1]                                        10           1           -      Covered
    Coverpoint cp4                                    100.00%         100           -      Covered
        covered/total bins:                                 4           4           -
        missing/total bins:                                 0           4           -
        % Hit:                                        100.00%         100           -
        bin auto[0]                                        79           1           -      Covered
        bin auto[1]                                        29           1           -      Covered
        bin auto[2]                                         6           1           -      Covered
        bin auto[3]                                         1           1           -      Covered

COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------
Covergroup                                             Metric        Goal        Bins      Status

----------------------------------------------------------------------------------------------------------
 TYPE /fsm_package/fsm_transaction/cg                 100.00%         100           -      Covered
    covered/total bins:                                    10          10           -
    missing/total bins:                                     0          10           -
    % Hit:                                            100.00%         100           -
    Coverpoint cp1                                    100.00%         100           -      Covered
        covered/total bins:                                 2           2           -
        missing/total bins:                                 0           2           -
        % Hit:                                        100.00%         100           -
        bin auto[0]                                        85           1           -      Covered
        bin auto[1]                                        30           1           -      Covered
    Coverpoint cp2                               100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  103          1          -    Covered
        bin auto[1]                                   12          1          -    Covered
    Coverpoint cp3                               100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  105          1          -    Covered
        bin auto[1]                                   10          1          -    Covered
    Coverpoint cp4                               100.00%        100          -    Covered
        covered/total bins:                            4          4          -
        missing/total bins:                            0          4          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                   79          1          -    Covered
        bin auto[1]                                   29          1          -    Covered
        bin auto[2]                                    6          1          -    Covered
        bin auto[3]                                    1          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

 Total Coverage By Instance (filtered view): 100.00%
```

3.9 8.Waveform

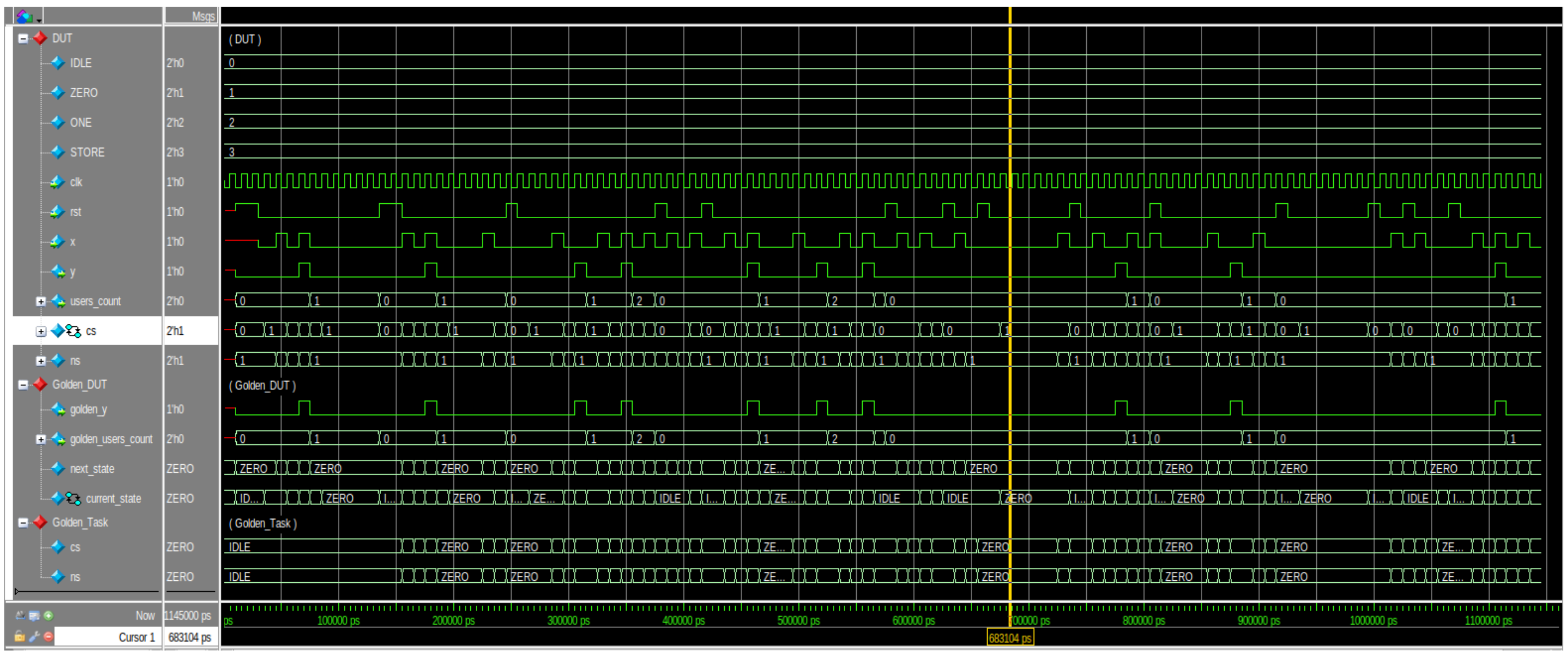
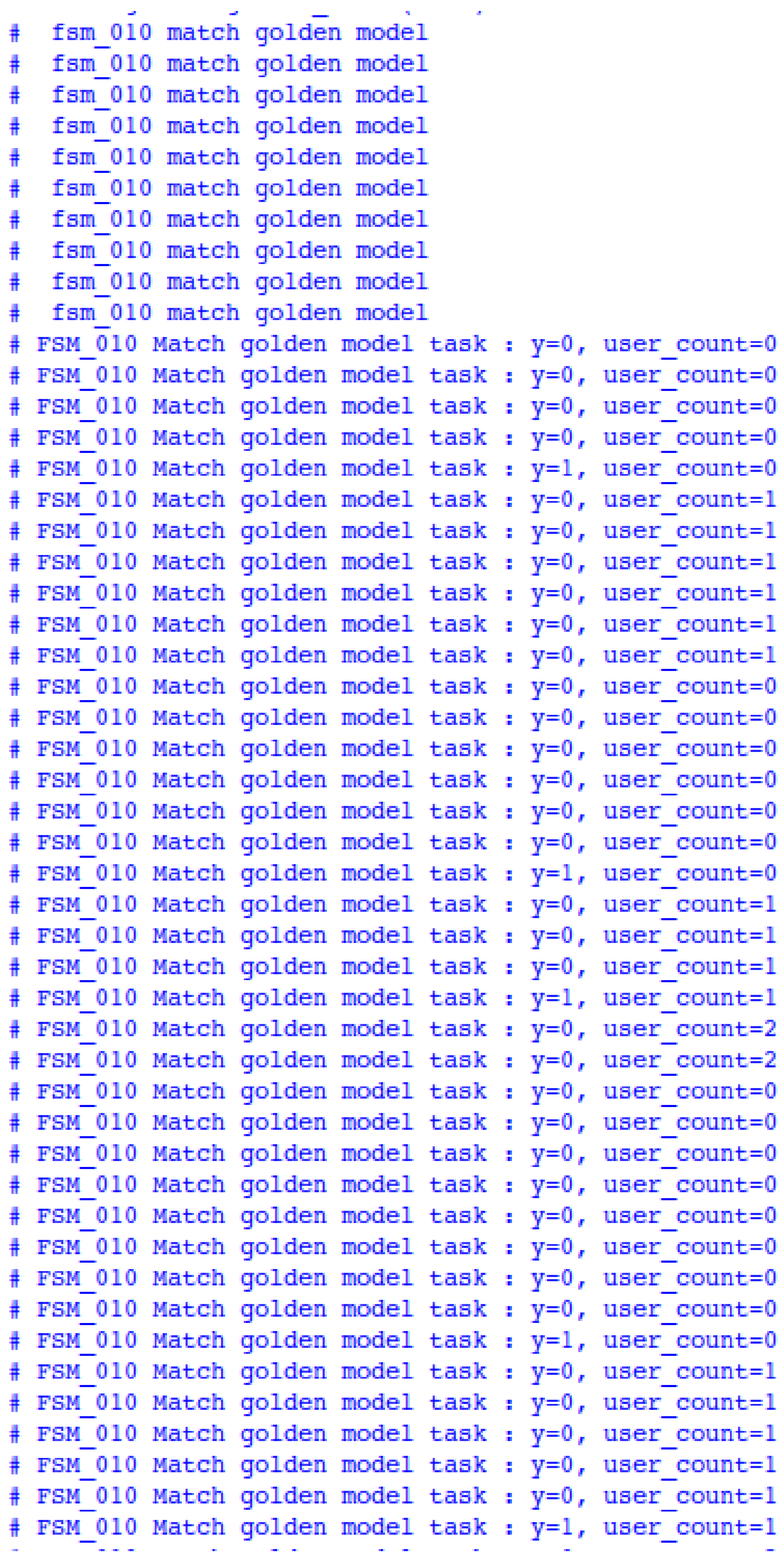


Figure 4: simulation waveform



```
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
#  fsm_010 match golden model
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=1, user_count=0
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=1, user_count=0
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=1, user_count=1
# FSM_010 Match golden model task : y=0, user_count=2
# FSM_010 Match golden model task : y=0, user_count=2
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=0, user_count=0
# FSM_010 Match golden model task : y=1, user_count=0
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=0, user_count=1
# FSM_010 Match golden model task : y=1, user_count=1
```

Figure 5: Transcript : all test cases passed