P1)

- Code

```
module two_state_array ();
    bit [11:0] my_array [4] ;
    initial begin
        my_array[0] = 12'h012 ;
        my_array[1] = 12'h345 ;
        my_array[2] = 12'h678 ;
        my_array[3] = 12'h9AB ;

        for (int i_for = 0 ; i_for < 4 ; i_for++ ) begin
            for (int j_for ; j_for < 12 ; j_for++) begin
                if (j_for == 5 ) $display("bit 5  of array using for loop  %d is %b " , i_for , my_array[i_for][5] );
                if (j_for == 4 ) $display("bit 4  of array using for loop  %d is %b " , i_for , my_array[i_for][4] );
            end
        end

        foreach (my_array[i_foreach]) begin
            foreach (my_array[i_foreach][j_foreach]) begin
                if (j_foreach == 5) $display("bit 5  of array using foreach loop %d is %b " , i_foreach , my_array[i_foreach][5] );
                if (j_foreach == 4) $display("bit 4  of array using foreach loop %d is %b " , i_foreach , my_array[i_foreach][4] );
            end
        end

    end
endmodule
```

- Transcript

```
# bit 4  of array using for loop          0 is 1
# bit 5  of array using for loop          0 is 0
# bit 4  of array using for loop          1 is 0
# bit 5  of array using for loop          1 is 0
# bit 4  of array using for loop          2 is 1
# bit 5  of array using for loop          2 is 1
# bit 4  of array using for loop          3 is 0
# bit 5  of array using for loop          3 is 1
# bit 5  of array using foreach loop         0 is 0
# bit 4  of array using foreach loop         0 is 1
# bit 5  of array using foreach loop         1 is 0
# bit 4  of array using foreach loop         1 is 0
# bit 5  of array using foreach loop         2 is 1
# bit 4  of array using foreach loop         2 is 1
# bit 5  of array using foreach loop         3 is 1
# bit 4  of array using foreach loop         3 is 0
```

P2)

- Testbench

```
import ALU_package::*;
module ALU_tb ();
    reg signed [3:0] A_tb ,B_tb ;
    reg clk_tb ;
    reg rst_tb ;
    reg [1:0] opcode_tb ;
    wire signed [4:0] C_tb ;
    reg  signed [4:0] expected_out ;

    localparam MAXPOS = 7 ;
    localparam MAXNEG = -8 ;

    integer error_count , correct_count ;

    initial begin
        clk_tb = 0;
        forever begin
            #2 clk_tb = ~clk_tb;
        end
    end

    ALU_4_bit DUT (clk_tb,rst_tb,opcode_tb,A_tb,B_tb,C_tb) ;

    initial begin
        ALU_class obj1 = new ;
        rst_tb = 0 ;
        correct_count = 0 ;
        error_count = 0 ;
        // TEST_1
        assert_reset () ;

        repeat (200) begin
            assert (obj1.randomize()) else $fatal("Randomization failed");
            rst_tb       = obj1.rst       ;
            A_tb         = obj1.A         ;
            B_tb         = obj1.B         ;
            opcode_tb    = obj1.opcode    ;
            check_result () ;
        end

        opcode_tb = 2'bxx;
        check_result () ;

        assert_reset();

        $display ("correct counter = %d , error counter = %d " , correct_count , error_count ) ;
        $stop ;
    end

    task check_result ();
        golden_model () ;
        @ (negedge clk_tb) ;
        if (expected_out == C_tb )begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
    endtask

    task golden_model ();
        if (rst_tb) expected_out = 0 ;
        else begin
            case (opcode_tb)
                2'b00:   expected_out = A_tb + B_tb;
                2'b01:   expected_out = A_tb - B_tb;
                2'b10:   expected_out = ~A_tb;
                2'b11:   expected_out = |B_tb;
                default: expected_out = 5'b0;
            endcase
        end
    endtask

    task assert_reset ();
        rst_tb = 1 ;
        @ (negedge clk_tb) ;
        if (C_tb == 0 )begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
        rst_tb = 0 ;
    endtask

endmodule
```

- Package

```
package ALU_package ;
    localparam MAXPOS =  7 ;
    localparam MAXNEG = -8 ;
    class ALU_class;
        typedef enum logic [1:0] { Add , Sub , Not_A , ReductionOR_B  } opcode_t ;
        randc logic signed [3:0] A ,B ;
        rand logic rst ;
        randc opcode_t opcode ;

        constraint RESET {
            rst dist {0:/99 , 1:/1} ;
        }

    endclass
endpackage
```

- Verification plan

| Label | Design Requirement Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|
| TEST_1 | When the reset is asserted, the output value should be low | Directed at the start of the simulation | | A checker in the testbench to make sure the output is correct |
| TEST_2 | Take 200 different input values from constraints , and chech the output with the golden model | Randomized | | A checker in the testbench to make sure the output is correct |
| TEST_3 | When the reset is asserted, the output value should be low | Directed at the end of the simulation | | A checker in the testbench to make sure the output is correct |

- DO file

```
vlib work
vlog ALU.v ALU_tb.svh package.svh  +cover -covercells
vsim -voptargs=+acc work.ALU_tb -cover
add wave *
coverage save ALU_4_bit.ucdb -onexit -du ALU_4_bit
run -all
```

- Design

```
module ALU_4_bit (
    input  clk,
    input  reset,
    input  [1:0] Opcode,   // The opcode
    input  signed [3:0] A, // Input data A in 2's complement
    input  signed [3:0] B, // Input data B in 2's complement

    output reg signed [4:0] C // ALU output in 2's complement

        );

   reg signed [4:0]      Alu_out; // ALU output in 2's complement

   localparam         Add             = 2'b00; // A + B
   localparam         Sub             = 2'b01; // A - B
   localparam         Not_A            = 2'b10; // ~A
   localparam         ReductionOR_B  = 2'b11; // |B

   // Do the operation
   always @* begin
      case (Opcode)
         Add:            Alu_out = A + B;
         Sub:            Alu_out = A - B;
         Not_A:          Alu_out = ~A;
         ReductionOR_B:  Alu_out = |B;
         default:  Alu_out = 5'b0;
      endcase
   end // always @ *

   // Register output C
   always @(posedge clk or posedge reset) begin
      if (reset)
        C <= 5'b0;
      else
        C<= Alu_out;
   end

endmodule

```

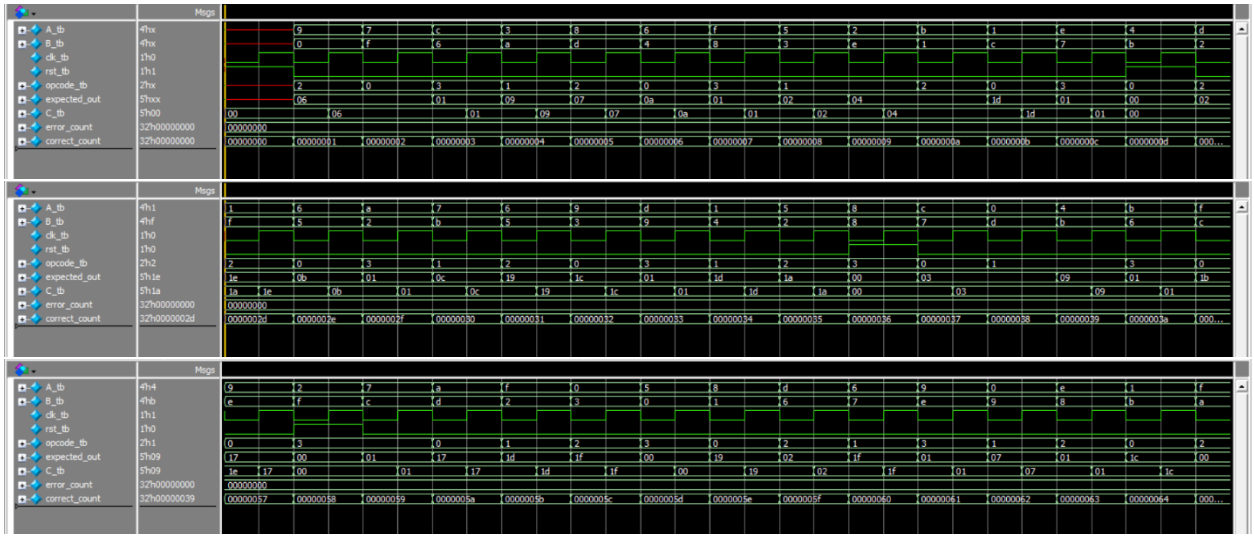
- Coverage report

```
# ================================================================================
# === Instance: /\ALU_tb#DUT
# === Design Unit: work.ALU_4_bit
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                         7         7         0   100.00%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\ALU_tb#DUT
#
#     Line         Item                     Count     Source
#     ----         ----                     -----     ------
#   File ALU.v
# ------------------------------------CASE Branch------------------------------------
#     21                                      201     Count coming in to CASE
#     22              1                        50             Add:            Alu_out = A + B;
#
#     23              1                        50             Sub:            Alu_out = A - B;
#
#     24              1                        50             Not_A:          Alu_out = ~A;
#
#     25              1                        50             ReductionOR_B:  Alu_out = |B;
#
#     26              1                         1             default:  Alu_out = 5'b0;
#
# Branch totals: 5 hits of 5 branches = 100.00%
```

```
#
# ------------------------------------IF Branch------------------------------------
#     32                                      205     Count coming in to IF
#     32              1                        10           if (reset)
#
#     34              1                       195           else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
#
# Statement Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Statements                       9         9         0   100.00%
#
```

```
#
# Toggle Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Toggles                         44        44         0   100.00%
#
# ================================Toggle Details================================
#
# Toggle Coverage for instance /\ALU_tb#DUT  --
#
#                                           Node      1H->0L      0L->1H  "Coverage"
#                                           ----------------------------------------
#                                         A[0-3]           1           1      100.00
#                                    Alu_out[4-0]          1           1      100.00
#                                         B[0-3]           1           1      100.00
#                                         C[4-0]           1           1      100.00
#                                     Opcode[0-1]          1           1      100.00
#                                            clk           1           1      100.00
#                                          reset           1           1      100.00
#
# Total Node Count     =         22
# Toggled Node Count   =         22
# Untoggled Node Count =          0
#
# Toggle Coverage      =     100.00% (44 of 44 bins)
#
```

- Questasim snippets



P3)

- Testbench

```
import FSM_package::*;
module FSM_tb();
    reg X_tb ;
    reg clk_tb ;
    reg rst_tb ;
    wire Y_tb ;
    wire [9:0] count_tb ;

    reg Y_expected ;
    reg [9:0] count_expected ;
    state_e cs_tb , ns_tb ;
    reg Y_passed , count_passed ;

    FSM_010 DUT (clk_tb, rst_tb, X_tb, Y_tb, count_tb) ;

    integer error_count , correct_count ;

    initial begin
        clk_tb = 0;
        forever begin
            #2 clk_tb = ~clk_tb;
        end
    end

    initial begin
        fsm_transaction obj = new ;
        rst_tb = 0 ;
        correct_count = 0 ;
        error_count = 0 ;
        assert_reset () ;

        repeat (1000) begin
            assert (obj.randomize()) else $fatal("Randomization failed");
            X_tb = obj.X ;
            rst_tb = obj.rst ;
            check_result () ;
        end

        assert_reset () ;

        $display ("correct counter = %d , error counter = %d " , correct_count , error_count ) ;
        $stop ;
    end

    task assert_reset ();
        rst_tb = 1 ;
        Y_expected = 0 ;
        count_expected = 0 ;
        ns_tb = IDLE ;
        @(negedge clk_tb) ;
        if (Y_expected == Y_tb && count_expected == count_tb) begin
            $display ("Test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("Test fails") ;
            error_count = error_count + 1 ;
        end
        rst_tb = 0 ;
    endtask

    task golden_model ();
        if (rst_tb) begin
            count_expected = 0 ;
            ns_tb = IDLE ;
        end
        else begin
            case (cs_tb)
                IDLE: begin
                    if (X_tb) ns_tb = IDLE ;
                    else ns_tb = ZERO ;
                end
                ZERO: begin
                    if (X_tb) ns_tb = ONE ;
                    else ns_tb = ZERO ;
                end
                ONE: begin
                    if (X_tb) ns_tb = IDLE ;
                    else ns_tb = STORE ;
                end
                STORE : begin
                    count_expected = count_expected + 1 ;
                    if (X_tb) ns_tb = IDLE ;
                    else ns_tb = ZERO ;
                end
            endcase
        end
    endtask

    task check_result();
        if (cs_tb == STORE) Y_expected = 1 ;
        else Y_expected = 0 ;
        if (Y_expected == Y_tb) Y_passed = 1 ;
        else Y_passed = 0 ;
        golden_model () ;
        @(posedge clk_tb);
        cs_tb = ns_tb ;
        @(negedge clk_tb) ;
        if (count_expected == count_tb) count_passed = 1 ;
        else count_passed = 0 ;
        if (Y_passed && count_passed ) begin
            $display ("Test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("Test fails") ;
            error_count = error_count + 1 ;
        end
    endtask
endmodule
```

- Package code

```
package FSM_package;
    typedef enum logic [1:0] { IDLE , ZERO , ONE , STORE } state_e ;
    class fsm_transaction;
        rand logic X ;
        rand logic rst ;

        constraint RESET {
            rst dist {0:/97 , 1:/3} ;
        }

        constraint X_rand {
            X dist {0:/67 , 1:/33} ;
        }

    endclass
endpackage
```

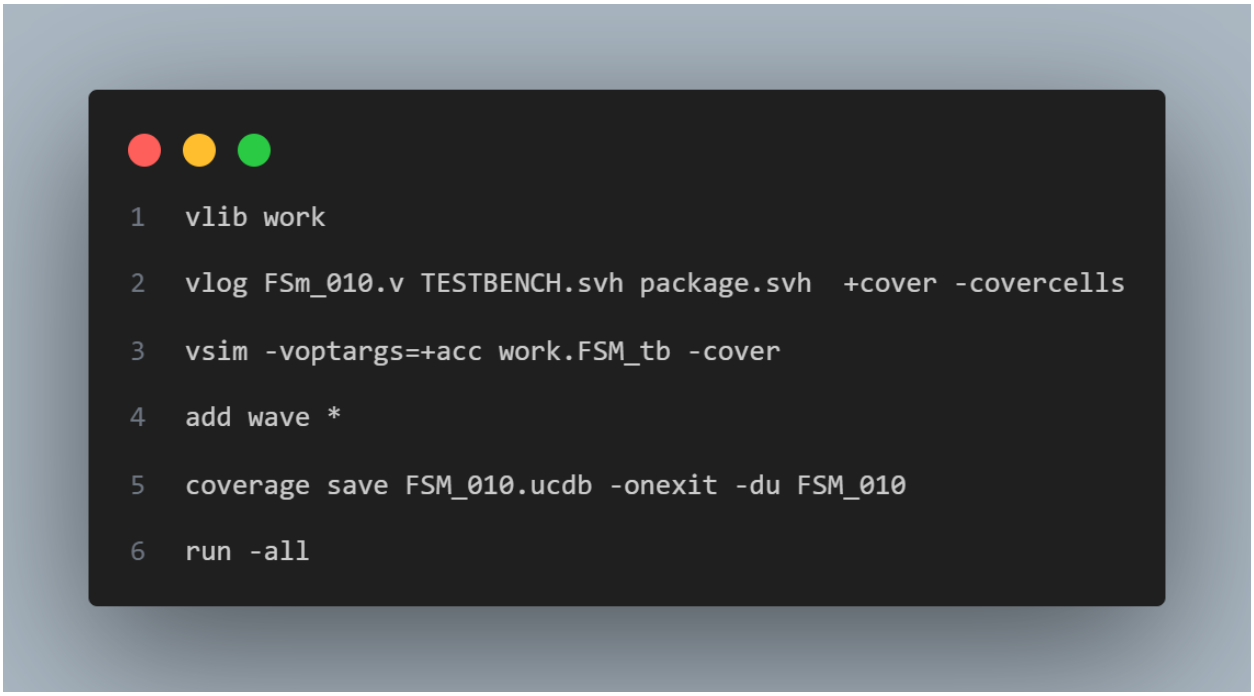
- Verification plan

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Label** | **Design Requirement Description** | **Stimulus Generation** | **Functional Coverage** | **Functionality Check** |
| 2 | TEST_1 | When the reset is asserted, the output value should be low | Directed at the start of the simulation | | A checker in the testbench to make sure the output is correct |
| 3 | TEST_2 | Take 1000 different input values from constraints , and chech the output with the golden model | Randomized | | A checker in the testbench to make sure the output is correct |
| 4 | TEST_3 | When the reset is asserted, the output value should be low | Directed at the end of the simulation | | A checker in the testbench to make sure the output is correct |

- 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 [9:0] 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
```

- DO file



```
vlib work
vlog FSm_010.v TESTBENCH.svh package.svh  +cover -covercells
vsim -voptargs=+acc work.FSM_tb -cover
add wave *
coverage save FSM_010.ucdb -onexit -du FSM_010
run -all
```

- Coverage report

```
#
# ================================================================================
# === Instance: /\FSM_tb#DUT
# === Design Unit: work.FSM_010
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                        20        20         0    100%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\FSM_tb#DUT
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File FSm_010.v
# ------------------------------------CASE Branch------------------------------------
#     21                                     1002     Count coming in to CASE
#     22              1                       232                       IDLE:
#
#     27              1                       359                       ZERO:
#
#     32              1                       278                       ONE:
#
#     37              1                       133                       STORE:
#
#
# Branch totals: 4 hits of 4 branches = 100.00%
#
```

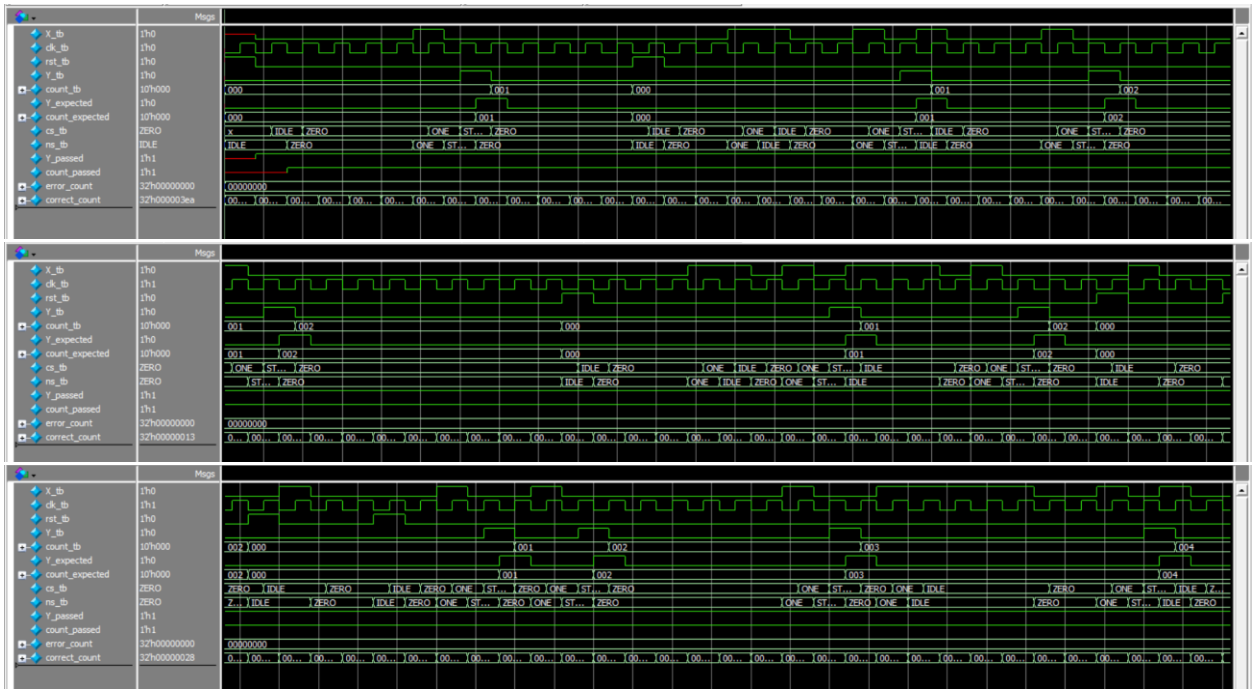
```
#
# ------------------------------------IF Branch------------------------------------
#     23                                    232     Count coming in to IF
#     23              1                     107                               if (x)
#
#     25              1                     125                               else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     28                                    359     Count coming in to IF
#     28              1                     174                               if (x)
#
#     30              1                     185                               else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     33                                    278     Count coming in to IF
#     33              1                     173                               if (x)
#
#     35              1                     105                               else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     38                                    133     Count coming in to IF
#     38              1                      31                               if (x)
#
#     40              1                     102                               else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     47                                    776     Count coming in to IF
#     47              1                      44                 if(rst) begin
#
#     50              1                     732                 else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     56                                    614     Count coming in to IF
#     56              1                      44                 if(rst) begin
#
#     59              1                     570                 else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     60                                    570     Count coming in to IF
#     60              1                     100                               if (cs == STORE)
#
#                                           470     All False Count
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     65                                    577     Count coming in to IF
#     65              1                     102      assign y = (cs == STORE)? 1:0;
#
#     65              2                     475      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%
#
```

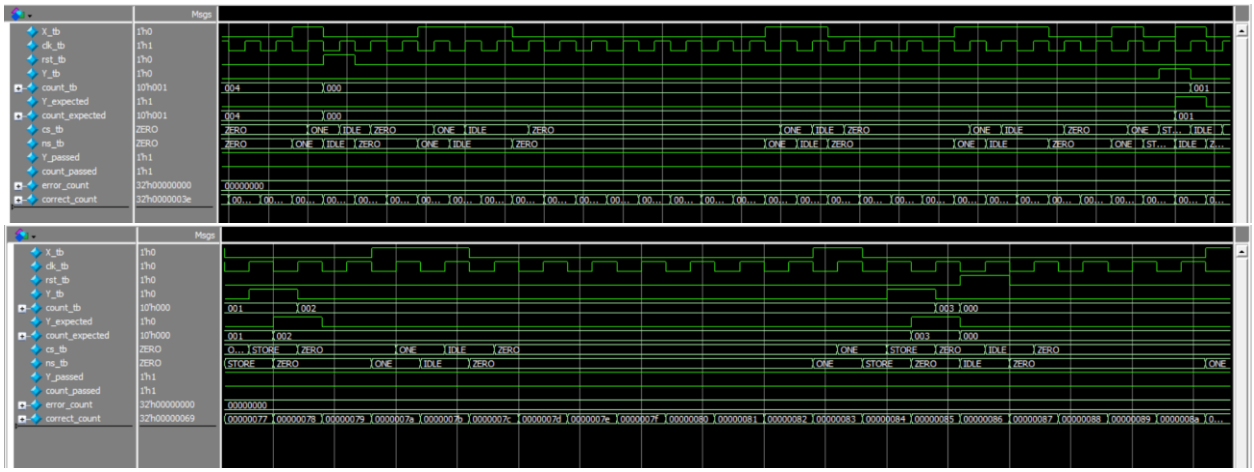
```
# ================================Condition Details================================
#
# Condition Coverage for instance /\FSM_tb#DUT  --
#
#   File FSm_010.v
# ----------------Focused Condition View-------------------
# Line       60 Item    1  (cs == 3)
# Condition totals: 1 of 1 input term covered = 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 instance /\FSM_tb#DUT  --
#
# 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                 186
#                      ZERO                 315
#                       ONE                 173
#                     STORE                 102
#     Covered Transitions :
#     ---------------------
# Line            Trans_ID           Hit_count          Transition
# ----            --------           ---------          ----------
#   26                   0                 116          IDLE -> ZERO
#   29                   1                 173          ZERO -> ONE
#   48                   2                  12          ZERO -> IDLE
#   36                   3                 102          ONE -> STORE
#   34                   4                  71          ONE -> IDLE
#   41                   5                  69          STORE -> ZERO
#   39                   6                  33          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%
#
#
# Toggle Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Toggles                         26        26         0   100.00%
#
# ================================Toggle Details================================
#
# Toggle Coverage for instance /\FSM_tb#DUT  --
#
#                                             Node      1H->0L      0L->1H                          "Coverage"
#                                             ----------------------------------------------------------------
#                                              clk           1           1                             100.00
#                                          cs[1-0]           1           1                             100.00
#                                          ns[1-0]           1           1                             100.00
#                                              rst           1           1                             100.00
#                                  users_count[4-0]          1           1                             100.00
#                                                x           1           1                             100.00
#                                                y           1           1                             100.00
#
# Total Node Count     =         13
# Toggled Node Count   =         13
# Untoggled Node Count =          0
#
# Toggle Coverage      =     100.00% (26 of 26 bins)
#
#
# Total Coverage By Instance (filtered view): 100%
#
```

- Questasim snippets






```
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# Test passes
# correct counter =        1002 , error counter =           0
# ** Note: $stop    : TESTBENCH.svh(42)
#    Time: 4008 ns  Iteration: 1  Instance: /FSM_tb
# Break in Module FSM_tb at TESTBENCH.svh line 42
```