Assignment 3 EXTRA

Digital Design Verification

Contents

1 Q1: Queue test

1.1 1. Testbench

```
module queue_tb;

initial begin

//=============================================================
// Declare int j and a queue q of type int
//=============================================================
int j;
int q [$];

//=============================================================
// initialize int j as 1 and queue q as (0, 2, 5)
//=============================================================
j = 1;
q = {0,2,5};

//=============================================================
// insert int j at index 1 in queue q and display q
//=============================================================
q.insert(1,j);
$display("after inserting %0d in index 1 new queue is : %0p",j,q);

//=============================================================
// delete index 1 element from queue q and display q
//=============================================================
q.delete(1);
$display("after deleting element in index 1 new queue is : %0p",q);

//=============================================================
// push an element (7) in the front in queue q and display q
//=============================================================
q.push_front(7);
$display("after pushing front element 7 new queue is : %0p",q);

//=============================================================
// push an element (9) at the back in queue q and display q
//=============================================================
q.push_back(9);
$display("after pushing back element 9 new queue is : %0p",q);

//=============================================================
// pop an element from back of queue q into j, display q, and j
//=============================================================
j = q.pop_back();
$display("after pop back into j, j = %0d and new queue is : %0p",j,q);

//=============================================================
// pop an element from front of queue q into j, display q, and j
//=============================================================
j = q.pop_front();
$display("after pop front into j, j = %0d and new queue is : %0p",j,q);

//=============================================================================================
// reverse, sort, reverse sort and shuffle the queue and display q after using each method
//=============================================================================================
// Reverse the queue and display q
q.reverse();
$display("After reverse: %p", q);

// Sort the queue and display q
q.sort();
$display("After sort: %p", q);

// Reverse sort the queue (sort then reverse) and display q
q.sort();
q.reverse();
$display("After reverse sort: %p", q);

// Shuffle the queue and display q
q.shuffle();
$display("After shuffle: %p", q);

$finish;

end
endmodule
```

```
VSIM 3> restart
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.queue_tb(fast)
VSIM 4> run -all
# after inserting 1 in index 1 new queue is : 0 1 2 5
# after deleting element in index 1 new queue is : 0 2 5
# after pushing front element 7 new queue is : 7 0 2 5
# after pushing back element 9 new queue is : 7 0 2 5 9
# after pop back into j, j = 9 and new queue is : 7 0 2 5
# after pop front into j, j = 7 and new queue is : 0 2 5
# After reverse: '{5, 2, 0}
# After sort: '{0, 2, 5}
# After reverse sort: '{5, 2, 0}
# After shuffle: '{2, 0, 5}
# ** Note: $finish    : C:/Users/Administrator/Desktop/assign 3 - verification diploma - EXTRA/Q1/queue_tb.sv(73)
#    Time: 0 ns  Iteration: 0  Instance: /queue_tb
# 1
# Break in Module queue_tb at C:/Users/Administrator/Desktop/assign 3 - verification diploma - EXTRA/Q1/queue_tb.sv line 73

VSIM 5>
```

Figure 1: Transcript

2 Q2: Adder

2.1 1. Testbench code

```

import adder_package::*;

module adder_tb;
  // inputs to instantiated Adder
  reg clk, reset;
  reg signed [3:0] A; // Input data A: 2's complement
  reg signed [3:0] B; // Input data B: 2's complement

  // Output of Adder
  wire signed [4:0] C; // Adder output: 2's complement

  integer error_count; // 32-bit signed
  integer correct_count; // 32-bit signed

  adder a1 (
    .clk(clk),
    .reset(reset),
    .A(A),     // Input data A: signed
    .B(B),     // Input data B: signed
    .C(C)      // Adder output: signed
  );

  // Create the clock and reset
    initial begin
        clk = 0;
        forever
            #5 clk = ~clk;
    end

  adder_rand_class stim;

    initial begin
        error_count = 0;
        correct_count = 0;

        stim = new;

        repeat (50) begin

            assert(stim.randomize()) else $error("Randimization Failed");

            @(negedge clk);
            A = stim.A;
            B = stim.B;
            reset = stim.reset;
            stim.C = C;

            if(stim.reset == 1'b1) begin
            end else begin
               stim.sample_A();
               stim.sample_B();
            end

            @(posedge clk);
            check_Task;

        end

        $display("%0t: At end of test error count is %0d and correct count = %0d", $time, error_count, correct_count);
        $finish;
    end

    task check_Task();
        logic signed [4:0] expected_result;

        #1; //due to simulator doesnnot given me output in posedge immediately

        if (reset && C == 0) begin
          correct_count = correct_count + 1;
        end else if (reset && C != 0) begin
          error_count = error_count + 1;
          $display("Error! For A=%0d and B=%0d C should equal %0d but is %0d", $time, A, B, expected_result, C);
          $stop;
        end

        if(!reset) begin
            expected_result = stim.A+stim.B;
            if (C == expected_result)begin
                correct_count = correct_count + 1;
            end else begin
                error_count = error_count + 1;
                $display("Error! For A=%0d and B=%0d C should equal %0d but is %0d", $time, A, B, expected_result, C);
                $stop;
            end
        end

    endtask

endmodule
```

2.2 2. Package code

```
package adder_package;
    typedef enum logic signed [31:0]{
```

```
        MAXPOS = 7,
        ZERO = 0,
        MAXNEG = -8
    } enum_t;

    class adder_rand_class;
        rand bit reset;
        rand bit signed [3:0] A;
        rand bit signed [3:0] B;
        bit signed [4:0] C;

        constraint rst_c {
            reset dist {1:/10, 0:/90};
        }

        constraint A_c {
            A dist {MAXPOS:/20, ZERO:/20, MAXNEG:/20, [-7:-1]:/10, [1:6]:/10};
        }

        constraint B_c {
            B dist {MAXPOS:/20, ZERO:/20, MAXNEG:/20, [-7:-1]:/10, [1:6]:/10};
        }

        covergroup Covgrp_A;
            cp1: coverpoint A {
                bins data_0 = {0};
                bins data_max = {MAXPOS};
                bins data_min = {MAXNEG};
                bins data_default = default;
            }

            cp2: coverpoint A {
                bins data_0max = (0 => MAXPOS);
                bins data_maxmin = (MAXPOS => MAXNEG);
                bins data_minmax = (MAXNEG => MAXPOS);
            }
        endgroup

        covergroup Covgrp_B;
            cp3: coverpoint B {
                bins data_0 = {0};
                bins data_max = {MAXPOS};
                bins data_min = {MAXNEG};
                bins data_default = default;
            }

            cp4: coverpoint B {
                bins data_0max = (0 => MAXPOS);
                bins data_maxmin = (MAXPOS => MAXNEG);
                bins data_minmax = (MAXNEG => MAXPOS);
            }
        endgroup

        function new();
            Covgrp_A = new();
            Covgrp_B = new();
        endfunction

        function void sample_A();
            Covgrp_A.sample();
        endfunction

        function void sample_B();
            Covgrp_B.sample();
        endfunction

    endclass
endpackage
```

2.3 3. Design code

```
module adder (
    input  clk,
    input  reset,
    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 // Adder output in 2's complement
);

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

endmodule
```

2.4 4. Bug Fixes

no bugs

2.5 5. Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
ADDER_1	When the reset is asserted, the output C value must be low	Directed at the start of the simulation with reset constraint: reset dist {1:/10, 0:/90}	Reset coverage through reset constraint	A checker in the testbench verifies C=0 when reset is high
ADDER_2	Verifying maximum negative value on A and maximum negative value on B	Constrained random with A_c and B_c constraints targeting MAXNEG (-8) values	Covered by Covgrp_A.cp1 and Covgrp_B.cp3 bins for data_min	A checker in the testbench to make sure the output is correct (-8 + -8 = -16)
ADDER_3	Verifying maximum negative value on A and zero value on B	Constrained random with distributions favoring MAXNEG (-8) and ZERO values	Covered by Covgrp_A.cp1 (data_min) and Covgrp_B.cp1 (data_0)	A checker in the testbench to make sure the output is correct (-8 + 0 = -8)
ADDER_4	Verifying maximum negative value on A and maximum positive value on B	Constrained random with distributions favoring MAXNEG (-8) and MAXPOS (7) values	Covered by Covgrp_A.cp1 (data_min) and Covgrp_B.cp1 (data_max)	A checker in the testbench to make sure the output is correct (-8 + 7 = -1)
ADDER_5	Verifying zero on A and maximum negative value on B	Constrained random with distributions favoring ZERO and MAXNEG (-8) values	Covered by Covgrp_A.cp1 (data_0) and Covgrp_B.cp1 (data_min)	A checker in the testbench to make sure the output is correct (0 + -8 = -8)
ADDER_6	Verifying zero on A and zero on B	Constrained random with distributions favoring ZERO values	Covered by Covgrp_A.cp1 (data_0) and Covgrp_B.cp1 (data_0)	A checker in the testbench to make sure the output is correct (0 + 0 = 0)
ADDER_7	Verifying zero on A and maximum positive value on B	Constrained random with distributions favoring ZERO and MAXPOS (7) values	Covered by Covgrp_A.cp1 (data_0) and Covgrp_B.cp1 (data_max)	A checker in the testbench to make sure the output is correct (0 + 7 = 7)
ADDER_8	Verifying maximum positive value on A and maximum negative value on B	Constrained random with distributions favoring MAXPOS (7) and MAXNEG (-8) values	Covered by Covgrp_A.cp1 (data_max) and Covgrp_B.cp1 (data_min)	A checker in the testbench to make sure the output is correct (7 + -8 = -1)
ADDER_9	Verifying maximum positive value on A and zero on B	Constrained random with distributions favoring MAXPOS (7) and ZERO values	Covered by Covgrp_A.cp1 (data_max) and Covgrp_B.cp1 (data_0)	A checker in the testbench to make sure the output is correct (7 + 0 = 7)
ADDER_10	Verifying maximum positive value on A and maximum positive value on B	Constrained random with distributions favoring MAXPOS (7) values	Covered by Covgrp_A.cp1 (data_max) and Covgrp_B.cp1 (data_max)	A checker in the testbench to make sure the output is correct (7 + 7 = 14)

Table 1: Verification Plan for the Adder Module

2.6 6. Do File

```
vlib work
vlog adder.sv adder_tb.sv adder_package.sv +cover -covercells
vsim -voptargs=+acc work.adder_tb -cover
add wave *
coverage save adder_tb.ucdb -onexit
run -all

# to run do file
#-- do run.txt
# to execute coverage report
#-- vcover report adder_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=adder
#-- vcover report -details -cvg -output adder_coverage_report.txt adder_tb.ucdb
```

2.7 7. Code Coverage Report

```
Coverage Report by DU with details

=================================================================================
=== Design Unit: work.adder
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         2         2         0   100.00%

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

Branch Coverage for Design Unit work.adder

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File adder.sv
----------------------------------IF Branch------------------------------------
    11                                       109     Count coming in to IF
    11              1                         18         if (reset)

    13              1                         91         else

Branch totals: 2 hits of 2 branches = 100.00%


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       3         3         0   100.00%

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

Statement Coverage for Design Unit work.adder ---

    Line         Item                      Count     Source
```

File adder.sv

```
1                                       module adder (
2                                           input  clk,
3                                           input  reset,
4                                           input  signed [3:0] A, // Input data A in 2's complement
5                                           input  signed [3:0] B, // Input data B in 2's complement
6                                           output reg signed [4:0] C // Adder output in 2's complement
7                                       );
8
9                                       // Register output C
10              1                109    always @(posedge clk or posedge reset) begin
11                                          if (reset)
12              1                 18            C <= 5'b0;
13                                          else
14              1                 91            C <= A + B;
```

Toggle Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
Toggles	30	30	0	100.00%

Toggle Details

Toggle Coverage for Design Unit work.adder

Node	1H–>0L	0L–>1H	"Coverage"
A[0–3]	1	1	100.00
B[0–3]	1	1	100.00
C[0–4]	1	1	100.00
clk	1	1	100.00
reset	1	1	100.00

```
Total Node Count        =        15
Toggled Node Count      =        15
Untoggled Node Count    =         0

Toggle Coverage         =    100.00% (30 of 30 bins)


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

2.8 8. Code Coverage Report

```
Coverage Report by instance with details

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

Covergroup Coverage:
    Covergroups                 2        na        na    100.00%
        Coverpoints/Crosses     4        na        na         na
            Covergroup Bins    12        12         0    100.00%
```

Covergroup	Metric	Goal	Bins	Status
TYPE /adder_package/adder_rand_class/Covgrp_A	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Covergroup instance \/adder_package::adder_rand_class::Covgrp_A	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered

covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0	23	1	–	Covered
bin data_max	20	1	–	Covered
bin data_min	17	1	–	Covered
default bin data_default	30		–	Occurred
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0max	4	1	–	Covered
bin data_maxmin	3	1	–	Covered
bin data_minmax	3	1	–	Covered
TYPE /adder_package/adder_rand_class/Covgrp_B	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Covergroup instance \/adder_package::adder_rand_class::Covgrp_B				
	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0	22	1	–	Covered
bin data_max	23	1	–	Covered
bin data_min	18	1	–	Covered
default bin data_default	27		–	Occurred
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0max	5	1	–	Covered
bin data_maxmin	4	1	–	Covered
bin data_minmax	8	1	–	Covered

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /adder_package/adder_rand_class/Covgrp_A	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Covergroup instance \/adder_package::adder_rand_class::Covgrp_A				
	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0	23	1	–	Covered
bin data_max	20	1	–	Covered
bin data_min	17	1	–	Covered
default bin data_default	30		–	Occurred
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0max	4	1	–	Covered
bin data_maxmin	3	1	–	Covered
bin data_minmax	3	1	–	Covered
TYPE /adder_package/adder_rand_class/Covgrp_B	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	

Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
Covergroup instance \/adder_package::adder_rand_class::Covgrp_B				
	100.00%	100	–	Covered
covered/total bins:	6	6	–	
missing/total bins:	0	6	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0	22	1	–	Covered
bin data_max	23	1	–	Covered
bin data_min	18	1	–	Covered
default bin data_default	27		–	Occurred
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	3	3	–	
missing/total bins:	0	3	–	
% Hit:	100.00%	100	–	
bin data_0max	5	1	–	Covered
bin data_maxmin	4	1	–	Covered
bin data_minmax	8	1	–	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 2

Total Coverage By Instance (filtered view): 100.00%

2.9 9.Waveform

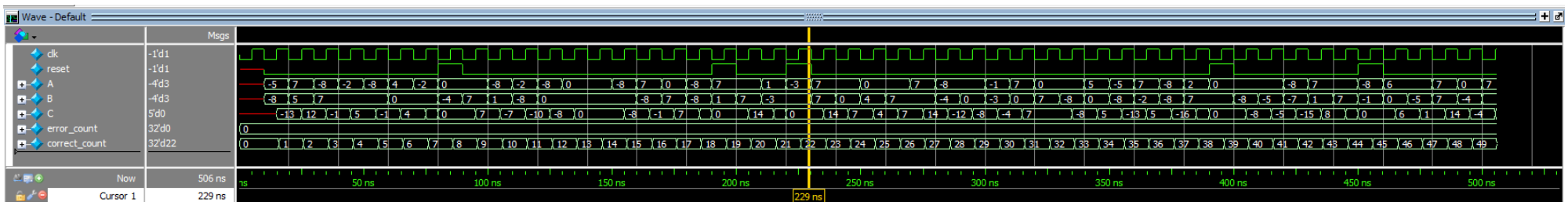

Figure 2: simulation waveform

```
# Saving coverage database on exit...
# End time: 16:14:00 on Apr 13,2025, Elapsed time: 0:02:45
# Errors: 3, Warnings: 1
# vsim -voptargs="+acc" work.adder_tb -coverage
# Start time: 16:14:00 on Apr 13,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.adder_package(fast)
# Loading work.adder_tb_sv_unit(fast)
# Loading work.adder_tb(fast)
# Loading work.adder(fast)
# 506: At end of test error count is 0 and correct count = 50
# ** Note: $finish    : adder_tb.sv(61)
#    Time: 506 ns  Iteration: 0  Instance: /adder_tb
# 1
# Break in Module adder_tb at adder_tb.sv line 61
```

Figure 3: Transcript : all test cases passed

3 Q3: FSM_010

3.1 1. Testbench code

```
1  `timescale 1ns/1ps
2
3  import fsm_package::*;
4
5  module fsm_010_tb;
6      //============================
7      // declare DUT signals
8      //============================
9      logic clk, rst, x;
10
11     logic y;
12     logic [1:0] users_count;
13
14     //==============================
15     // declare golden model signals
16     //==============================
17     logic golden_y;
18     logic [1:0] golden_users_count;
19
20     //============================
21     // instantiate FSM "DUT"
22     //============================
23     FSM_010 DUT (.*);
24
25     //============================
26     // instantiate golden model
27     //============================
28     golden_model golden_DUT (.*);
29
30     //======================
31     // 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;
    fsm.y_exp = 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;
            cp5: coverpoint x { bins zero_one_zero = (0=>1=>0);}
        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)
```

```
100         golden_users_count <= golden_users_count + 1;
101     end
102 end
103 
104 
105 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 8. Code 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%
```

3.9 9. Functional Coverage Report

```
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]                                      83          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]                                     101          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]                                     103          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]                                      78          1          -    Covered
        bin auto[1]                                      28          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]                                      83          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]	101	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]	103	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]	78	1	–	Covered
bin auto[1]	28	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%

3.10 10.Waveform

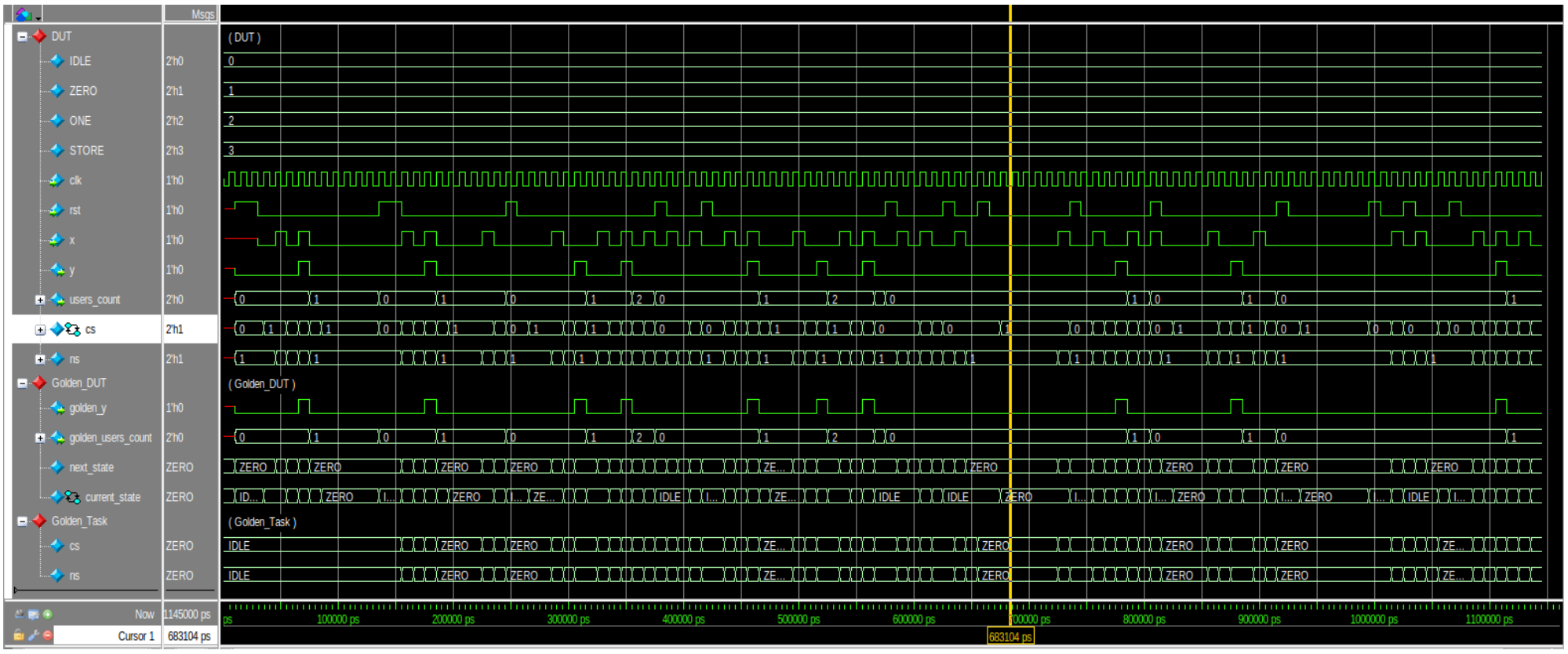

Figure 4: simulation waveform

```
#  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=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=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