Data Types & Constrained Random

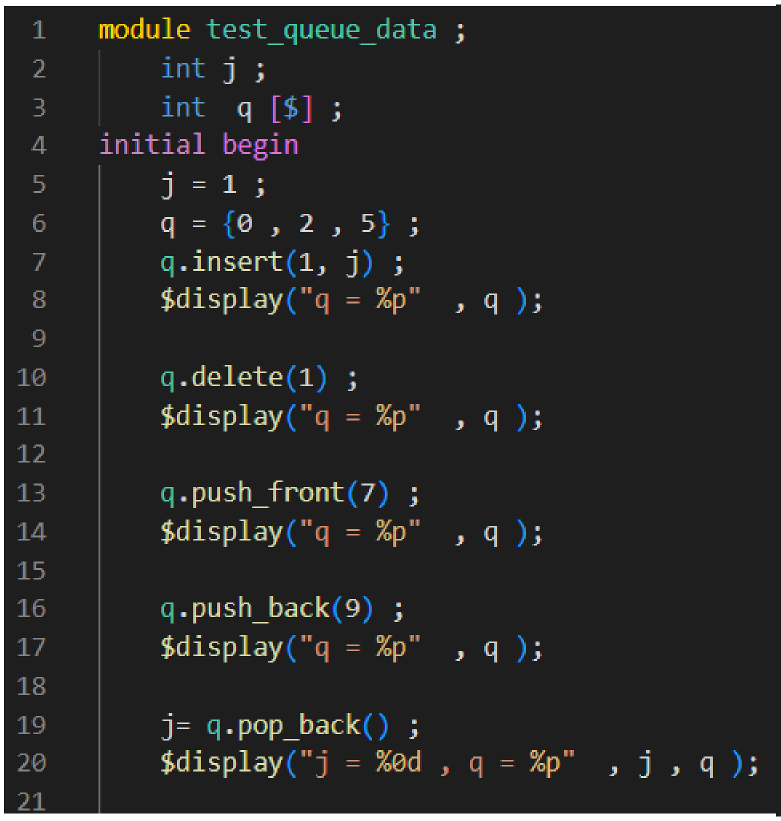
Assignment3_extra

By: Mohamed Sayed Mohamed Soliman

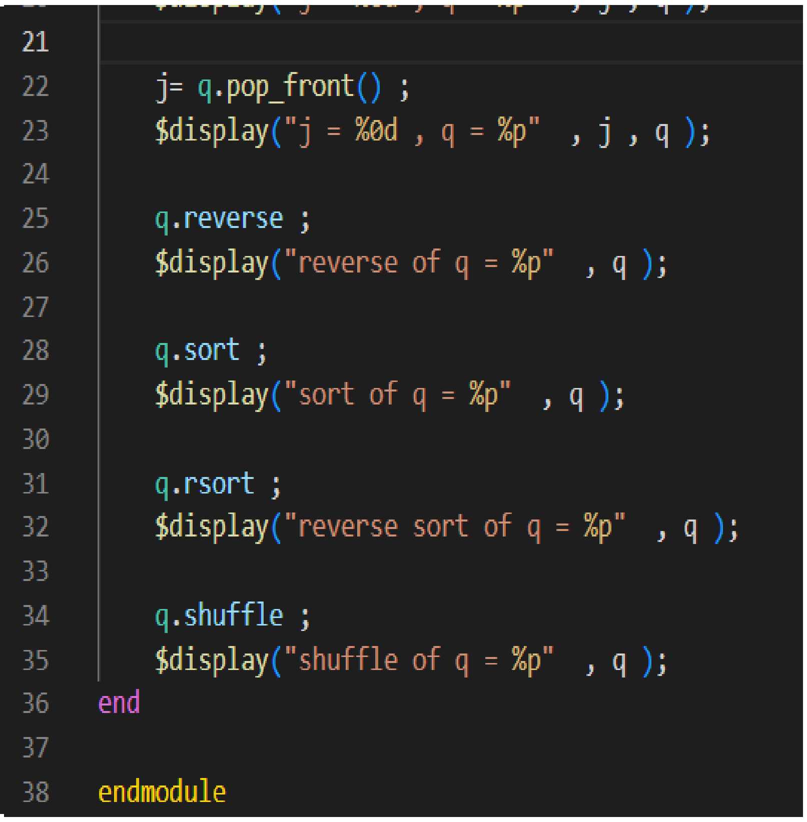
Question 1

- Write a module to test queue data type and its predefined methods. Run Questasim to make sure the display statements are working as expected.
- Declare int j and a queue q of type int
- initialize int j as 1 and queue q as (0, 2, 5)
- insert int j at index 1 in queue q and display q
- delete index 1 element from queue q and display q
- push an element (7) in the front in queue q and display q
- push an element (9) at the back in queue q and display q
- pop an element from back of queue q into j, display q, and j
- pop an element from front of queue q into j, display q, and j
- reverse, sort, reverse sort and shuffle the queue and display q after using each method

1. Code of the design

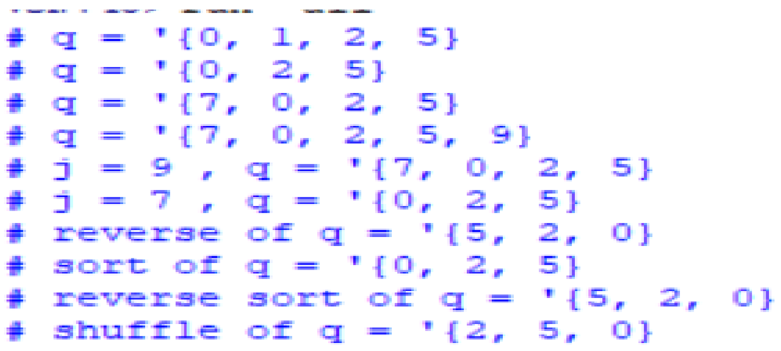


```
module test_queue_data ;
    int j ;
    int  q [$] ;
initial begin
    j = 1 ;
    q = {0 , 2 , 5} ;
    q.insert(1, j) ;
    $display("q = %p"  , q );

    q.delete(1) ;
    $display("q = %p"  , q );

    q.push_front(7) ;
    $display("q = %p"  , q );

    q.push_back(9) ;
    $display("q = %p"  , q );

    j= q.pop_back() ;
    $display("j = %0d , q = %p"  , j , q );

```



```

    j= q.pop_front() ;
    $display("j = %0d , q = %p"  , j , q );

    q.reverse ;
    $display("reverse of q = %p"  , q );

    q.sort ;
    $display("sort of q = %p"  , q );

    q.rsort ;
    $display("reverse sort of q = %p"  , q );

    q.shuffle ;
    $display("shuffle of q = %p"  , q );
end

endmodule
```

2. Result of the simulation



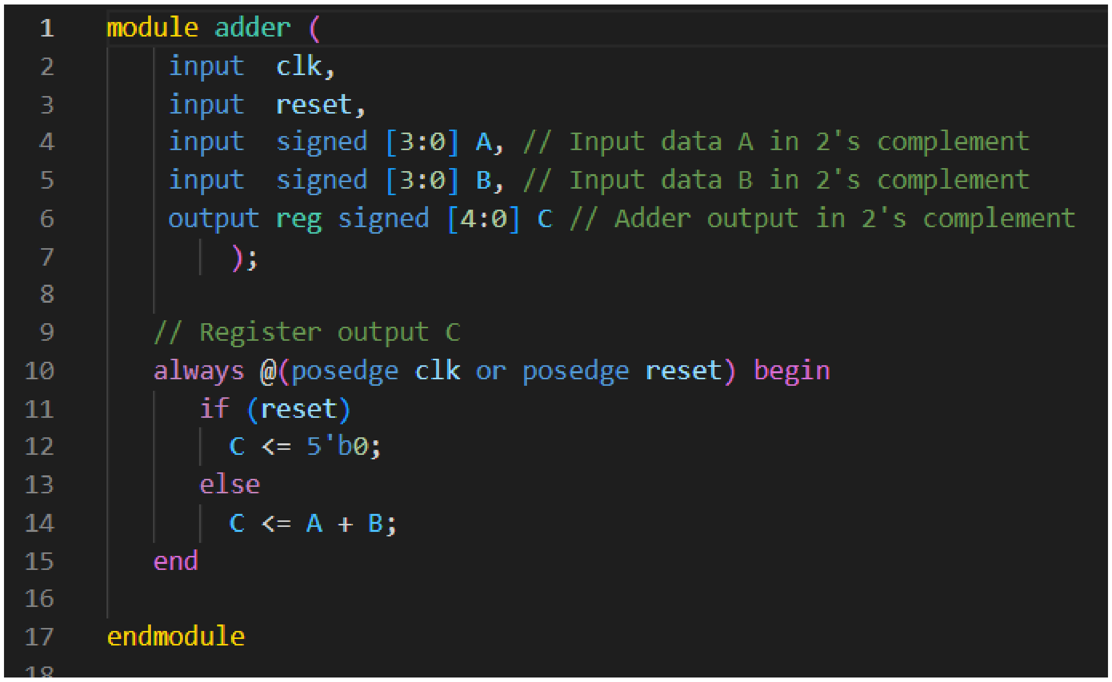
```
# q = '{0, 1, 2, 5}
# q = '{0, 2, 5}
# q = '{7, 0, 2, 5}
# q = '{7, 0, 2, 5, 9}
# j = 9 , q = '{7, 0, 2, 5}
# j = 7 , q = '{0, 2, 5}
# reverse of q = '{5, 2, 0}
# sort of q = '{0, 2, 5}
# reverse sort of q = '{5, 2, 0}
# shuffle of q = '{2, 5, 0}
```

Question 2

➢ adder

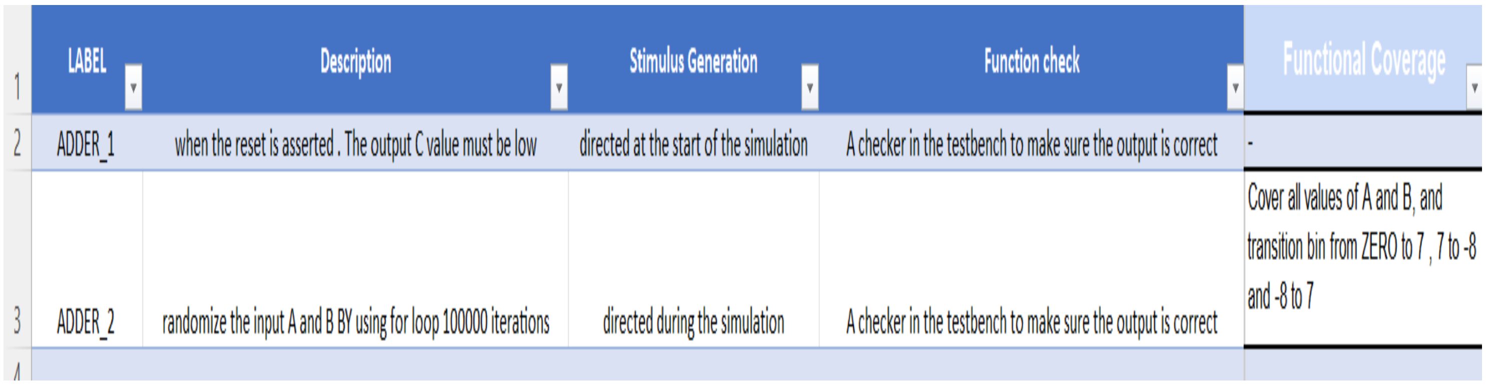
1. Create a package that have a user defined enum that takes the value MAXPOS, ZERO, and MAXNEG.
2. Create a class to randomize the design inputs under the following constraints
    a. Reset to be asserted with a low probability that you decide
    b. Constrain the adder inputs to (A, B) have the MAXPOS, ZERO and MAXNEG values more often than the remaining values
3. Functional coverage model in the class
    a. identical Covergroups for ports A and B (Covgrp_A and Convgrp_B)
    b. Each covergroup has 2 coverpoints
    c. First coverpoint will cover the following bins

1. Code Design



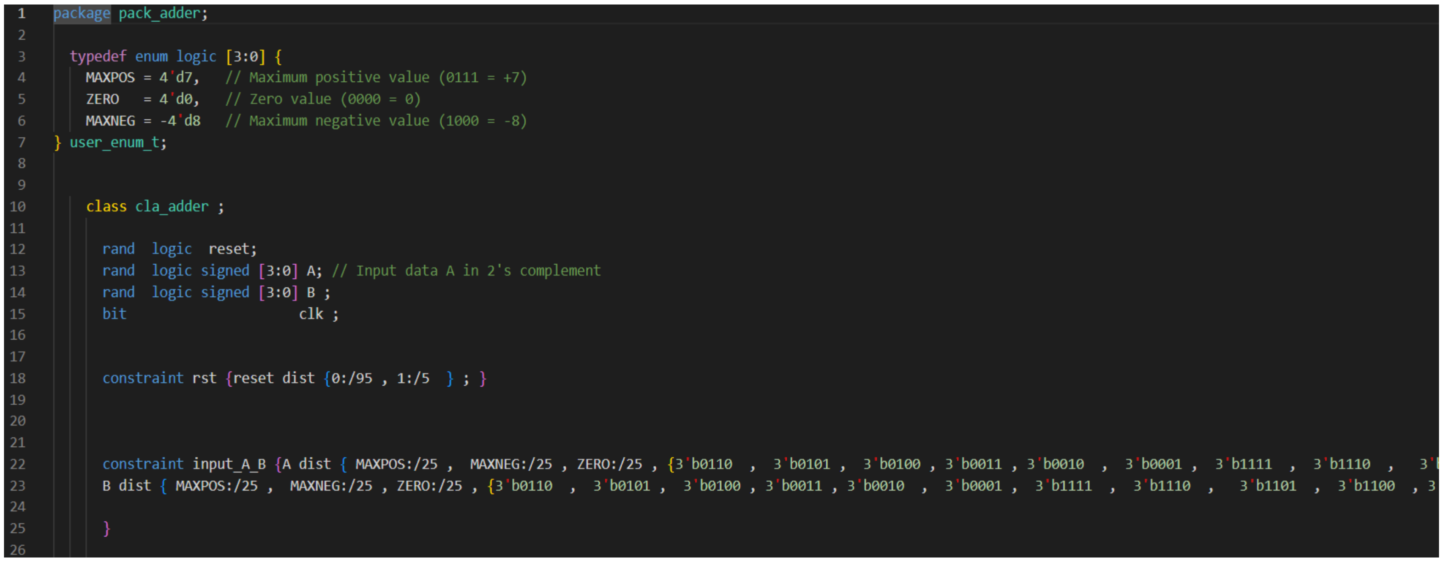
```verilog
module adder (
    input  clk,
    input  reset,
    input  signed [3:0] A, // Input data A in 2's complement
    input  signed [3:0] B, // Input data B in 2's complement
    output reg signed [4:0] C // Adder output in 2's complement
      );

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

endmodule
```

2. Verification plan

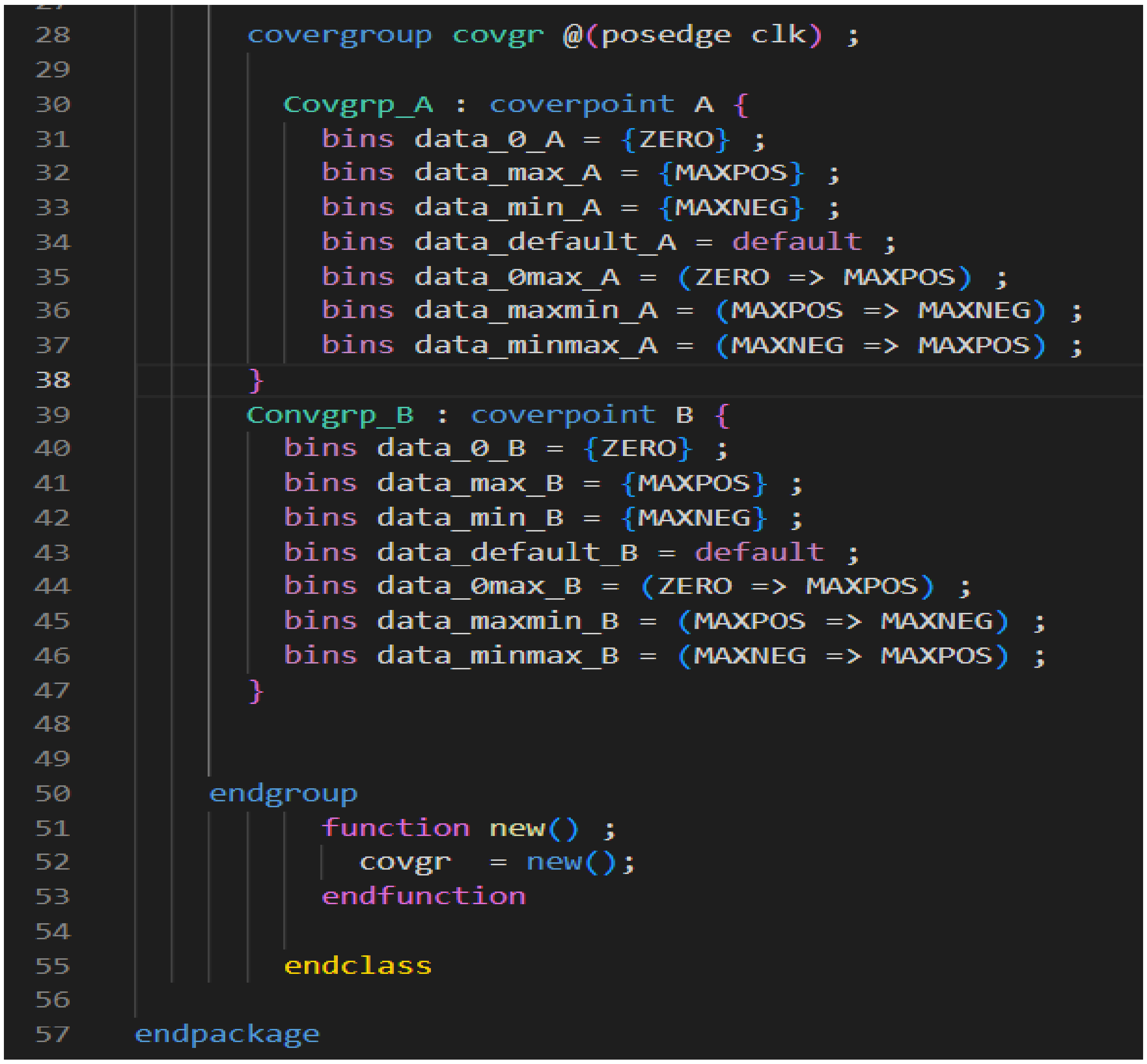


LABEL	Description	Stimulus Generation	Function check	Functional Coverage
ADDER_1	when the reset is asserted . The output C value must be low	directed at the start of the simulation	A checker in the testbench to make sure the output is correct	-
ADDER_2	randomize the input A and B BY using for loop 100000 iterations	directed during the simulation	A checker in the testbench to make sure the output is correct	Cover all values of A and B, and transition bin from ZERO to 7 , 7 to -8 and -8 to 7

3. alu Package

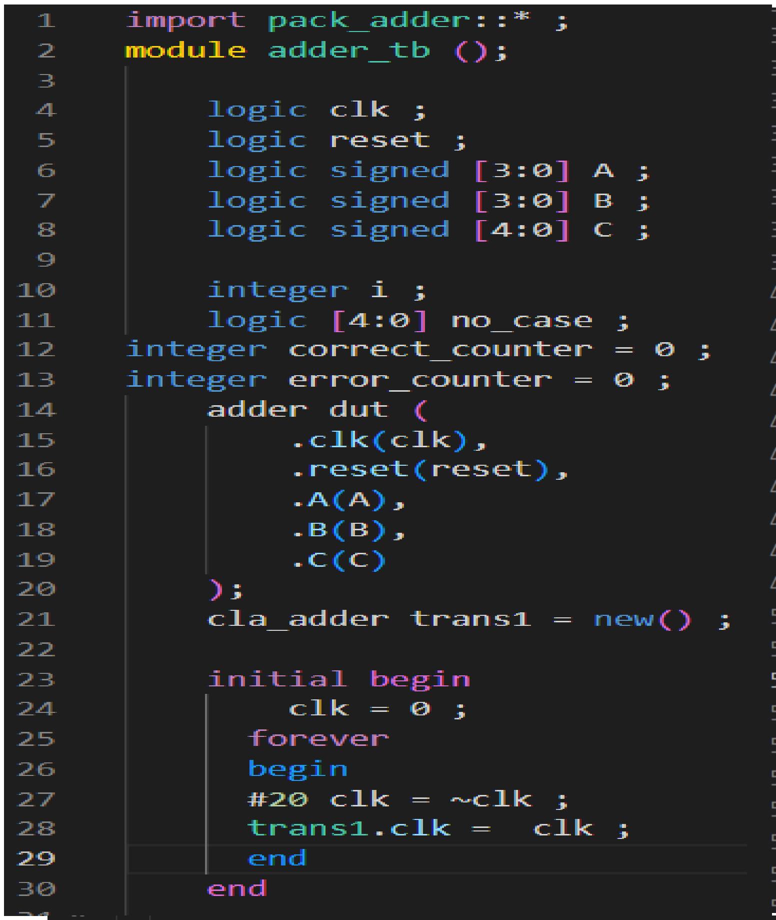


```
package pack_adder;

  typedef enum logic [3:0] {
    MAXPOS = 4'd7,    // Maximum positive value (0111 = +7)
    ZERO   = 4'd0,    // Zero value (0000 = 0)
    MAXNEG = -4'd8    // Maximum negative value (1000 = -8)
} user_enum_t;


    class cla_adder ;

      rand  logic  reset;
      rand  logic signed [3:0] A; // Input data A in 2's complement
      rand  logic signed [3:0] B ;
      bit                    clk ;


      constraint rst {reset dist {0:/95 , 1:/5  } ; }



      constraint input_A_B {A dist { MAXPOS:/25 ,  MAXNEG:/25 , ZERO:/25 , {3'b0110  ,  3'b0101 ,  3'b0100 , 3'b0011 , 3'b0010  ,  3'b0001 ,  3'b1111  ,  3'b1110  ,   3'
      B dist { MAXPOS:/25 ,  MAXNEG:/25 , ZERO:/25 , {3'b0110  ,  3'b0101 ,  3'b0100 , 3'b0011 , 3'b0010  ,  3'b0001 ,  3'b1111  ,  3'b1110  ,   3'b1101  ,  3'b1100  , 3

      }

```

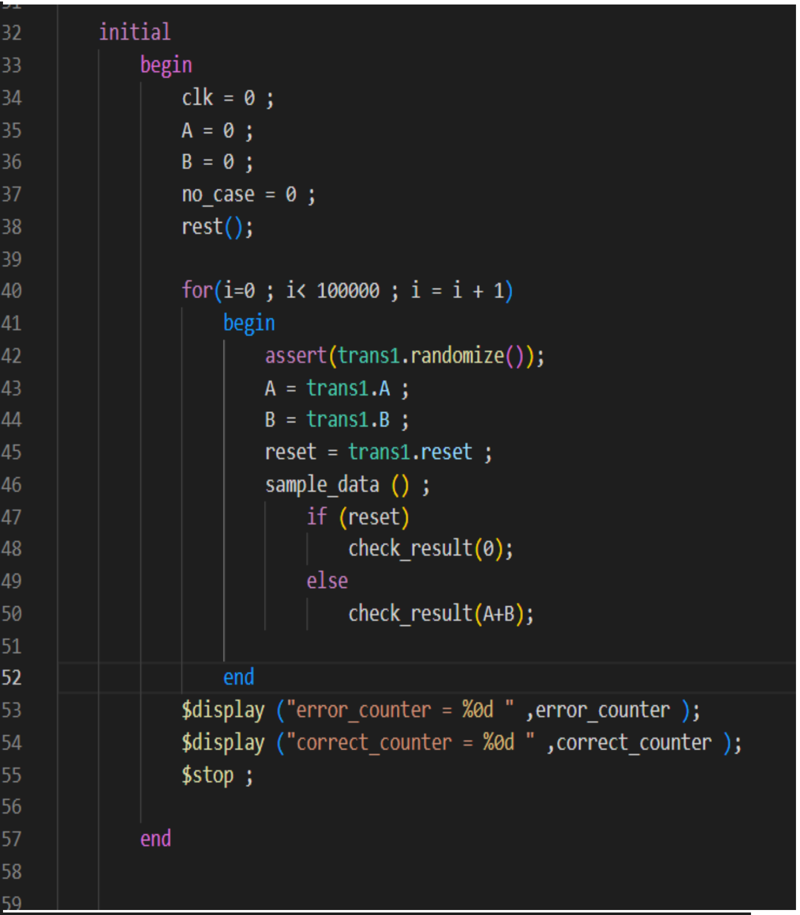


```
        covergroup covgr @(posedge clk) ;

          Covgrp_A : coverpoint A {
            bins data_0_A = {ZERO} ;
            bins data_max_A = {MAXPOS} ;
            bins data_min_A = {MAXNEG} ;
            bins data_default_A = default ;
            bins data_0max_A = (ZERO => MAXPOS) ;
            bins data_maxmin_A = (MAXPOS => MAXNEG) ;
            bins data_minmax_A = (MAXNEG => MAXPOS) ;
        }
        Convgrp_B : coverpoint B {
          bins data_0_B = {ZERO} ;
          bins data_max_B = {MAXPOS} ;
          bins data_min_B = {MAXNEG} ;
          bins data_default_B = default ;
          bins data_0max_B = (ZERO => MAXPOS) ;
          bins data_maxmin_B = (MAXPOS => MAXNEG) ;
          bins data_minmax_B = (MAXNEG => MAXPOS) ;
        }


      endgroup
            function new() ;
              covgr  = new();
            endfunction

          endclass

  endpackage
```

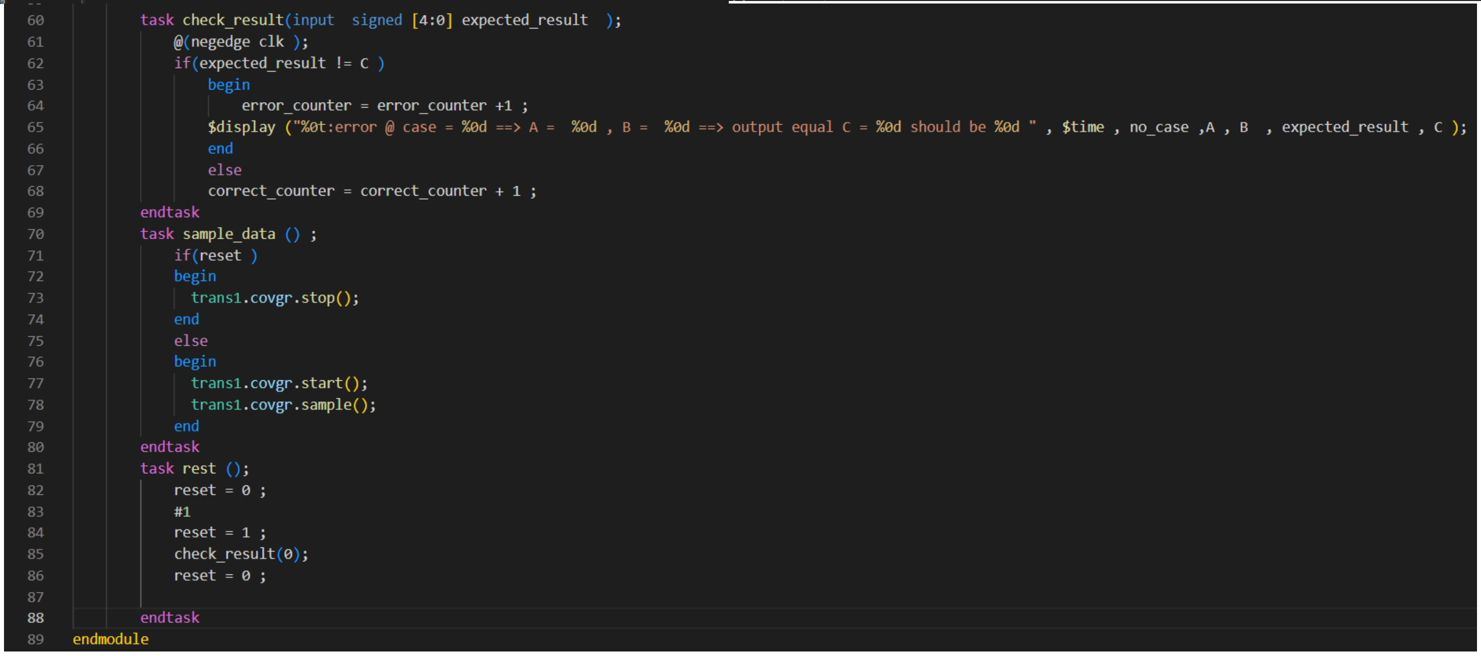
4. alu Testbench



```
import pack_adder::* ;
module adder_tb ();

    logic clk ;
    logic reset ;
    logic signed [3:0] A ;
    logic signed [3:0] B ;
    logic signed [4:0] C ;

    integer i ;
    logic [4:0] no_case ;
integer correct_counter = 0 ;
integer error_counter = 0 ;
    adder dut (
        .clk(clk),
        .reset(reset),
        .A(A),
        .B(B),
        .C(C)
    );
    cla_adder trans1 = new() ;

    initial begin
        clk = 0 ;
      forever
      begin
      #20 clk = ~clk ;
      trans1.clk =  clk ;
      end
    end
```

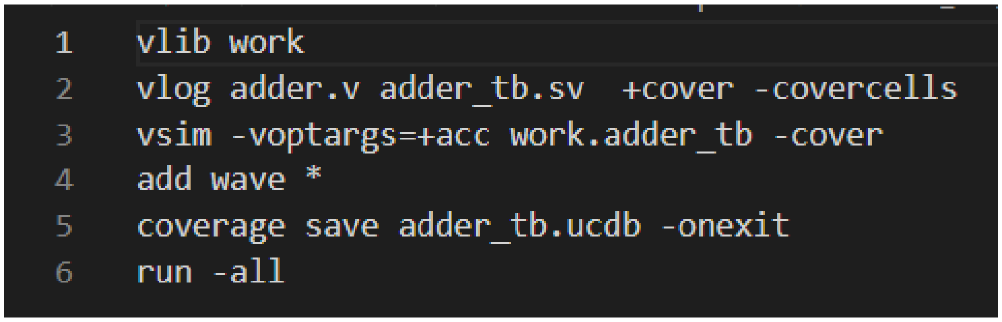


```
    initial
        begin
            clk = 0 ;
            A = 0 ;
            B = 0 ;
            no_case = 0 ;
            rest();

            for(i=0 ; i< 100000 ; i = i + 1)
                begin
                    assert(trans1.randomize());
                    A = trans1.A ;
                    B = trans1.B ;
                    reset = trans1.reset ;
                    sample_data () ;
                        if (reset)
                            check_result(0);
                        else
                            check_result(A+B);

                end
            $display ("error_counter = %0d " ,error_counter );
            $display ("correct_counter = %0d " ,correct_counter );
            $stop ;

        end
```



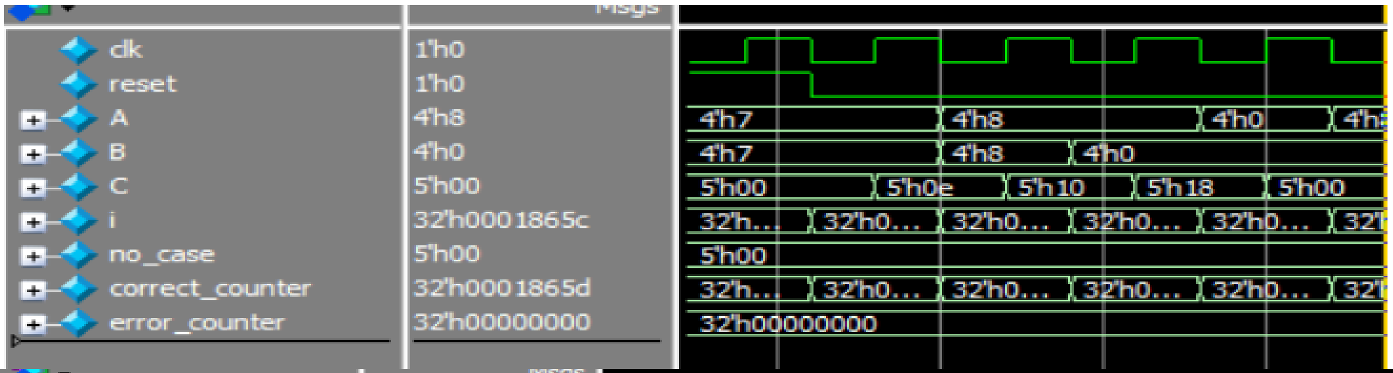
```
        task check_result(input  signed [4:0] expected_result  );
            @(negedge clk );
            if(expected_result != C )
                begin
                    error_counter = error_counter +1 ;
                $display ("%0t:error @ case = %0d ==> A =  %0d , B =  %0d ==> output equal C = %0d should be %0d " , $time , no_case ,A , B  , expected_result , C );
                end
                else
                correct_counter = correct_counter + 1 ;
        endtask
        task sample_data () ;
            if(reset )
            begin
              trans1.covgr.stop();
            end
            else
            begin
              trans1.covgr.start();
              trans1.covgr.sample();
            end
        endtask
        task rest ();
            reset = 0 ;
            #1
            reset = 1 ;
            check_result(0);
            reset = 0 ;

        endtask
endmodule
```

5. Do File

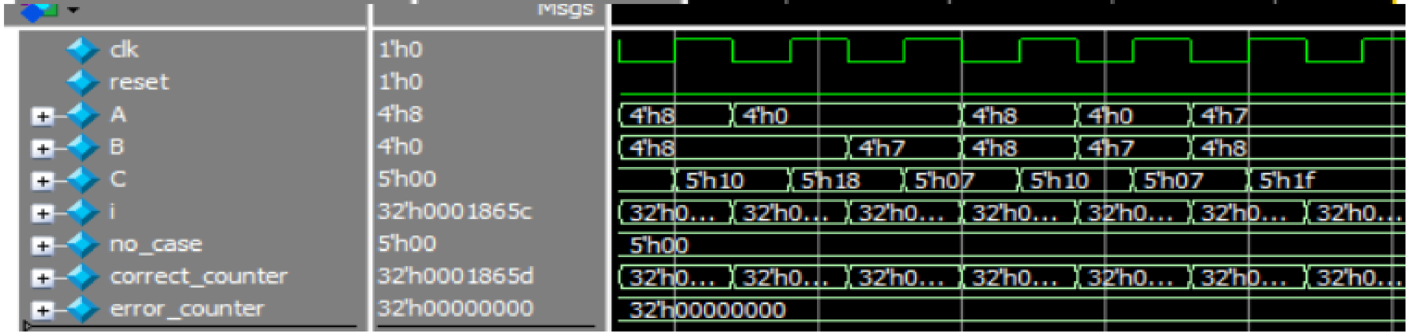


```
vlib work
vlog adder.v adder_tb.sv  +cover -covercells
vsim -voptargs=+acc work.adder_tb -cover
add wave *
coverage save adder_tb.ucdb -onexit
run -all
```

6. Waveform

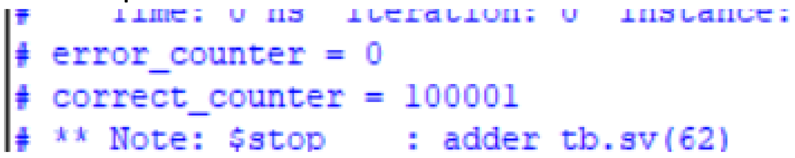


```
clk              1'h0
reset            1'h0
A                4'h8
B                4'h0
C                5'h00
i                32'h0001865c
no_case          5'h00
correct_counter  32'h0001865d
error_counter    32'h00000000
```



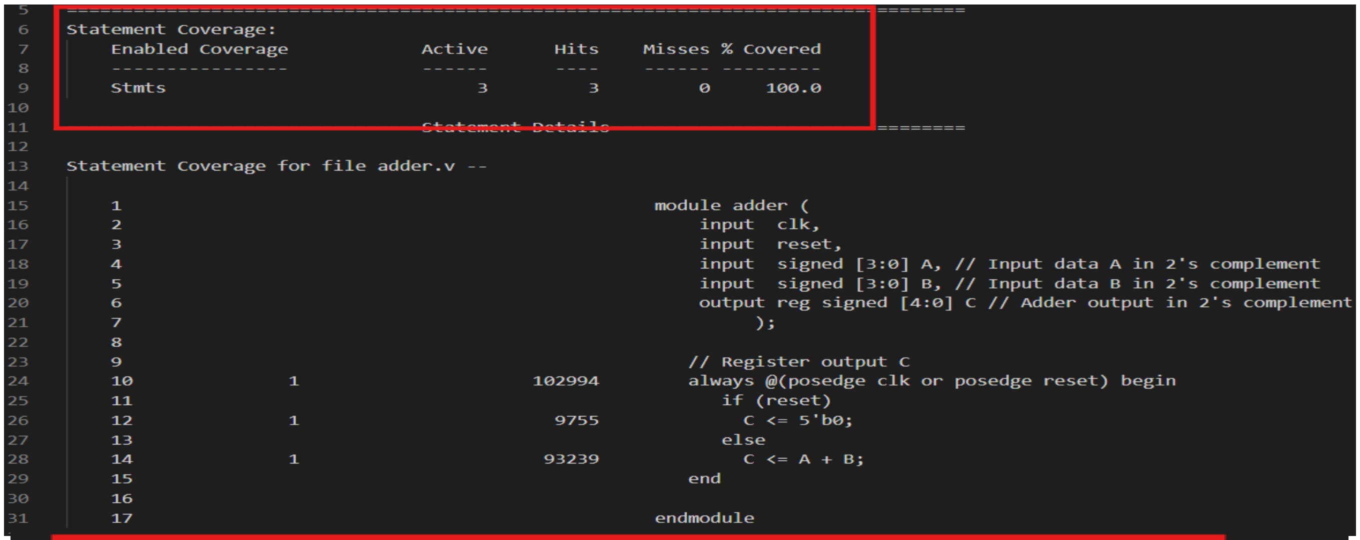
```
clk              1'h0
reset            1'h0
A                4'h8
B                4'h0
C                5'h00
i                32'h0001865c
no_case          5'h00
correct_counter  32'h0001865d
error_counter    32'h00000000
```

7. Transcript

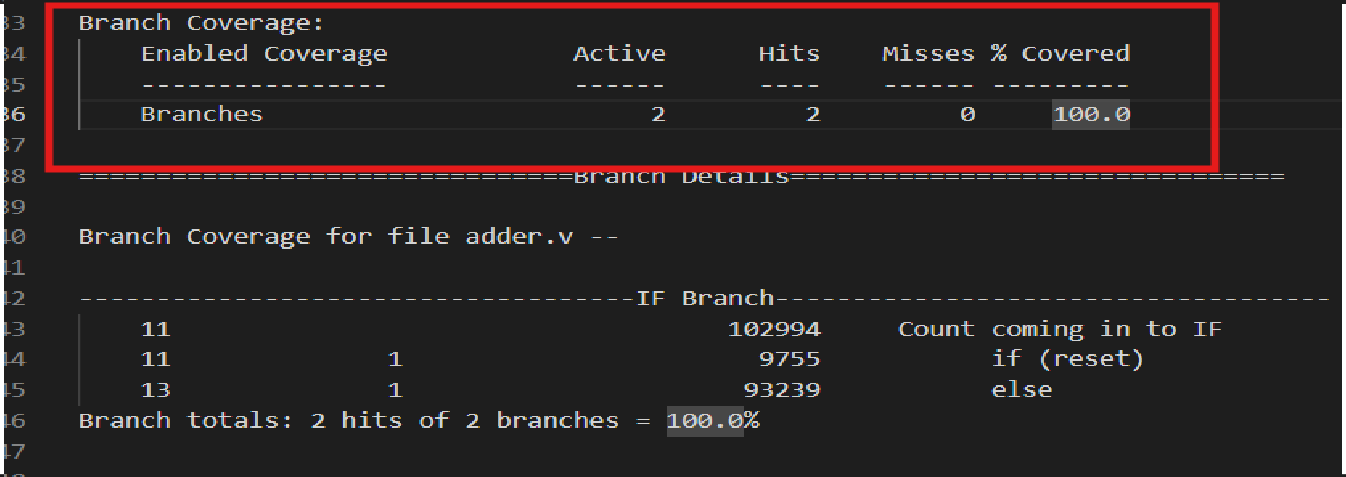


```
#     Time: 0 ns  Iteration: 0  Instance:
# error_counter = 0
# correct_counter = 100001
# ** Note: $stop    : adder_tb.sv(62)
```

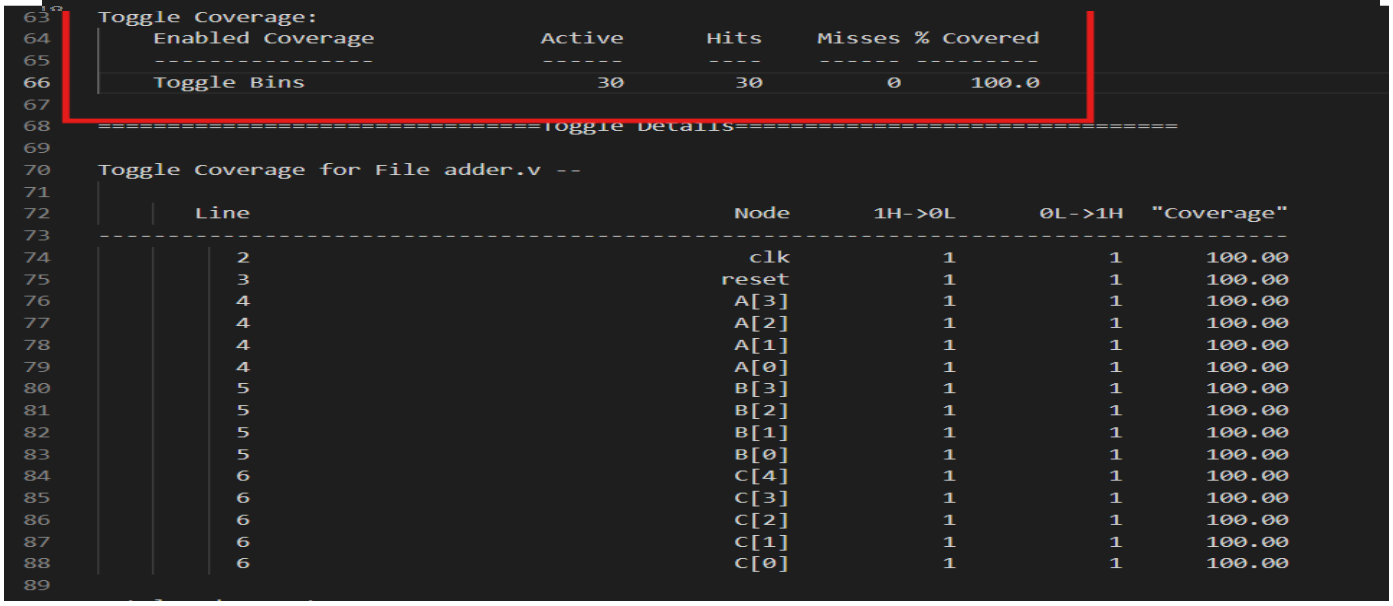
8. Code Coverage



```
=================================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                              3         3         0     100.0

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

Statement Coverage for file adder.v --

    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                    102994          always @(posedge clk or posedge reset) begin
    11                                                     if (reset)
    12             1                      9755               C <= 5'b0;
    13                                                     else
    14             1                     93239               C <= A + B;
    15                                                  end
    16
    17                                               endmodule
```

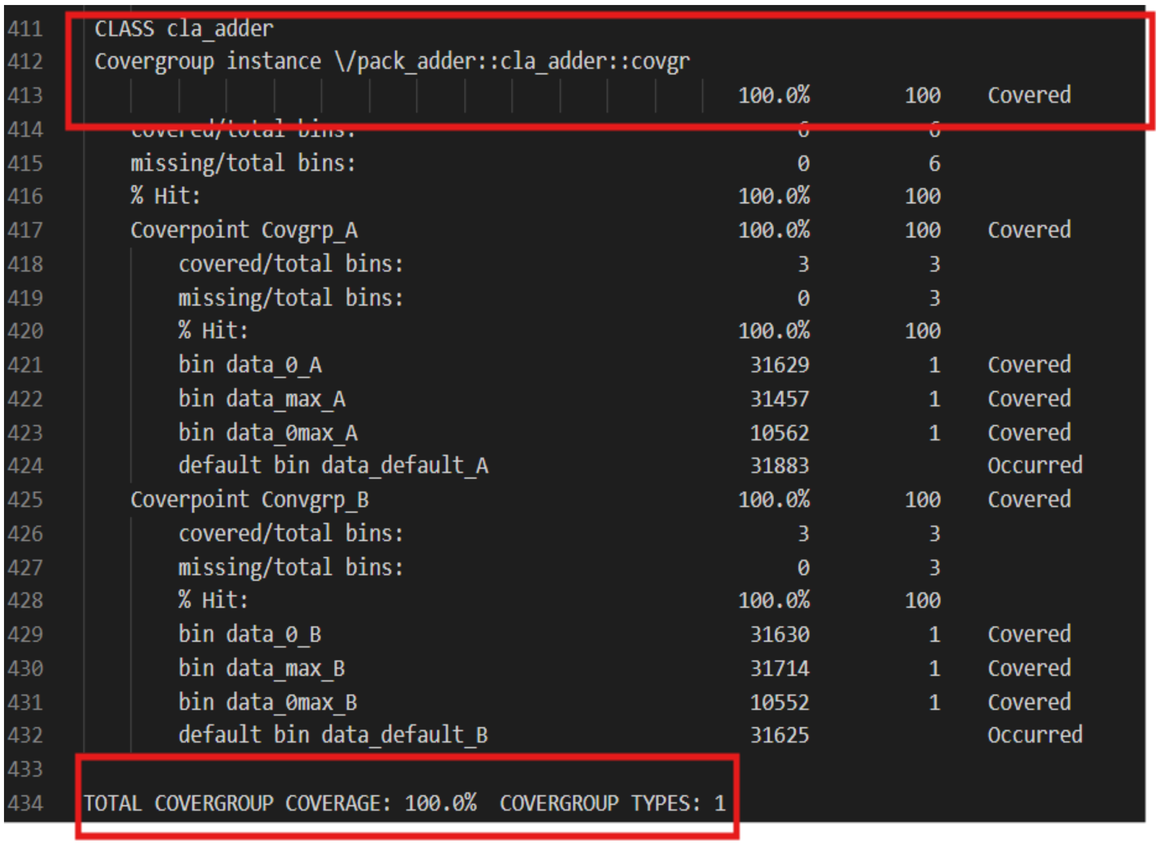


```
Branch Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Branches                           2         2         0     100.0

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

Branch Coverage for file adder.v --

------------------------------------IF Branch------------------------------------
    11                                    102994     Count coming in to IF
    11             1                        9755           if (reset)
    13             1                       93239           else
Branch totals: 2 hits of 2 branches = 100.0%
```



```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       30        30         0     100.0

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

Toggle Coverage for File adder.v --

       Line                                 Node      1H->0L      0L->1H   "Coverage"
---------------------------------------------------------------------------------------
          2                                  clk           1           1      100.00
          3                                reset           1           1      100.00
          4                                 A[3]           1           1      100.00
          4                                 A[2]           1           1      100.00
          4                                 A[1]           1           1      100.00
          4                                 A[0]           1           1      100.00
          5                                 B[3]           1           1      100.00
          5                                 B[2]           1           1      100.00
          5                                 B[1]           1           1      100.00
          5                                 B[0]           1           1      100.00
          6                                 C[4]           1           1      100.00
          6                                 C[3]           1           1      100.00
          6                                 C[2]           1           1      100.00
          6                                 C[1]           1           1      100.00
          6                                 C[0]           1           1      100.00
```

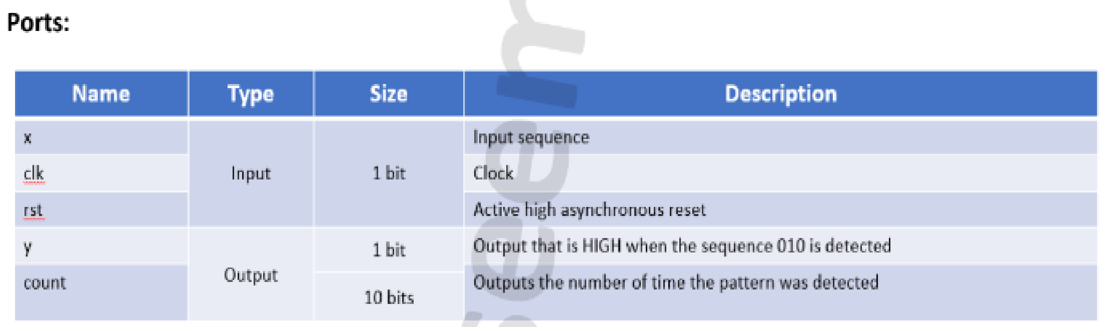
9. Function Coverage



```
CLASS cla_adder
Covergroup instance \/pack_adder::cla_adder::covgr
                                                     100.0%        100    Covered
    covered/total bins:                                   6          6
    missing/total bins:                                   0          6
    % Hit:                                           100.0%        100
    Coverpoint Covgrp_A                              100.0%        100    Covered
        covered/total bins:                               3          3
        missing/total bins:                               0          3
        % Hit:                                       100.0%        100
        bin data_0_A                                  31629          1    Covered
        bin data_max_A                                31457          1    Covered
        bin data_0max_A                               10562          1    Covered
        default bin data_default_A                    31883               Occurred
    Coverpoint Convgrp_B                             100.0%        100    Covered
        covered/total bins:                               3          3
        missing/total bins:                               0          3
        % Hit:                                       100.0%        100
        bin data_0_B                                  31630          1    Covered
        bin data_max_B                                31714          1    Covered
        bin data_0max_B                               10552          1    Covered
        default bin data_default_B                    31625               Occurred

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```

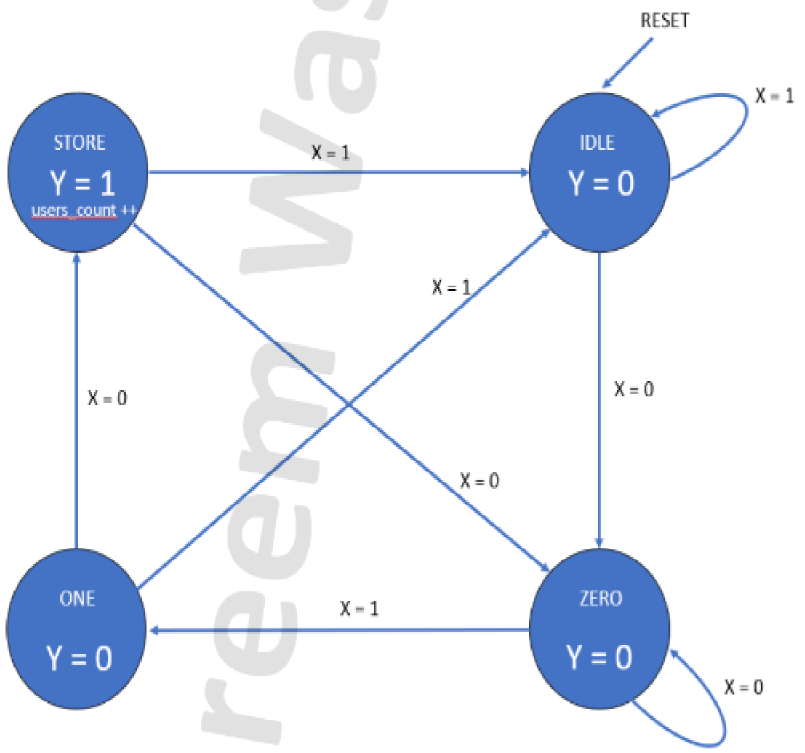
Question 3

➢ FSM

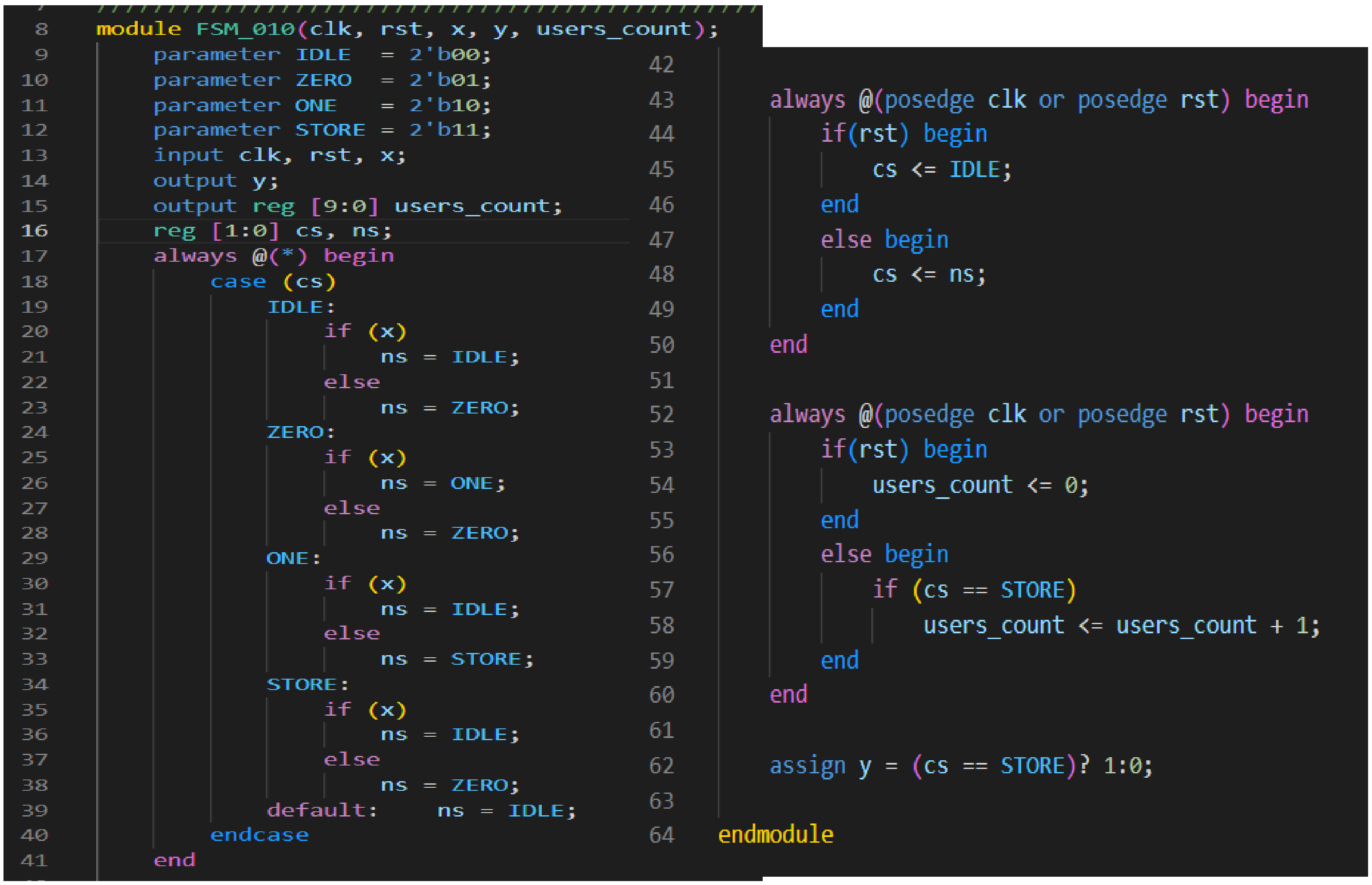


Ports:

Name	Type	Size	Description
x	Input	1 bit	Input sequence
clk			Clock
rst			Active high asynchronous reset
y	Output	1 bit	Output that is HIGH when the sequence 010 is detected
count		10 bits	Outputs the number of time the pattern was detected

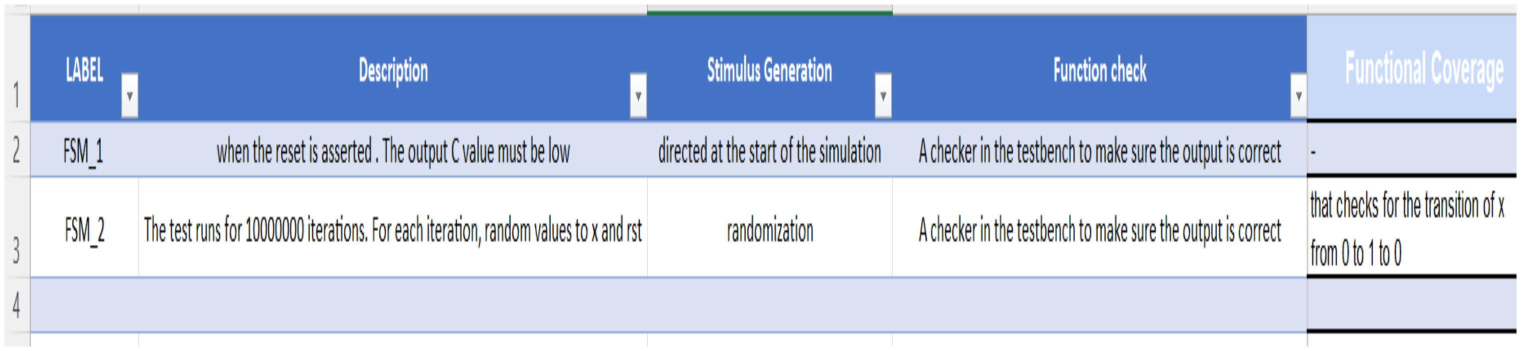


1. Code Design



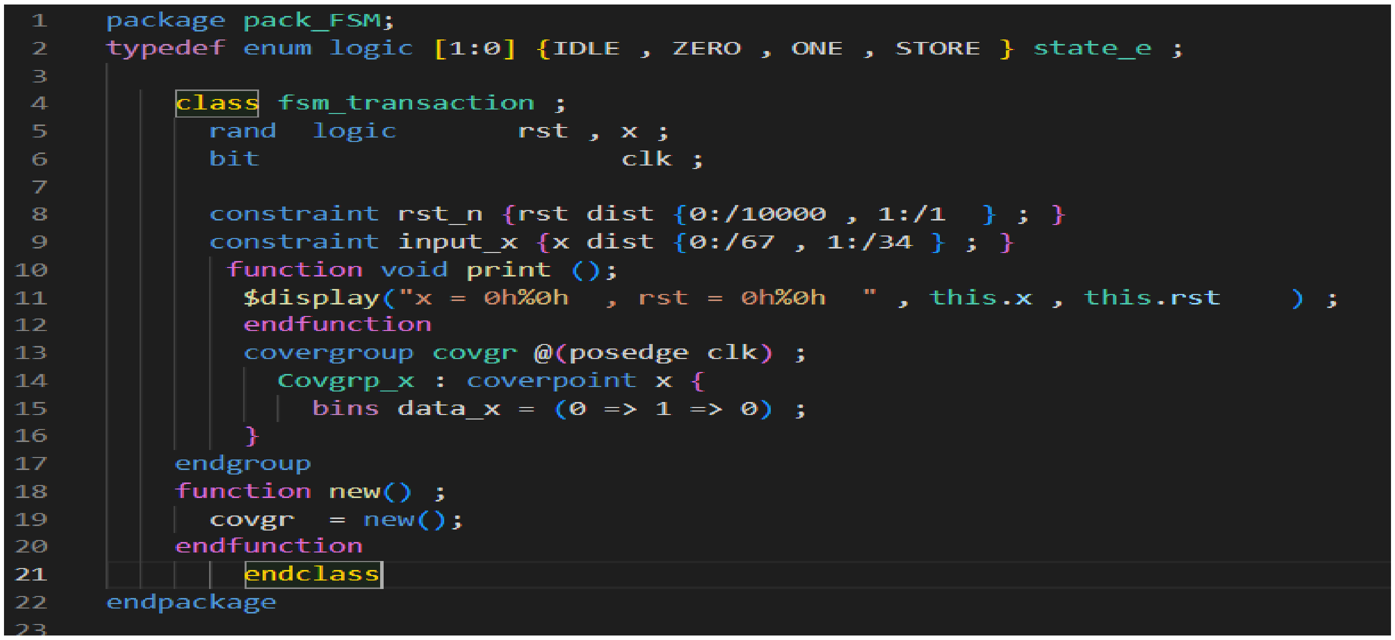
```verilog
//////////////////////////////////////////////////////////////
module FSM_010(clk, rst, x, y, users_count);
    parameter IDLE  = 2'b00;
    parameter ZERO  = 2'b01;
    parameter ONE   = 2'b10;
    parameter STORE = 2'b11;
    input clk, rst, x;
    output y;
    output reg [9:0] users_count;
    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
```

2. Verification plan



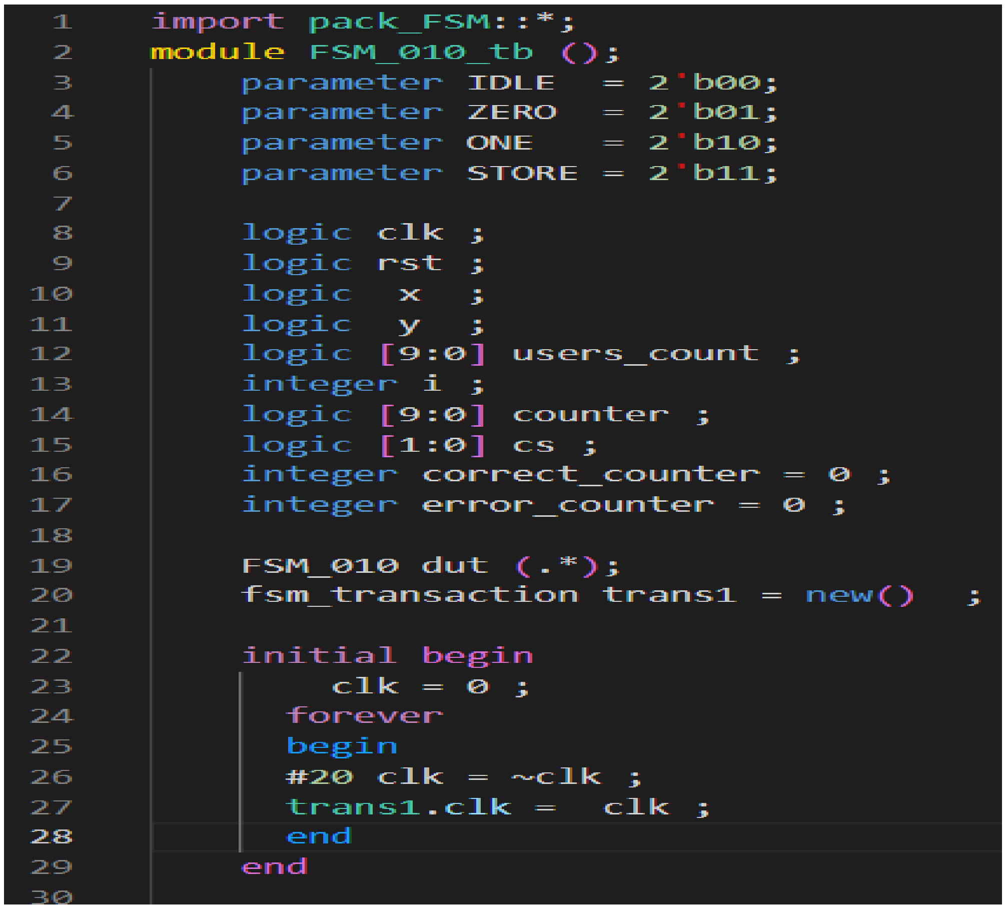
	LABEL	Description	Stimulus Generation	Function check	Functional Coverage
2	FSM_1	when the reset is asserted . The output C value must be low	directed at the start of the simulation	A checker in the testbench to make sure the output is correct	-
3	FSM_2	The test runs for 10000000 iterations. For each iteration, random values to x and rst	randomization	A checker in the testbench to make sure the output is correct	that checks for the transition of x from 0 to 1 to 0
4					

3. FSM Package

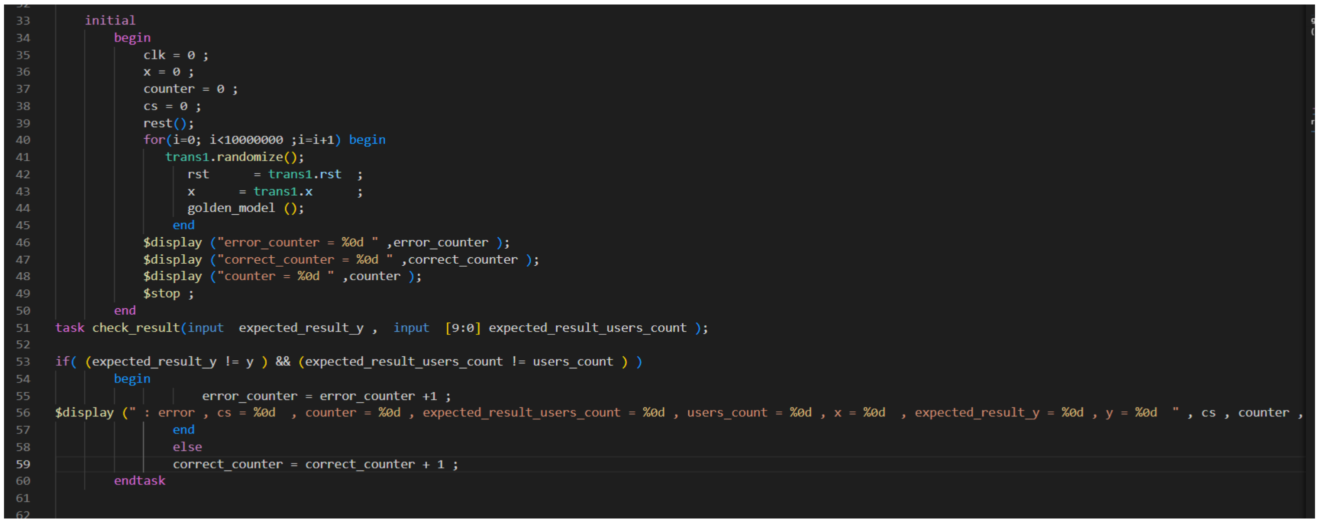


```
package pack_FSM;
typedef enum logic [1:0] {IDLE , ZERO , ONE , STORE } state_e ;

        class fsm_transaction ;
          rand  logic        rst , x ;
          bit                      clk ;

          constraint rst_n {rst dist {0:/10000 , 1:/1   } ; }
          constraint input_x {x dist {0:/67 , 1:/34 } ; }
           function void print ();
            $display("x = 0h%0h   , rst = 0h%0h   " , this.x , this.rst     ) ;
            endfunction
            covergroup covgr @(posedge clk) ;
              Covgrp_x : coverpoint x {
                bins data_x = (0 => 1 => 0) ;
            }
        endgroup
        function new() ;
          covgr  = new();
        endfunction
            endclass
endpackage
```

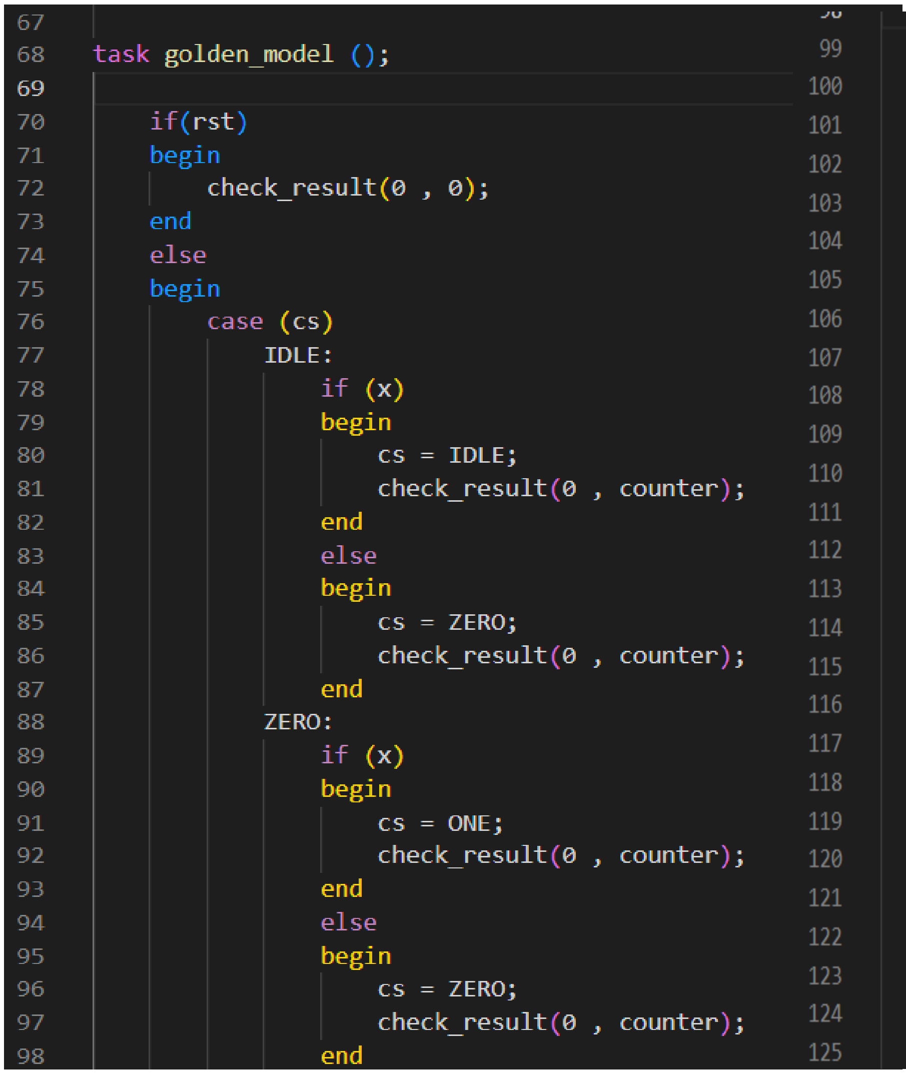
4. FSM Testbench



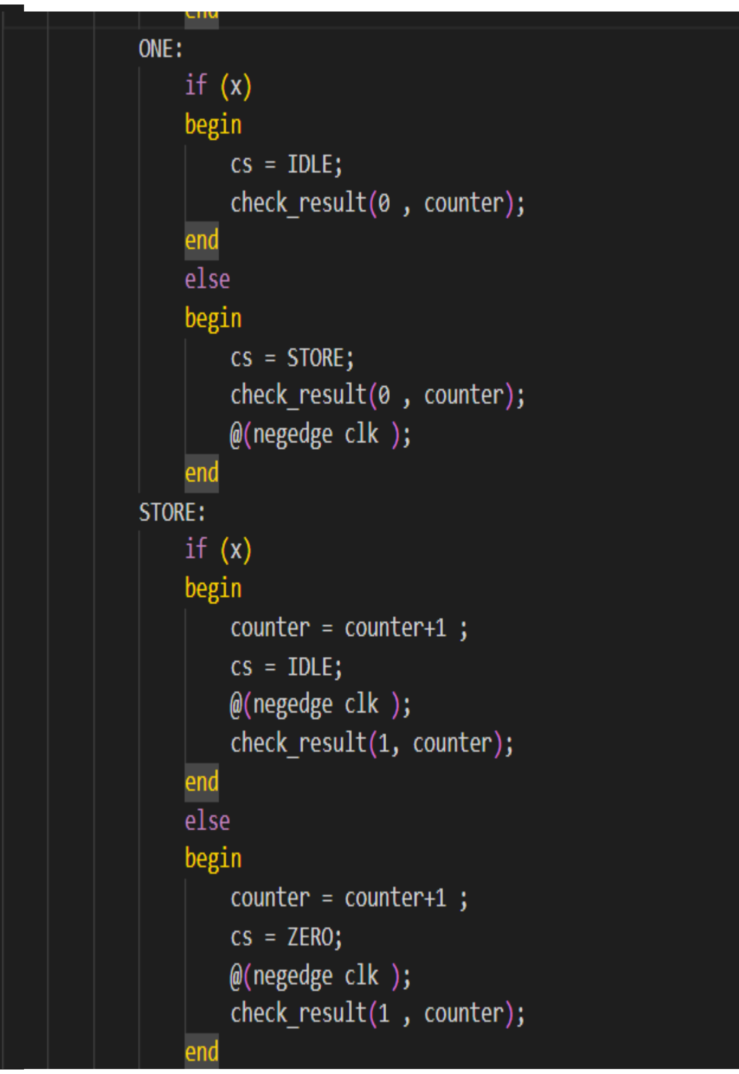
```
import pack_FSM::*;
module FSM_010_tb ();
    parameter IDLE   = 2'b00;
    parameter ZERO   = 2'b01;
    parameter ONE    = 2'b10;
    parameter STORE = 2'b11;

    logic clk ;
    logic rst ;
    logic  x   ;
    logic  y   ;
    logic [9:0] users_count ;
    integer i ;
    logic [9:0] counter ;
    logic [1:0] cs ;
    integer correct_counter = 0 ;
    integer error_counter = 0 ;

    FSM_010 dut (.*);
    fsm_transaction trans1 = new()   ;

    initial begin
        clk = 0 ;
      forever
      begin
      #20 clk = ~clk ;
      trans1.clk =  clk ;
      end
    end
```



```verilog
    initial
        begin
            clk = 0 ;
            x = 0 ;
            counter = 0 ;
            cs = 0 ;
            rest();
            for(i=0; i<10000000 ;i=i+1) begin
               trans1.randomize();
                  rst       = trans1.rst   ;
                  x      = trans1.x       ;
                  golden_model ();
                end
            $display ("error_counter = %0d " ,error_counter );
            $display ("correct_counter = %0d " ,correct_counter );
            $display ("counter = %0d " ,counter );
            $stop ;
        end
task check_result(input  expected_result_y ,  input  [9:0] expected_result_users_count );

if( (expected_result_y != y ) && (expected_result_users_count != users_count ) )
        begin
                  error_counter = error_counter +1 ;
$display (" : error , cs = %0d  , counter = %0d , expected_result_users_count = %0d , users_count = %0d , x = %0d  , expected_result_y = %0d , y = %0d  " , cs , counter ,
                end
                else
                correct_counter = correct_counter + 1 ;
        endtask
```

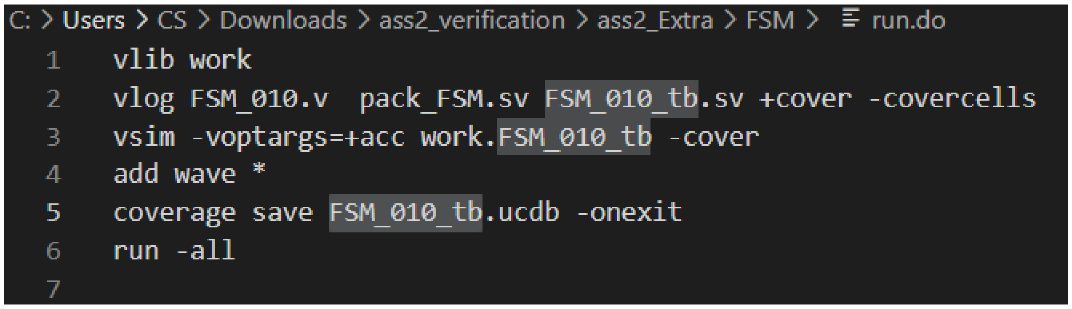


```verilog
task golden_model ();

    if(rst)
    begin
        check_result(0 , 0);
    end
    else
    begin
        case (cs)
            IDLE:
                if (x)
                begin
                    cs = IDLE;
                    check_result(0 , counter);
                end
                else
                begin
                    cs = ZERO;
                    check_result(0 , counter);
                end
            ZERO:
                if (x)
                begin
                    cs = ONE;
                    check_result(0 , counter);
                end
                else
                begin
                    cs = ZERO;
                    check_result(0 , counter);
                end
```



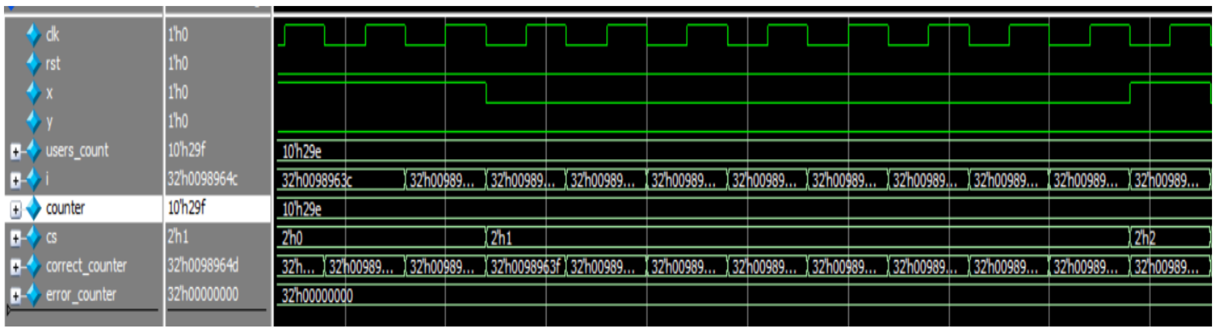
```verilog
            ONE:
                if (x)
                begin
                    cs = IDLE;
                    check_result(0 , counter);
                end
                else
                begin
                    cs = STORE;
                    check_result(0 , counter);
                    @(negedge clk );
                end
            STORE:
                if (x)
                begin
                    counter = counter+1 ;
                    cs = IDLE;
                    @(negedge clk );
                    check_result(1, counter);
                end
                else
                begin
                    counter = counter+1 ;
                    cs = ZERO;
                    @(negedge clk );
                    check_result(1 , counter);
                end
```

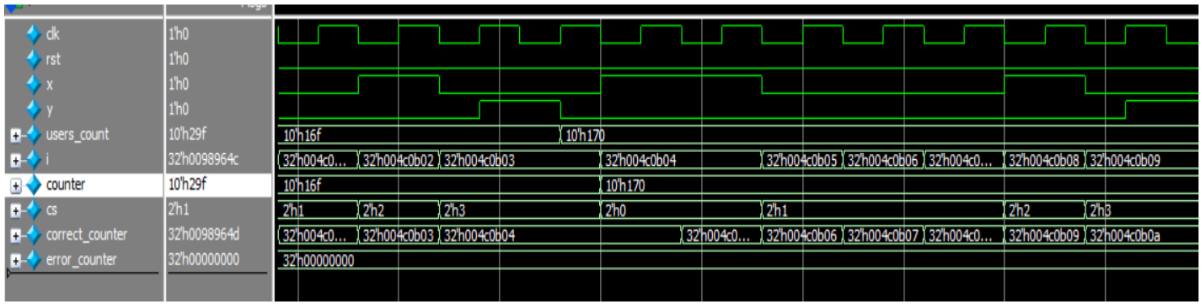
5. Do File



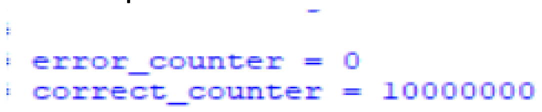
```
C: > Users > CS > Downloads > ass2_verification > ass2_Extra > FSM > run.do
vlib work
vlog FSM_010.v  pack_FSM.sv FSM_010_tb.sv +cover -covercells
vsim -voptargs=+acc work.FSM_010_tb -cover
add wave *
coverage save FSM_010_tb.ucdb -onexit
run -all
```

6. Waveform



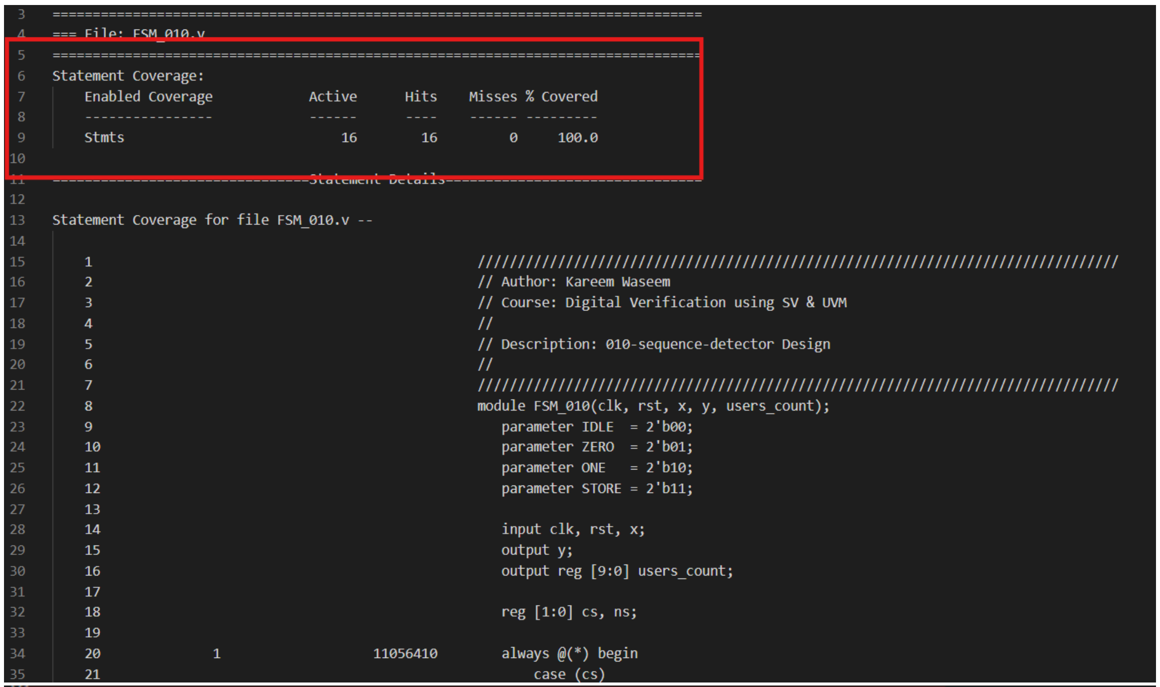


7. Transcript

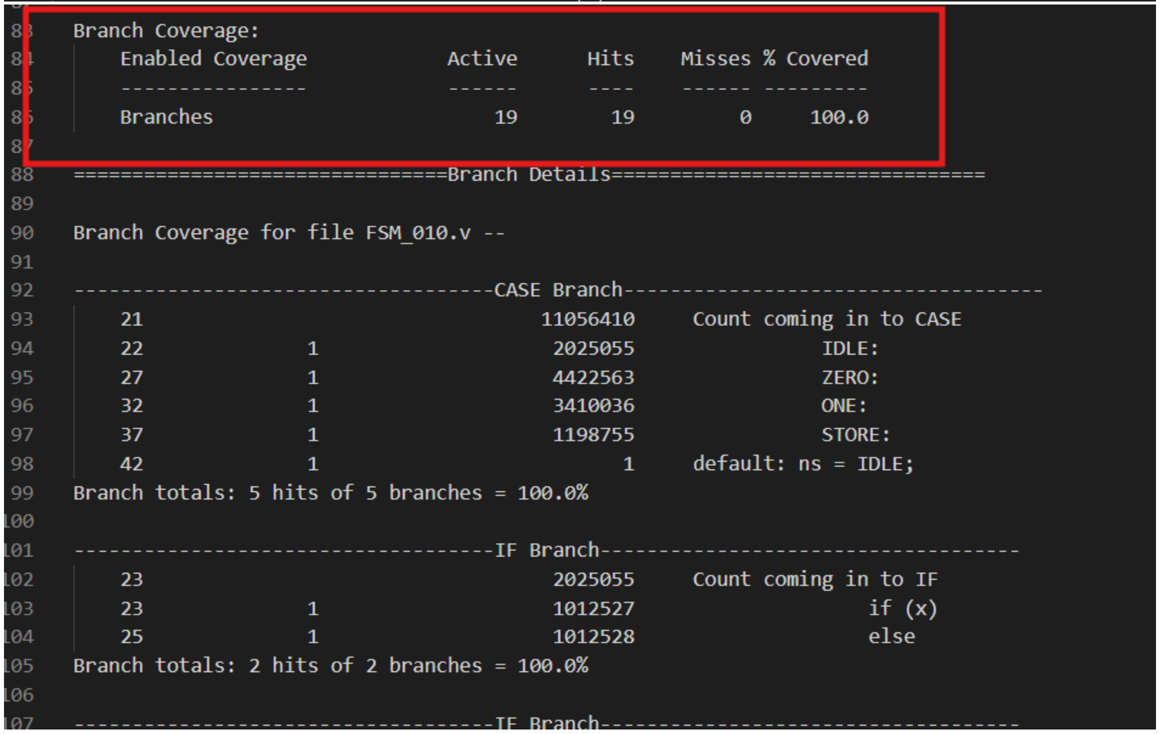


```
error_counter = 0
correct_counter = 10000000
```

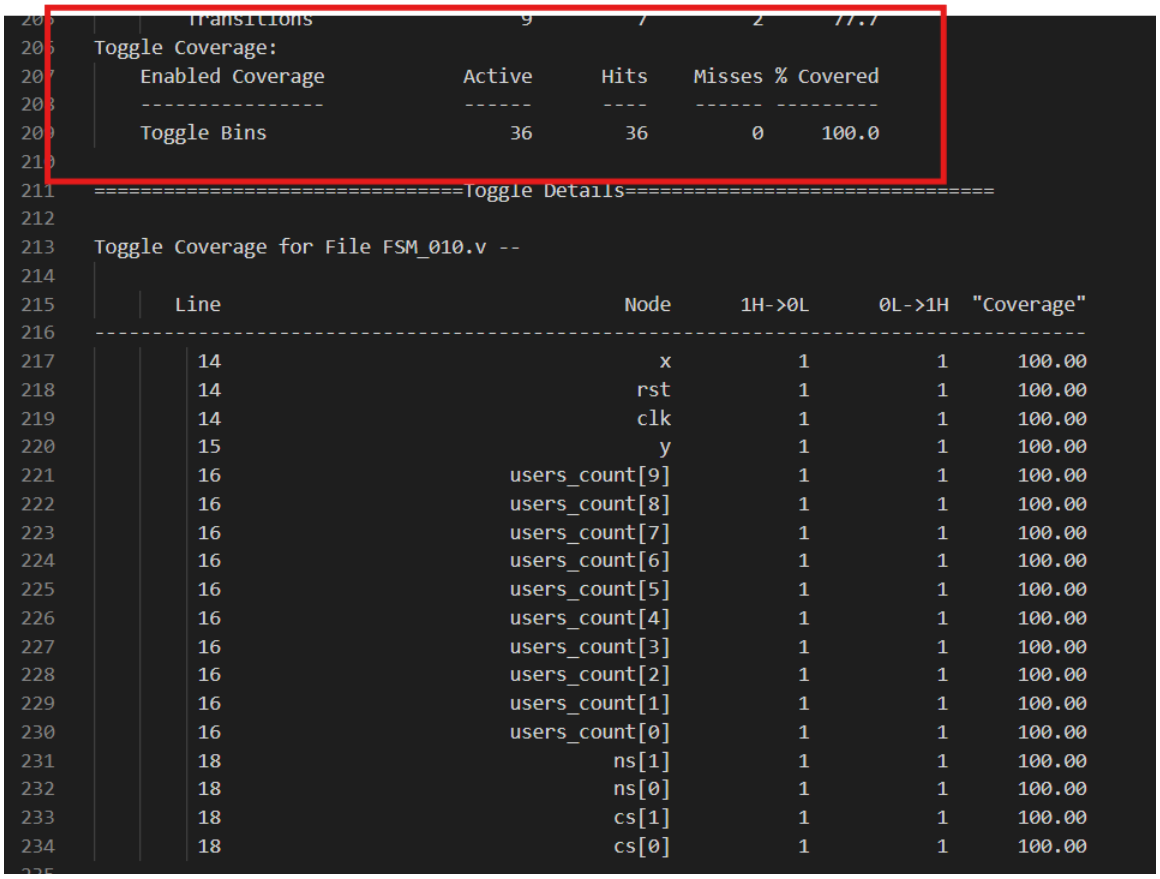
8. Code Coverage



```
================================================================================
=== File: FSM_010.v
================================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                             16        16         0     100.0

--------------------------------Statement Details--------------------------------

Statement Coverage for file FSM_010.v --

    1                                                ////////////////////////////////////////////////////////////////////////////////
    2                                                // Author: Kareem Waseem
    3                                                // Course: Digital Verification using SV & UVM
    4                                                //
    5                                                // Description: 010-sequence-detector Design
    6                                                //
    7                                                ////////////////////////////////////////////////////////////////////////////////
    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 [9:0] users_count;
    17
    18                                                  reg [1:0] cs, ns;
    19
    20              1                 11056410          always @(*) begin
    21                                                     case (cs)
```

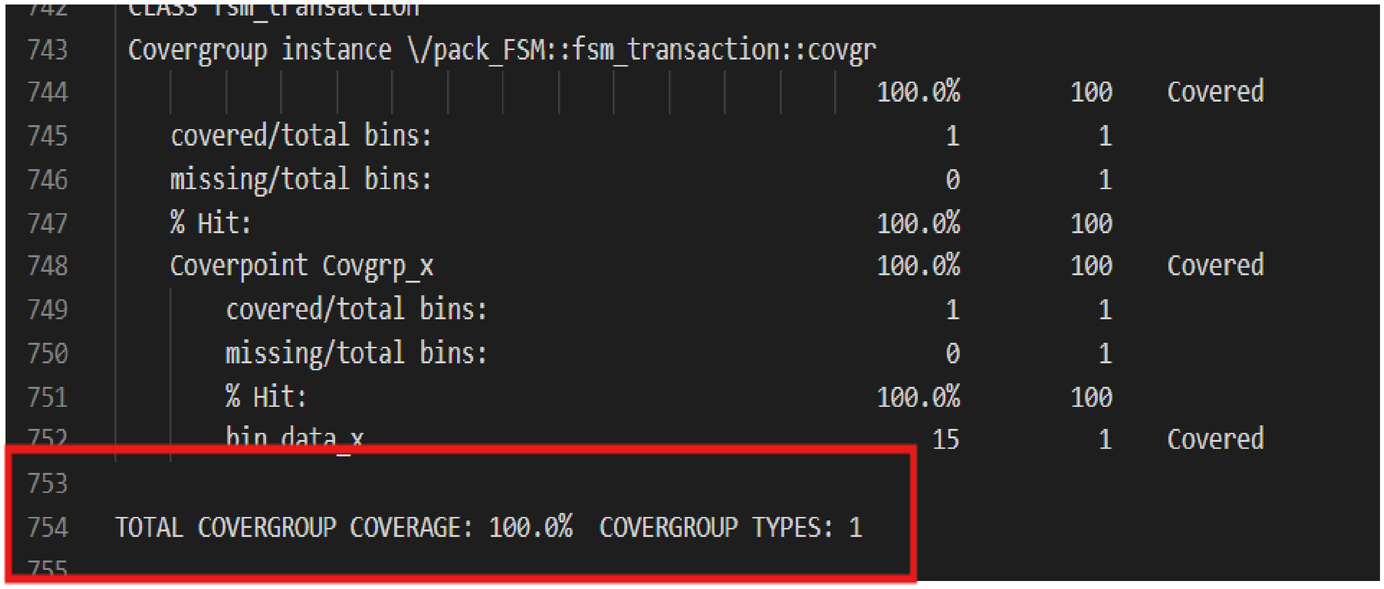


```
Branch Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Branches                          19        19         0     100.0

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

Branch Coverage for file FSM_010.v --

------------------------------------CASE Branch------------------------------------
    21                                  11056410     Count coming in to CASE
    22              1                    2025055                 IDLE:
    27              1                    4422563                 ZERO:
    32              1                    3410036                 ONE:
    37              1                    1198755                 STORE:
    42              1                          1     default: ns = IDLE;
Branch totals: 5 hits of 5 branches = 100.0%

------------------------------------IF Branch------------------------------------
    23                                   2025055     Count coming in to IF
    23              1                    1012527                     if (x)
    25              1                    1012528                     else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
```



```
        Transitions                      9         7         2      77.7
    Toggle Coverage:
        Enabled Coverage            Active      Hits    Misses % Covered
        ----------------            ------      ----    ------ ---------
        Toggle Bins                     36        36         0     100.0

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

    Toggle Coverage for File FSM_010.v --

           Line                                 Node      1H->0L      0L->1H  "Coverage"
    ---------------------------------------------------------------------------------------
             14                                    x           1           1      100.00
             14                                  rst           1           1      100.00
             14                                  clk           1           1      100.00
             15                                    y           1           1      100.00
             16                       users_count[9]           1           1      100.00
             16                       users_count[8]           1           1      100.00
             16                       users_count[7]           1           1      100.00
             16                       users_count[6]           1           1      100.00
             16                       users_count[5]           1           1      100.00
             16                       users_count[4]           1           1      100.00
             16                       users_count[3]           1           1      100.00
             16                       users_count[2]           1           1      100.00
             16                       users_count[1]           1           1      100.00
             16                       users_count[0]           1           1      100.00
             18                                ns[1]           1           1      100.00
             18                                ns[0]           1           1      100.00
             18                                cs[1]           1           1      100.00
             18                                cs[0]           1           1      100.00
```

10.Function Coverage



```
 CLASS fsm_transaction
 Covergroup instance \/pack_FSM::fsm_transaction::covgr
                                                 100.0%        100    Covered
    covered/total bins:                              1           1
    missing/total bins:                              0           1
    % Hit:                                      100.0%         100
    Coverpoint Covgrp_x                         100.0%         100    Covered
        covered/total bins:                          1           1
        missing/total bins:                          0           1
        % Hit:                                  100.0%         100
        bin data_x                                  15           1    Covered

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1
```