Data Types & Constrained Random

Assignment2_extra

By: Mohamed Sayed Mohamed Soliman

Question 1

- Write the SystemVerilog code to:

1. Declare a 2-state array, my_array, that holds four 12-bit values
2. initialize my_array in initial block so that:
   - my_array[0] = 12'h012
   - my_array[1] = 12'h345
   - my_array[2] = 12'h678
   - my_array[3] = 12'h9AB
3. Traverse my_array and print out bits [5:4] of each 12-bit element
   - Using a for loop
   - Using a foreach loop

1. Code of the design

```
1   module test_dynamic_array_extra ;
2
3   bit [11:0]  my_array [0:3] ;
4   integer i ;
5   initial begin
6       my_array = '{12'h012 , 12'h345 , 12'h678 , 12'h9AB } ;
7
8       $display("print bt using foreach");
9
10      foreach(my_array[j])
11      begin
12          $display("my_array[%0d][5:4] = %b", j , my_array[j][5:4] );
13      end
14
15      $display("print bt using for loop");
16
17      for( i = 0 ; i < 4 ; i = i+1)
18      begin
19          $display("my_array[%0d][5:4] = %b", i , my_array[i][5:4] );
20      end
21  end
22
23  endmodule
24
```

2. Result of the simulation

```
# print bt using foreach
# my_array[0][5:4] = 01
# my_array[1][5:4] = 00
# my_array[2][5:4] = 11
# my_array[3][5:4] = 10
# print bt using for loop
# my_array[0][5:4] = 01
# my_array[1][5:4] = 00
# my_array[2][5:4] = 11
# my_array[3][5:4] = 10
```

Question 2

➢ ALU

- Create a package in a file that has the following
    - typedef enum for the opcode
    - Create a class to randomize all ALU inputs
        - Constraint the reset to be low most of the time
        - Use the typedef enum to declare the opcode variable of the class
- Create another file that has the testbench module
    - Import the above package
    - Use the typedef enum for the opcode variable
    - Your testbench will use constrained randomization to drive the stimulus inside of a repeat block
    - Make the testbench self-checking using a check_result task
    - Monitor the output to display errors if occurred
    - Use a do file to run the testbench
    - Generate a code coverage report (100% design code coverage is expected. Less than100% must be justified.)

1. Code 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 = 0 ;
      endcase
   end // always @ *

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

endmodule
```

2. Verification plan

LABEL	Description	Stimulus Generation	Function check
alu_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
alu_2	The test runs for 2000 iterations. For each iteration, random values to A ,B and opcode	Randomization	A checker in the testbench to make sure the output is correct

3. alu Package

```
package pack_ALU;
typedef enum logic [1:0] {Add , Sub , Not_A , ReductionOR_B } reg_e ;
localparam MAXPOS = 7  ;
localparam MAXNEG = -8 ;
localparam zero = 0 ;

    class cla_ALU ;
     rand  logic signed [3:0] A, B ;
     rand  reg_e Opcode ;
     rand  logic       reset ;

     constraint rst_n {reset dist {0:/99 , 1:/1  } ; }

     constraint input_opcode {Opcode dist {[0:3]:/100 } ; }

      function void print ();
       $display("A = 0h%0h  , B = 0h%0h , Opcode = 0h%0h , reset = 0h%0h  " , this.A , this.B , this.Opcode , this.reset   ) ;
       endfunction

       endclass

endpackage
```

4. alu Testbench

```
1   import pack_ALU::*;
2
3   module ALU_4_bit_tb ();
4
5       localparam MAXPOS = 7  ;
6       localparam MAXNEG = -8 ;
7
8       logic clk ;
9       logic reset ;
10      logic   [1:0]  Opcode;
11      logic signed [3:0] A ;
12      logic signed [3:0] B ;
13      logic signed [4:0] C ;
14      integer i ;
15
16     localparam           Add            = 2'b00; // A + B
17     localparam           Sub            = 2'b01; // A - B
18     localparam           Not_A          = 2'b10; // ~A
19     localparam           ReductionOR_B  = 2'b11; // |B
20
21      logic [4:0] no_case ;
22      integer correct_counter = 0 ;
23      integer error_counter = 0 ;
24
25  ALU_4_bit dut (
26          .clk(clk),
27          .reset(reset),
28          .Opcode(Opcode),
29          .A(A),
30          .B(B),
31          .C(C)
32      );
33
```

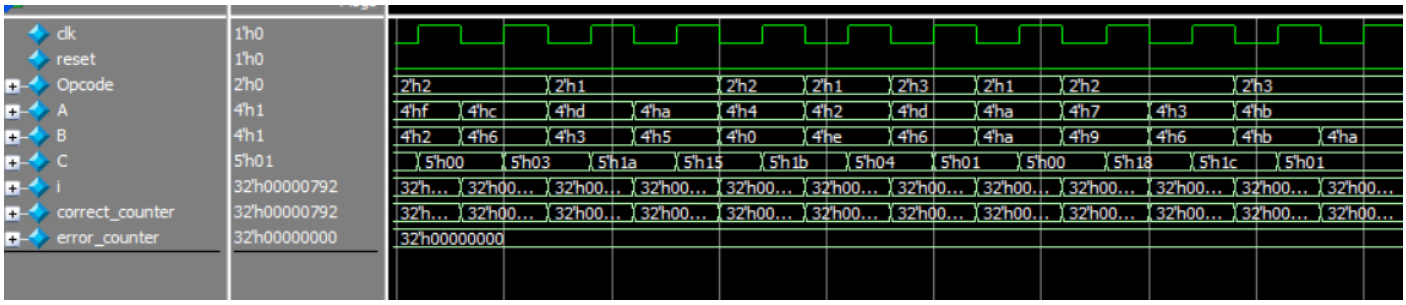
```
34      always #20 clk = ~clk ;
35
36      cla_ALU trans1 , trans2 ;
37
38      initial
39          begin
40              clk = 0 ;
41              A = 0 ;
42              B = 0 ;
43              Opcode = 0 ;
44              no_case = 0 ;
45              trans1 = new();
46              rest();
47              for(i=0; i<2000 ;i=i+1) begin
48                 trans1.randomize();
49                    A       = trans1.A      ;
50                    B       = trans1.B      ;
51                    Opcode  = trans1.Opcode ;
52                    reset   = trans1.reset  ;
53                      if(reset)
54                      begin
55                          check_result(0);
56                      end
57                      else
58                      begin
59                    case (Opcode)
60                      Add:             check_result(A + B);
61                      Sub:             check_result(A - B);
62                      Not_A:           check_result(~A);
63                      ReductionOR_B:   check_result(|B);
64                     default:  check_result(0);
65                endcase
66                  end
67
68          end
69
70
71          $display ("error_counter = %0d " ,error_counter );
72          $display ("correct_counter = %0d " ,correct_counter );
73          $stop ;
74
75      end
76
77
78      task check_result(input  signed [4:0] expected_result  );
79          @(negedge clk );
80          if(expected_result != C )
81              begin
82                  error_counter = error_counter +1 ;
83              $display ("%0t:error no_case = %0d  ; ==> A =  %0d , B =  %0d ==> output equal C = %0d should be %0d " , $time , no_case  ,A , B  , expected_result , C );
84              end
85              else
86              correct_counter = correct_counter + 1 ;
87
88
89      endtask
90
```

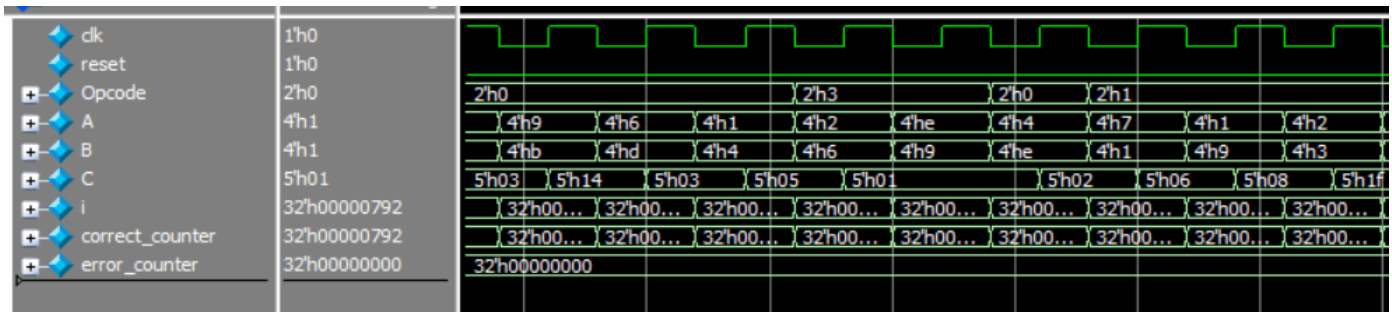
```
90
91              task rest ();
92                   reset = 1 ;
93                   #20;
94                   reset = 0 ;
95
96              endtask
97
98
99      endmodule
```

5. Do File

```
: > Users > CS > Downloads > ass2_verification > ass2_Extra > ass2_alu > ≡ run.do
1   vlib work
2   vlog ALU_4_bit.v ALU_4_bit_tb.sv +cover -covercells
3   vsim -voptargs=+acc work.ALU_4_bit_tb -cover
4   add wave *
5   coverage save alu_tb.ucdb -onexit
6   run -all
7
```

6. Waveform





7. Transcript

```
error_counter = 0
correct_counter = 2000
** Note: $stop    : ALU_4_bit_tb.sv(73)
```

8. Code Coverage

```
================================================================================
=== File: ALU_4_bit.v
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         0     100.0

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

Statement Coverage for file ALU_4_bit.v --

    1                                           module ALU_4_bit (
    2                                               input  clk,
    3                                               input  reset,
    4                                               input  [1:0] Opcode,   // The opcode
    5                                               input  signed [3:0] A, // Input data A in 2's complement
    6                                               input  signed [3:0] B, // Input data B in 2's complement
    7
    8                                               output reg signed [4:0] C // ALU output in 2's complement
    9
    10                                                  );
    11
    12                                          reg signed [4:0]       Alu_out; // ALU output in 2's complement
    13
    14                                          localparam          Add            = 2'b00; // A + B
    15                                          localparam          Sub            = 2'b01; // A - B
    16                                          localparam          Not_A            = 2'b10; // ~A
    17                                          localparam          ReductionOR_B  = 2'b11; // |B
    18
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         6         6         0     100.0

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

Branch Coverage for file ALU_4_bit.v --

-----------------------------------CASE Branch-----------------------------------
    21                                    2000     Count coming in to CASE
    22              1                      521           Add:            Alu_out = A + B;
    23              1                      497           Sub:            Alu_out = A - B;
    24              1                      466           Not_A:          Alu_out = ~A;
    25              1                      516           ReductionOR_B:  Alu_out = |B;
    26              1                        E             default: Alu_out = 0 ;
Branch totals: 4 hits of 4 branches = 100.0%

-----------------------------------IF Branch-----------------------------------
    32                                    1974     Count coming in to IF
    32              1                       29          if (reset)
    34              1                     1945          else
Branch totals: 2 hits of 2 branches = 100.0%
```

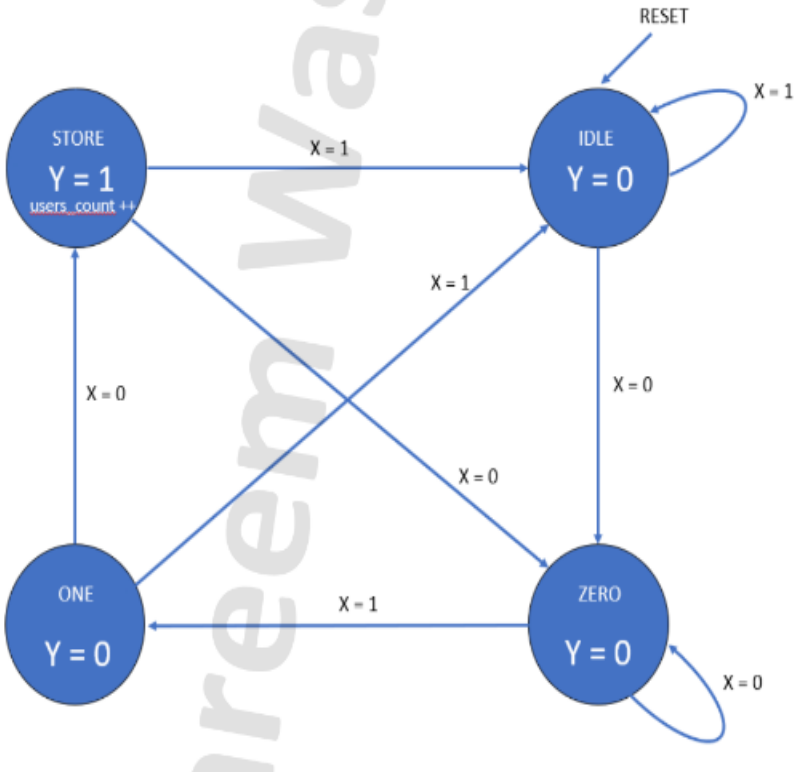
```
        Transitions                 0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     44        44         0     100.0

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

Toggle Coverage for File ALU_4_bit.v --

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

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



Jareem Waseem

1. Code 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
```

2. Verification plan

LABEL	Description	Stimulus Generation	Function check
alu_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
alu_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

3. FSM Package

```
1   package pack_FSM;
2
3
4   typedef enum logic [1:0] {IDLE , ZERO , ONE , STORE } state_e ;
5
6       class fsm_transaction ;
7         rand  logic       rst , x ;
8
9         constraint rst_n {rst dist {0:/99 , 1:/1  } ; }
10
11        constraint input_x {x dist {0:/67 , 1:/34 } ; }
12
13        function void print ();
14         $display("x = 0h%0h  , rst = 0h%0h  " , this.x , this.rst    ) ;
15         endfunction
16
17         endclass
18
19  endpackage
20
```

4. FSM Testbench

```
1   import pack_FSM::*;
2
3   module FSM_010_tb ();
4
5       parameter IDLE  = 2'b00;
6       parameter ZERO  = 2'b01;
7       parameter ONE   = 2'b10;
8       parameter STORE = 2'b11;
9
10      logic clk ;
11      logic rst ;
12      logic  x  ;
13      logic  y  ;
14      logic [9:0] users_count ;
15
16
17      integer i ;
18      logic [9:0] counter ;
19
20      logic [1:0] cs ;
21      integer correct_counter = 0 ;
22      integer error_counter = 0 ;
23
24      FSM_010 dut (.*);
25
26      always #20 clk = ~clk ;
27
28      fsm_transaction trans1 , trans2 ;
```

```
29
30      initial
31          begin
32              clk = 0 ;
33              x = 0 ;
34              counter = 0 ;
35              cs = 0 ;
36              trans1 = new();
37              rest();
38              for(i=0; i<10000000 ;i=i+1) begin
39                 trans1.randomize();
40                    rst      = trans1.rst  ;
41                    x      = trans1.x      ;
42
43                    golden_model ();
44
45                 end
46              $display ("error_counter = %0d " ,error_counter );
47              $display ("correct_counter = %0d " ,correct_counter );
48              $display ("counter = %0d " ,counter );
49
50              $stop ;
51
52          end
53
```

```
52          end
53  task check_result(input  expected_result_y ,  input  [9:0] expected_result_users_count );
54
55  if( (expected_result_y != y ) && (expected_result_users_count != users_count ) )
56          begin
57                      error_counter = error_counter +1 ;
58  $display (" : error , cs = %0d  , counter = %0d , expected_result_users_count = %0d , users_count = %0d , x = %0d  , expected_result_y = %0d , y = %0d  " , cs , counter
59                  end
60                  else
61                  correct_counter = correct_counter + 1 ;
62
63
64          endtask
65
```

```
67
68  task golden_model ();
69
70      if(rst)
71      begin
72          check_result(0 , 0);
73      end
74      else
75      begin
76          case (cs)
77              IDLE:
78                  if (x)
79                  begin
80                      cs = IDLE;
81                      check_result(0 , counter);
82                  end
83                  else
84                  begin
85                      cs = ZERO;
86                      check_result(0 , counter);
87                  end
88              ZERO:
89                  if (x)
90                  begin
91                      cs = ONE;
92                      check_result(0 , counter);
93                  end
94                  else
95                  begin
96                      cs = ZERO;
97                      check_result(0 , counter);
98                  end
```

```
99              ONE:
100                 if (x)
101                 begin
102                     cs = IDLE;
103                     check_result(0 , counter);
104                 end
105                 else
106                 begin
107                     cs = STORE;
108                     check_result(0 , counter);
109                     @(negedge clk );
110                 end
111             STORE:
112                 if (x)
113                 begin
114                     counter = counter+1 ;
115                     cs = IDLE;
116                     @(negedge clk );
117                     check_result(1, counter);
118                 end
119                 else
120                 begin
121                     counter = counter+1 ;
122                     cs = ZERO;
123                     @(negedge clk );
124                     check_result(1 , counter);
125                 end
```

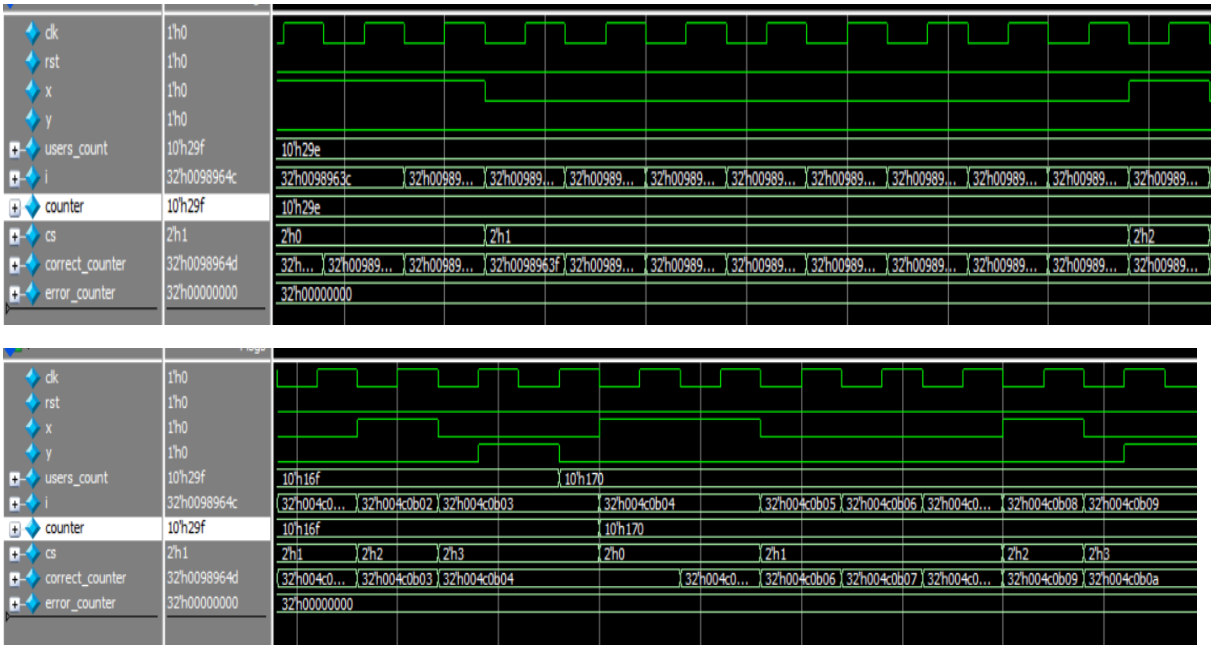
```
126                 default:    begin
127                     cs = IDLE;
128                     check_result(0 , counter);
129                 end
130             endcase
131             @(negedge clk );
132
133     end
134
135
136
137     endtask
138
139             task rest ();
140                 rst = 0 ;
141                 #1
142                 rst = 1 ;
143                 #20;
144                 rst = 0 ;
145
146             endtask
147
148
149 endmodule
```

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

```

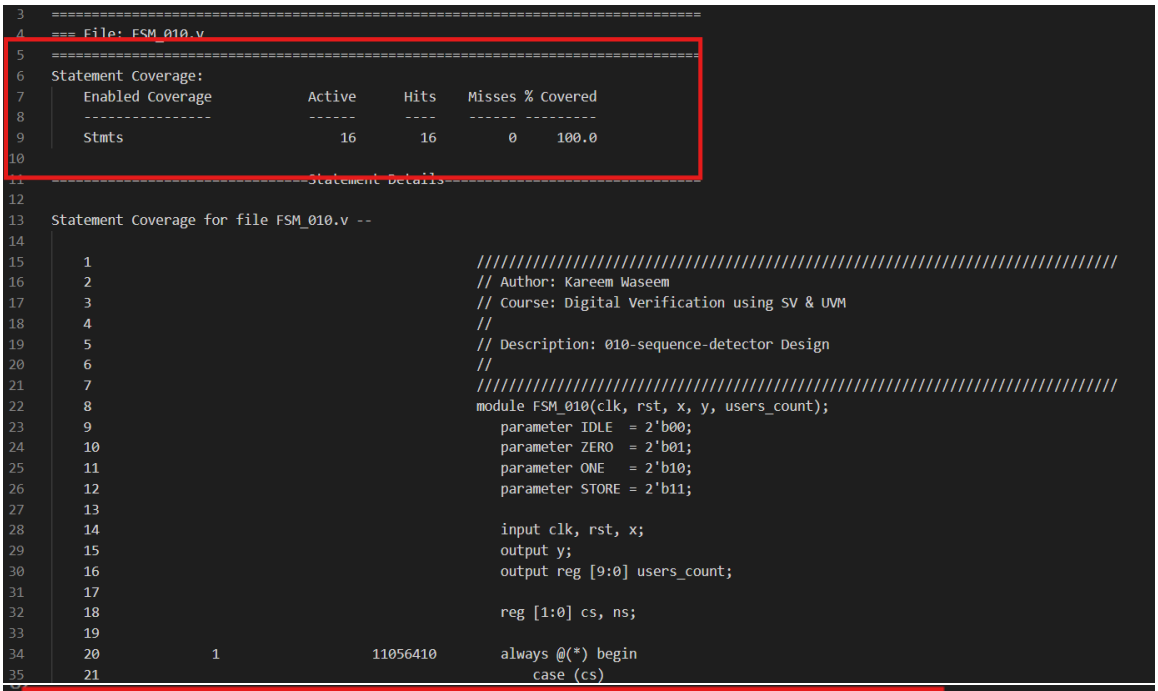
6. Waveform



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
```