يوسف أحمد محمد ابراهيم

# Assignment 2

Extra

# Question 1:

1. code :

```
module array();

    bit [11:0] my_array[4];

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

        $display("*** a. using for loop: ***");
        for(int i = 0; i < 4; i++)
            $display(">>> my_array[%0d][5:4] = %0d <<<", i, my_array[i][5:4]);

        $display("*** b. using foreach: ***");
        foreach(my_array[j])
            $display(">>> my_array[%0d][5:4] = %0d <<<", j, my_array[j][5:4]);
    end
endmodule
```

2. Results :

```
# *** a. using for loop: ***
# >>> my_array[          0][5:4] = 1 <<<
# >>> my_array[          1][5:4] = 0 <<<
# >>> my_array[          2][5:4] = 3 <<<
# >>> my_array[          3][5:4] = 2 <<<
# *** b. using foreach: ***
# >>> my_array[          0][5:4] = 1 <<<
# >>> my_array[          1][5:4] = 0 <<<
# >>> my_array[          2][5:4] = 3 <<<
# >>> my_array[          3][5:4] = 2 <<<
```

# Question 2:

1. RTL design:

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

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

        );

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

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

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

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

endmodule
```

*The design is working properly.*

## 2. Verification plan:

| | Label | Design Requirement Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|---|
| 2 | RESET test | When the reset is asserted, outputs should be low. | Directed at the start and the end of the simulation, and called in radomization step. | - | **check_result()** task checks that output is 0 during reset. |
| 3 | RANDOMIZE test | Output of the dut design is equal to the output of the golden model | Randomized test using **rand_stimuls** class with constraints 90% | - | **check_result()** verifies correct output using golden model. |

3. packege code:

```
1   package ALU_pkg;
2       typedef enum logic [1:0] {
5           Not_A,
6           ReductionOR_B
7       } opcode_e;
8
9
10      class alu_input;
11          rand logic         clk;
12          rand logic         reset;
13          rand opcode_e      Opcode;
14          rand logic [3:0]   A;
15          rand logic [3:0]   B;
16
17          // no need for constructor
18
19          constraint c1 {
20              reset dist {0 := 9, 1 := 1};
21          }
22      endclass
23  endpackage
```

# 4. Testbench code:

```
import ALU_pkg::*;

module ALU_tb();
    logic                clk;
    logic                reset;
    opcode_e             Opcode;
    logic signed  [3:0] A;
    logic signed  [3:0] B;
    logic signed  [4:0] C;
    logic signed  [4:0] C_exp;

    integer correct_count, error_count;

    alu_input my_input;

    ALU_4_bit dut(.*);

    initial begin
        clk = 0;
        forever
            #1 clk = ~clk;
    end

    initial begin
        correct_count = 0;
        error_count = 0;
        Opcode  = Add;
        A       = 0;
        B       = 0;

        //reset_test
        reset   = 0;
        check_result();
```

# 4. Testbench code:

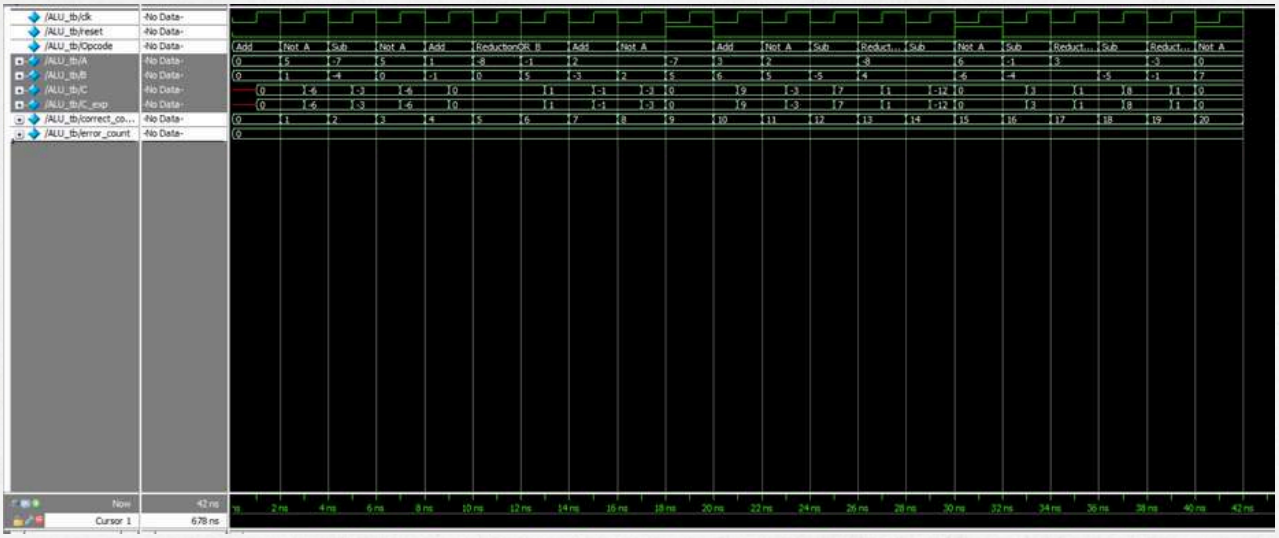
```
35          //randomize test
36          my_input = new();
37          repeat(20) begin
38              assert(my_input.randomize());
39              reset = my_input.reset;
40              Opcode = my_input.Opcode;
41              A = my_input.A;
42              B = my_input.B;
43              check_result();
44          end
45
46          $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
47          $stop;
48      end
49
50      always @(posedge clk, posedge reset) begin
51          if (reset)
52              C_exp <= 5'b0;
53          else
54              case (Opcode)
55                  Add:            C_exp <= A + B;
56                  Sub:            C_exp <= A - B;
57                  Not_A:          C_exp <= ~A;
58                  ReductionOR_B:  C_exp <= |B;
59              endcase
60      end
61
62
63      task check_result();
64          @(negedge clk);
65          if(C != C_exp) begin
66              $display("*** ERROR! at time %0t, C = %0d, Expected = %0d***",
67                  $time, C, C_exp);
68              error_count++;
69          end
70          else
71              correct_count++;
72
73      endtask
74  endmodule
```

5. Do file :

```
vlib work
vlog 2_ALU_pkg.sv 2_ALU.v 2_ALU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALU_tb -cover
add wave *
coverage save 2_ALU_tb.ucdb -onexit -du work.ALU
coverage exclude -src 2_ALU.v -line 26 -code s
coverage exclude -src 2_ALU.v -line 26 -code b

run -all
```

*Excluded default case as it will never occur.*

# 6. Qesta sim wave snippets (unsigned radix)

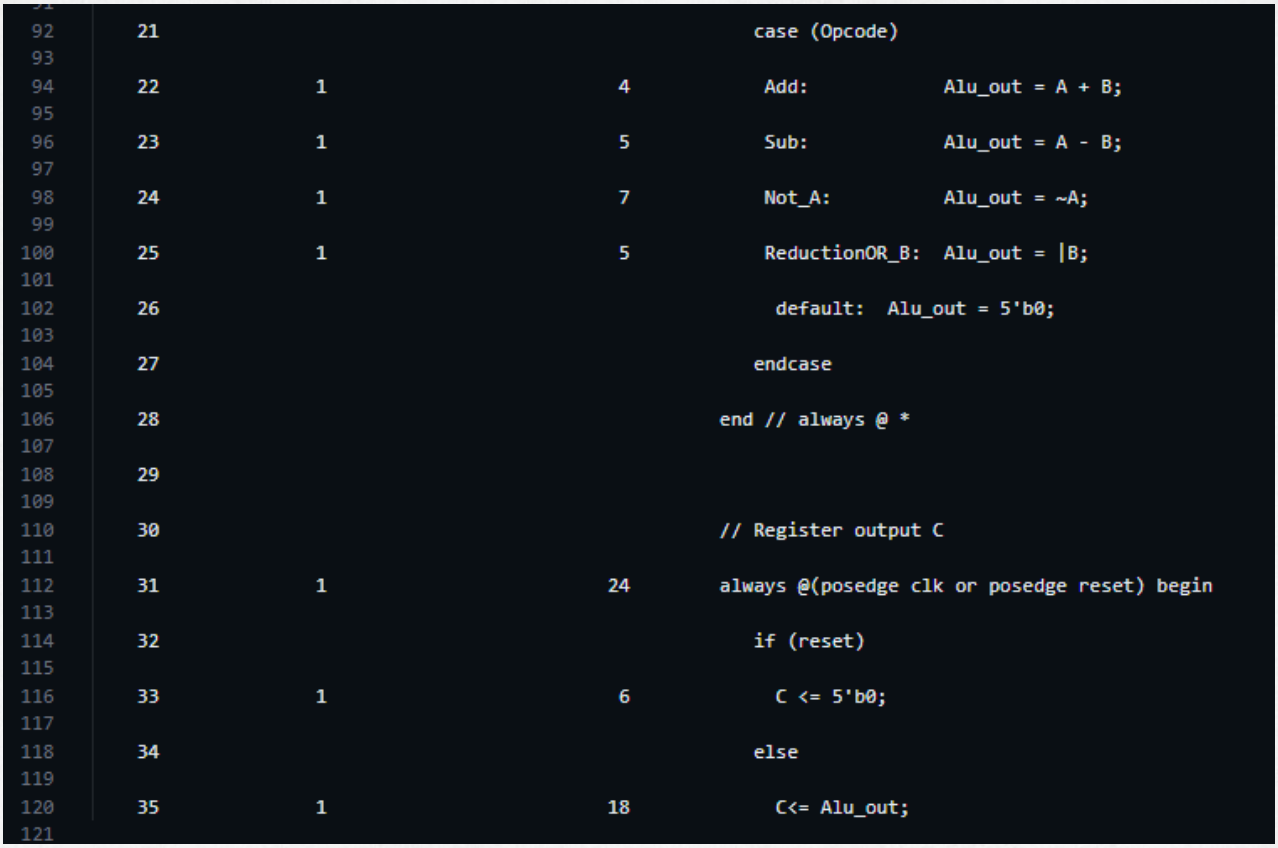



```
# *** ERROR count: 0, CORRECT count: 21
# ** Note: $stop    : 2_ALU_tb.sv(47)
#    Time: 42 ns  Iteration: 1  Instance: /ALU_tb
# Break in Module ALU_tb at 2_ALU_tb.sv line 47
```

# 7. Statement coverage report :

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                      8         8         0   100.00%

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

Statement Coverage for instance /\ALU_tb#dut  --

    Line         Item                    Count     Source
    ----         ----                    -----     ------
  File 2_ALU.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

    19                                             // Do the operation

    20             1                       21      always @* begin

    21                                                case (Opcode)
```

# 7. Statement coverage report :



| | | | | |
|---|---|---|---|---|
| 92 | 21 | | | case (Opcode) |
| 93 | | | | |
| 94 | 22 | 1 | 4 | Add: Alu_out = A + B; |
| 95 | | | | |
| 96 | 23 | 1 | 5 | Sub: Alu_out = A - B; |
| 97 | | | | |
| 98 | 24 | 1 | 7 | Not_A: Alu_out = ~A; |
| 99 | | | | |
| 100 | 25 | 1 | 5 | ReductionOR_B: Alu_out = \|B; |
| 101 | | | | |
| 102 | 26 | | | default: Alu_out = 5'b0; |
| 103 | | | | |
| 104 | 27 | | | endcase |
| 105 | | | | |
| 106 | 28 | | | end // always @ * |
| 107 | | | | |
| 108 | 29 | | | |
| 109 | | | | |
| 110 | 30 | | | // Register output C |
| 111 | | | | |
| 112 | 31 | 1 | 24 | always @(posedge clk or posedge reset) begin |
| 113 | | | | |
| 114 | 32 | | | if (reset) |
| 115 | | | | |
| 116 | 33 | 1 | 6 | C <= 5'b0; |
| 117 | | | | |
| 118 | 34 | | | else |
| 119 | | | | |
| 120 | 35 | 1 | 18 | C<= Alu_out; |
| 121 | | | | |

# 8. Branch coverage report :

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         6         6         0   100.00%

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

Branch Coverage for instance /\ALU_tb#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 2_ALU.v
------------------------------------CASE Branch------------------------------------
    21                                        21     Count coming in to CASE
    22             1                           4           Add:            Alu_out = A + B;

    23             1                           5           Sub:            Alu_out = A - B;

    24             1                           7           Not_A:          Alu_out = ~A;

    25             1                           5           ReductionOR_B:  Alu_out = |B;

Branch totals: 4 hits of 4 branches = 100.00%

-----------------------------------IF Branch------------------------------------
    32                                        24     Count coming in to IF
    32             1                           6          if (reset)

    34             1                          18          else

Branch totals: 2 hits of 2 branches = 100.00%
```

# 9. Toggle coverage report :

```
================================Toggle Details================================

Toggle Coverage for instance /\ALU_tb#dut  --

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

Total Node Count      =         22
Toggled Node Count    =         22
Untoggled Node Count  =          0

Toggle Coverage       =     100.00% (44 of 44 bins)


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

# Question 3:

1. RTL design:

```
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      always @(*) begin
21          case (cs)
22              IDLE:
23                  if (x)
24                      ns = IDLE;
25                  else
26                      ns = ZERO;
27              ZERO:
28                  if (x)
29                      ns = ONE;
30                  else
31                      ns = ZERO;
32              ONE:
33                  if (x)
34                      ns = IDLE;
35                  else
36                      ns = STORE;
37              STORE:
38                  if (x)
39                      ns = IDLE;
40                  else
41                      ns = ZERO;
42              default:    ns = IDLE;
43          endcase
44      end
```

```
45
46      always @(posedge clk or posedge rst) begin
47          if(rst) begin
48              cs <= IDLE;
49          end
50          else begin
51              cs <= ns;
52          end
53      end
54
55      always @(posedge clk or posedge rst) begin
56          if(rst) begin
57              users_count <= 0;
58          end
59          else begin
60              if (cs == STORE)
61                  users_count <= users_count + 1;
62          end
63      end
64
65      assign y = (cs == STORE)? 1:0;
66
67  endmodule
```

## 2. Verification plan:

| | Label | Design Requirement Description | Stimulus Generation | Functional Coverage | Functionality Check |
|---|---|---|---|---|---|
| 2 | RESET test | When the reset is asserted, outputs should be low. | Directed at the start and the end of the simulation, and called in radomization step. | - | **check_result()** task checks that output is 0 during reset. |
| 3 | RANDOMIZE test | Output of the dut design is equal to the output of the golden model | Randomized test using rand_stimuls class with constraints 90% deactive reset, | - | **check_result()** verifies correct output using golden model. |

3. packege code:

```
2   package FSM_pkg;
3       class fsm_transaction;
4           rand logic x;
5           rand logic rst;
6
7           constraint c1 {
8               rst dist {1 := 1, 0 := 90};
9               x dist   {0 := 67, 1:= (100 - 67)};
10          }
11      endclass
12  endpackage
```

# 4. Testbench code:

```
1   import FSM_pkg::*;
2
3   module FSM_010_tb();
4       logic clk;
5       logic rst;
6       logic x;
7       logic y;
8       logic [9:0]users_count;
9
10      logic y_exp;
11      logic [9:0]users_count_exp;
12
13      integer correct_count, error_count;
14
15      fsm_transaction my_input;
16
17      FSM_010 dut(.*);
18      FSM_010_golden golden(clk, rst, x, y_exp, users_count_exp);
19
20      initial begin
21          clk = 0;
22          forever
23              #1 clk = ~clk;
24      end
25
26      initial begin
27          correct_count = 0;
28          error_count = 0;
29          x = 0;
30
31          //reset_test
32          rst   = 1;
33          check_result();
34
35          //randomize test
36          my_input = new();
37          repeat(10000) begin
38              assert(my_input.randomize());
39              rst = my_input.rst;
40              x = my_input.x;
41              check_result();
42          end
```

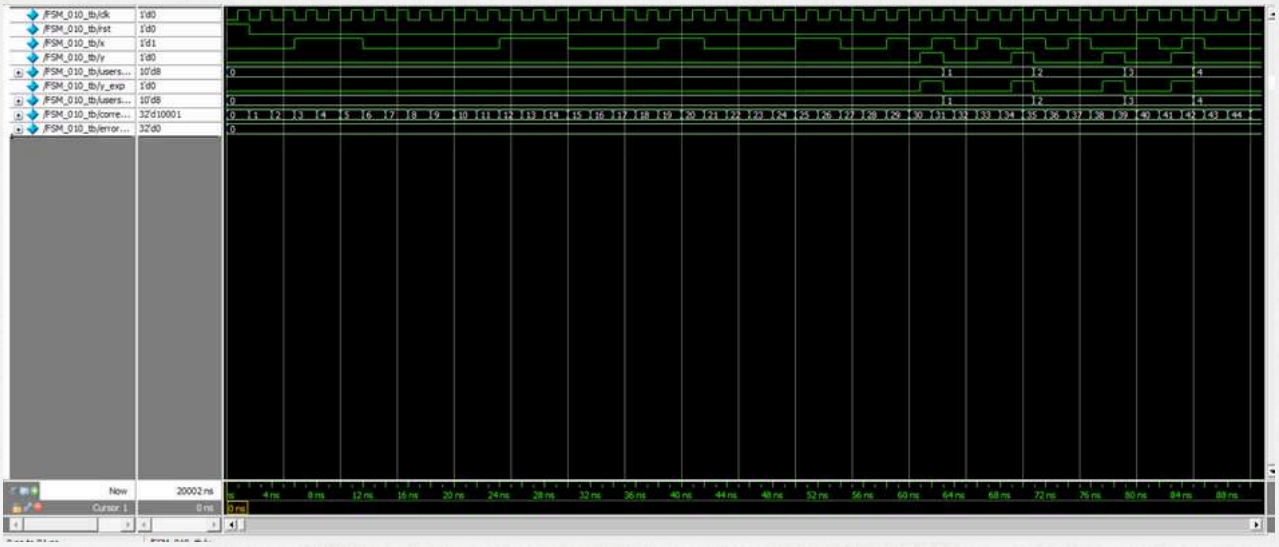
# 4. Testbench code:

```
43
44          $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
45          $stop;
46      end
47
48      task check_result();
49          @(negedge clk);
50          if(y != y_exp || users_count != users_count_exp) begin
51              $display("*** ERROR! at time %0t, y = %0d, Expected = %0d***",
52                  $time, y, y_exp);
53              error_count++;
54          end
55          else
56              correct_count++;
57
58      endtask
59
60  endmodule
```

# 5. Do file :

```
1   vlib work
2   vlog 3_golden_model.sv 3_FSM_pkg.sv 3_FSM_010.v 3_FSM_tb.sv +cover -covercells
3   vsim -voptargs=+acc work.FSM_010_tb -cover
4   add wave *
5   coverage save 3_FSM_tb.ucdb -onexit -du work.FSM_010
6   coverage exclude -du FSM_010 -togglenode {users_count[6]}
7   coverage exclude -du FSM_010 -togglenode {users_count[7]}
8   coverage exclude -du FSM_010 -togglenode {users_count[8]}
9   coverage exclude -du FSM_010 -togglenode {users_count[9]}
10
11  run -all
```

# 6. Qesta sim wave snippets (Unsigned):



```
# *** ERROR count: 0, CORRECT count: 10001
# ** Note: $stop    : 3_FSM_tb.sv(45)
#    Time: 20002 ns  Iteration: 1  Instance: /FSM_010_tb
# Break in Module FSM_010_tb at 3_FSM_tb.sv line 45
```

*since users_count is 10 bits, it is difficult to get 100% coverage for it, i need to make alot more loops.*

# 7. Statement coverage report :

```
Statement Coverage for instance /\FSM_010_tb#dut  --

  Line         Item                    Count     Source
  ----         ----                    -----     ------
File 3_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 [9:0] users_count;
  17
  18                                              reg [1:0] cs, ns;
  19
  20             1                    10245       always @(*) begin
  21                                                case (cs)
  22                                                  IDLE:
  23                                                    if (x)
  24             1                     1014               ns = IDLE;
  25                                                    else
  26             1                     1073               ns = ZERO;
  27                                                  ZERO:
```

# 7. Statement coverage report :

```
27                                      ZERO:
28                                        if (x)
29              1                1785       ns = ONE;
30                                        else
31              1                1821       ns = ZERO;
32                                      ONE:
33                                        if (x)
34              1                1767       ns = IDLE;
35                                        else
36              1                1195       ns = STORE;
37                                      STORE:
38                                        if (x)
39              1                 404       ns = IDLE;
40                                        else
41              1                1185       ns = ZERO;
42              1                   1     default:  ns = IDLE;
43                                    endcase
44                                  end
45
46              1                7505  always @(posedge clk or posedge rst) begin
47                                    if(rst) begin
```

# 7. Statement coverage report :

```
48              1                       200           cs <= IDLE;
49                                                  end
50                                                  else begin
51              1                      7305           cs <= ns;
52                                                  end
53                                                end
54
55              1                      5981      always @(posedge clk or posedge rst) begin
56                                                  if(rst) begin
57              1                       200           users_count <= 0;
58                                                  end
59                                                  else begin
60                                                    if (cs == STORE)
61              1                      1168             users_count <= users_count + 1;
62                                                  end
63                                                end
64
65              1                      5828      assign y = (cs == STORE)? 1:0;
```

# 8. Branch coverage report:

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        21        21         0   100.00%

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

Branch Coverage for instance /\FSM_010_tb#dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 3_FSM_010.v
-----------------------------------CASE Branch-----------------------------------
    21                                     10245     Count coming in to CASE
    22             1                        2087          IDLE:

    27             1                        3606          ZERO:

    32             1                        2962          ONE:

    37             1                        1589          STORE:

    42             1                           1          default:  ns = IDLE;

Branch totals: 5 hits of 5 branches = 100.00%

-------------------------------------IF Branch-------------------------------------
    23                                       2087     Count coming in to IF
    23             1                         1014           if (x)

    25             1                         1073           else

Branch totals: 2 hits of 2 branches = 100.00%

-------------------------------------IF Branch-------------------------------------
    28                                       3606     Count coming in to IF
    28             1                         1785           if (x)

    30             1                         1821           else

Branch totals: 2 hits of 2 branches = 100.00%
```

# 8. Branch coverage report:

```
49  ------------------------------------IF Branch------------------------------------
50      33                                      2962     Count coming in to IF
51      33              1                       1767             if (x)
52
53      35              1                       1195             else
54
55  Branch totals: 2 hits of 2 branches = 100.00%
56
57  ------------------------------------IF Branch------------------------------------
58      38                                      1589     Count coming in to IF
59      38              1                        404             if (x)
60
61      40              1                       1185             else
62
63  Branch totals: 2 hits of 2 branches = 100.00%
64
65  ------------------------------------IF Branch------------------------------------
66      47                                      7505     Count coming in to IF
67      47              1                        200        if(rst) begin
68
69      50              1                       7305        else begin
70
71  Branch totals: 2 hits of 2 branches = 100.00%
72
73  ------------------------------------IF Branch------------------------------------
74      56                                      5981     Count coming in to IF
75      56              1                        200        if(rst) begin
76
77      59              1                       5781        else begin
78
79  Branch totals: 2 hits of 2 branches = 100.00%
80
81  ------------------------------------IF Branch------------------------------------
82      60                                      5781     Count coming in to IF
83      60              1                       1168          if (cs == STORE)
84
85                                              4613     All False Count
86  Branch totals: 2 hits of 2 branches = 100.00%
```

```
88  ------------------------------------IF Branch------------------------------------
89      65                                      5827     Count coming in to IF
90      65              1                       1185      assign y = (cs == STORE)? 1:0;
91
92      65              2                       4642      assign y = (cs == STORE)? 1:0;
93
94  Branch totals: 2 hits of 2 branches = 100.00%
```

# 9. FSM coverage:

```
================================FSM Details================================

FSM Coverage for instance /\FSM_010_tb#dut  --

FSM_ID: cs
    Current State Object : cs
    ---------------------
    State Value MapInfo :
    --------------------
Line          State Name               Value
----          ----------               -----
  22                IDLE                   0
  27                ZERO                   1
  32                 ONE                   2
  37               STORE                   3
    Covered States :
    ----------------
                    State          Hit_count
                    -----          ---------
                     IDLE               1517
                     ZERO               3036
                      ONE               1767
                    STORE               1185
    Covered Transitions :
    ---------------------
Line            Trans_ID          Hit_count          Transition
----            --------          ---------          ----------
  26                   0               1053          IDLE -> ZERO
  29                   1               1767          ZERO -> ONE
  48                   2                 54          ZERO -> IDLE
  36                   3               1185          ONE -> STORE
  34                   4                582          ONE -> IDLE
  41                   5                768          STORE -> ZERO
  39                   6                417          STORE -> IDLE


    Summary                     Bins      Hits    Misses  Coverage
    -------                     ----      ----    ------  --------
        FSM States                 4         4         0   100.00%
        FSM Transitions            7         7         0   100.00%
```

# 10. Toggle coverage:

```
309  Toggle Coverage:
310      Enabled Coverage              Bins      Hits    Misses  Coverage
311      ----------------              ----      ----    ------  --------
312      Toggles                         28        28         0   100.00%
313
314  ================================Toggle Details================================
315
316  Toggle Coverage for instance /\FSM_010_tb#dut  --
317
318                                           Node      1H->0L      0L->1H  "Coverage"
319                                           ----------------------------------------
320                                            clk           1           1      100.00
321                                          cs[1-0]         1           1      100.00
322                                          ns[1-0]         1           1      100.00
323                                            rst           1           1      100.00
324                                     users_count[5-0]     1           1      100.00
325                                              x           1           1      100.00
326                                              y           1           1      100.00
327
328  Total Node Count        =         14
329  Toggled Node Count      =         14
330  Untoggled Node Count    =          0
331
332  Toggle Coverage         =     100.00% (28 of 28 bins)
333
334
335  Total Coverage By Instance (filtered view): 100.00%
```