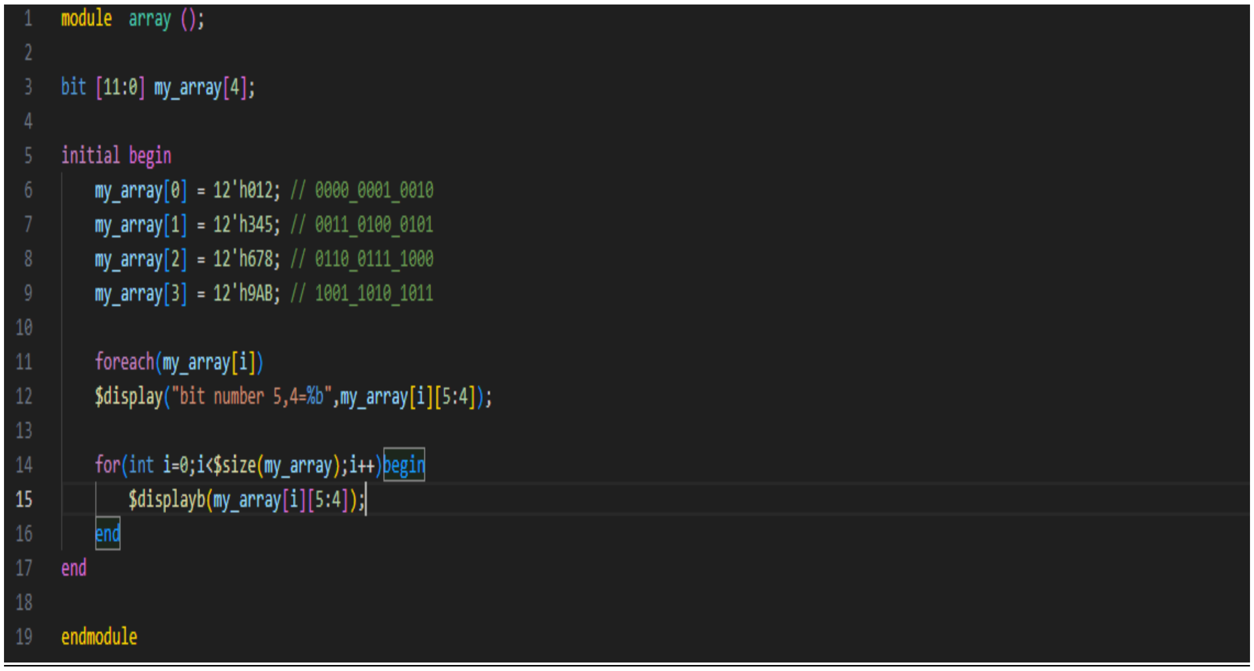
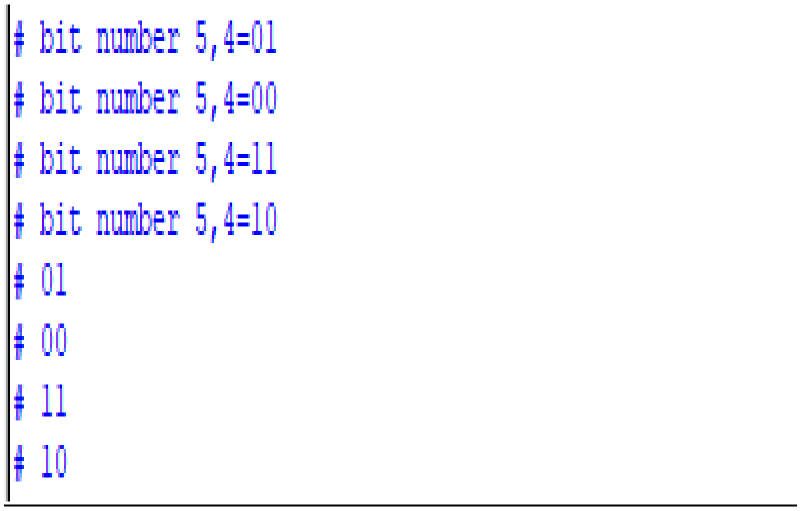
Assigment 2 _extra

Array



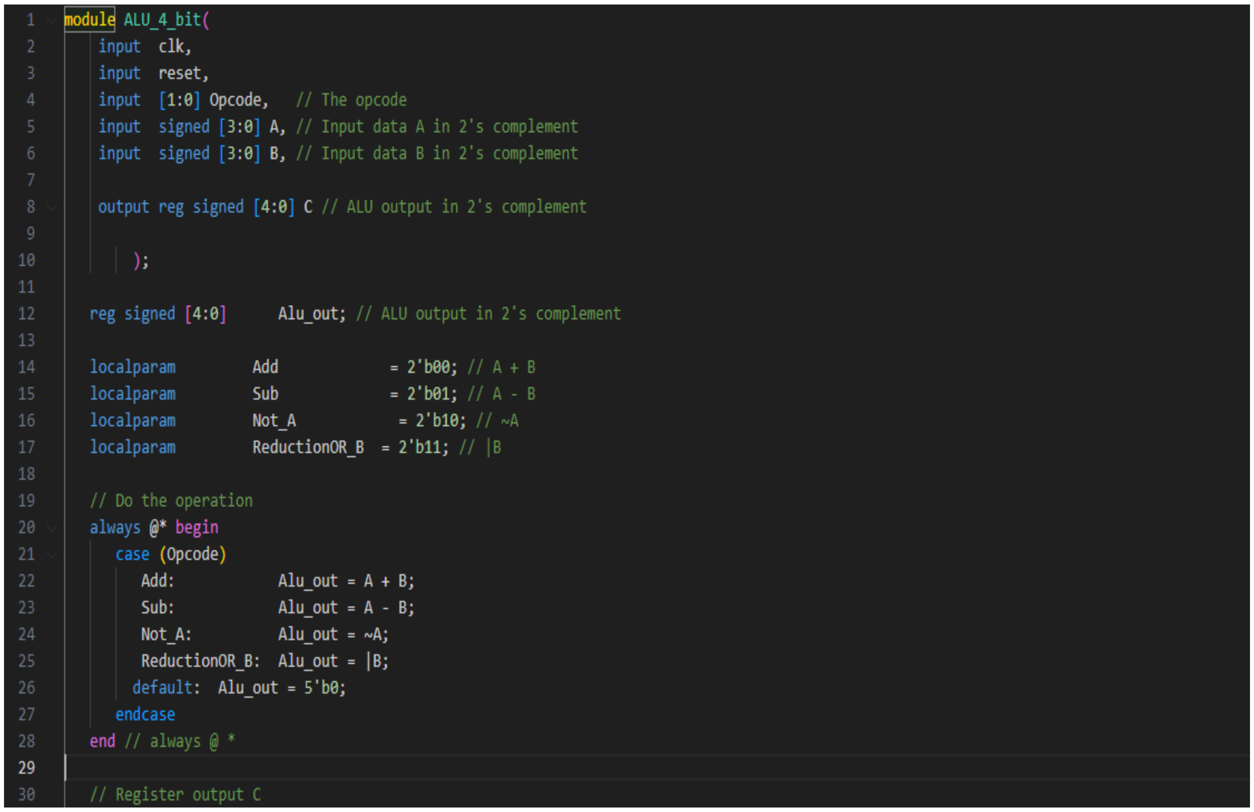
```
module  array ();

bit [11:0] my_array[4];

initial begin
    my_array[0] = 12'h012; // 0000_0001_0010
    my_array[1] = 12'h345; // 0011_0100_0101
    my_array[2] = 12'h678; // 0110_0111_1000
    my_array[3] = 12'h9AB; // 1001_1010_1011

    foreach(my_array[i])
    $display("bit number 5,4=%b",my_array[i][5:4]);

    for(int i=0;i<$size(my_array);i++)begin
        $displayb(my_array[i][5:4]);
    end
end

endmodule
```



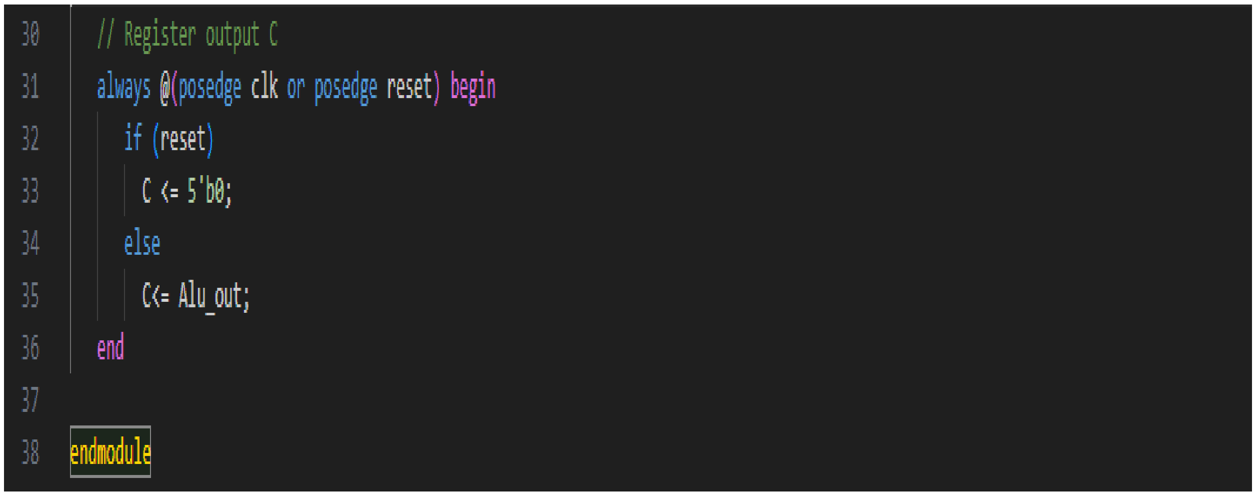
```
# bit number 5,4=01
# bit number 5,4=00
# bit number 5,4=11
# bit number 5,4=10
# 01
# 00
# 11
# 10
```

ALU project

Design

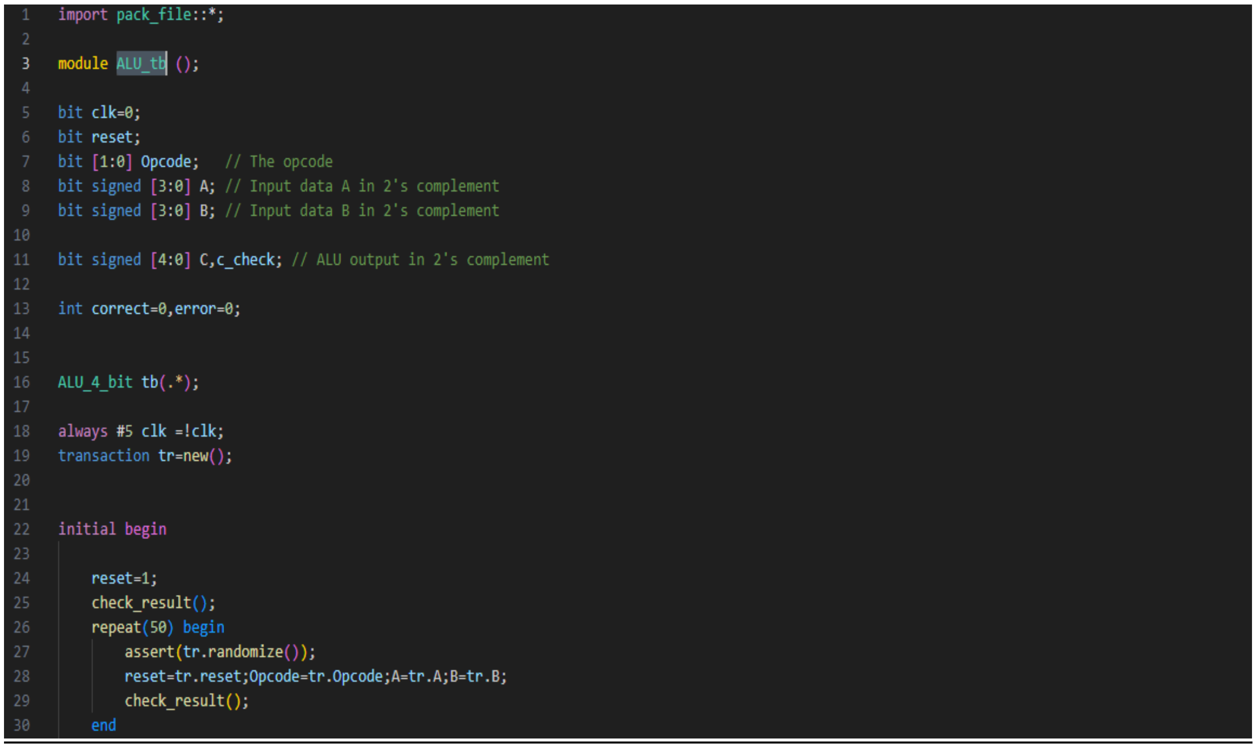


```
module ALU_4_bit(
    input  clk,
    input  reset,
    input  [1:0] Opcode,   // The opcode
    input  signed [3:0] A, // Input data A in 2's complement
    input  signed [3:0] B, // Input data B in 2's complement

    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
```

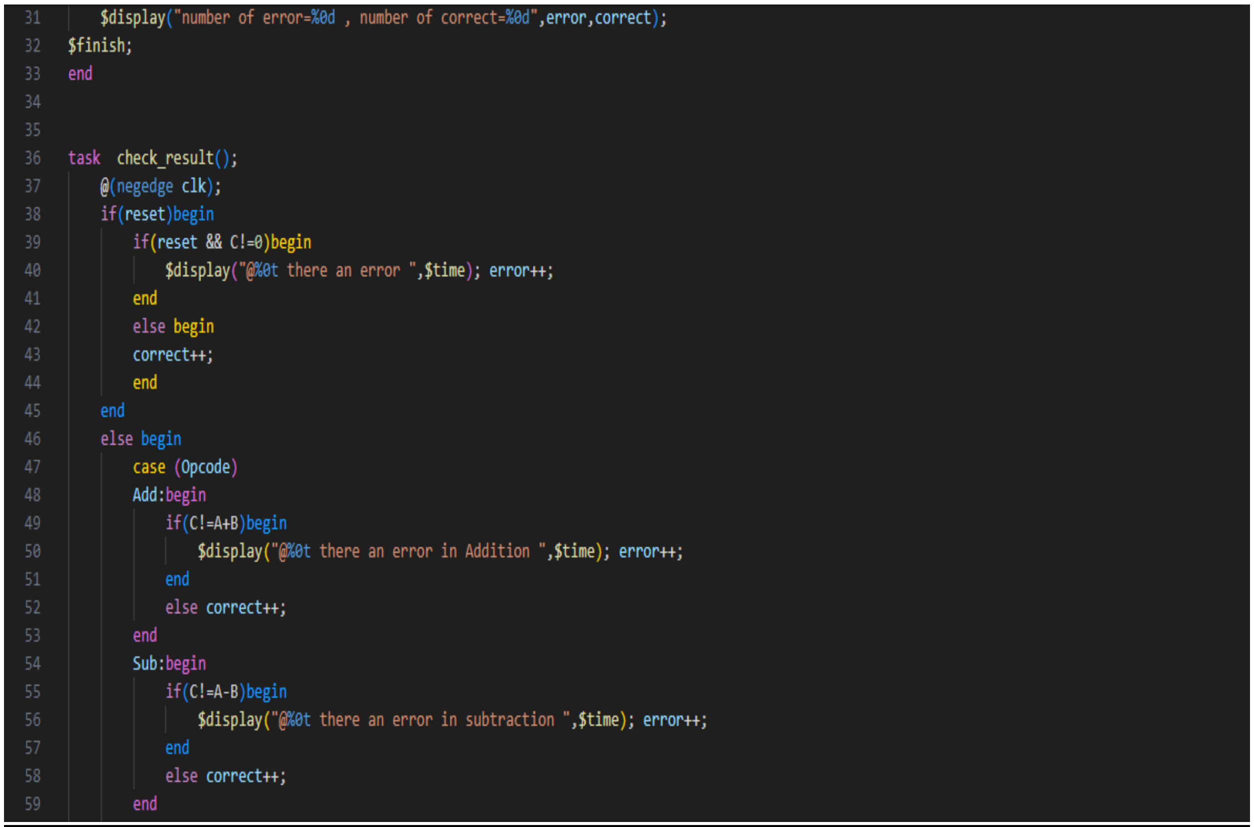


```
   // Register output C
   always @(posedge clk or posedge reset) begin
     if (reset)
       C <= 5'b0;
     else
       C<= Alu_out;
   end

endmodule
```

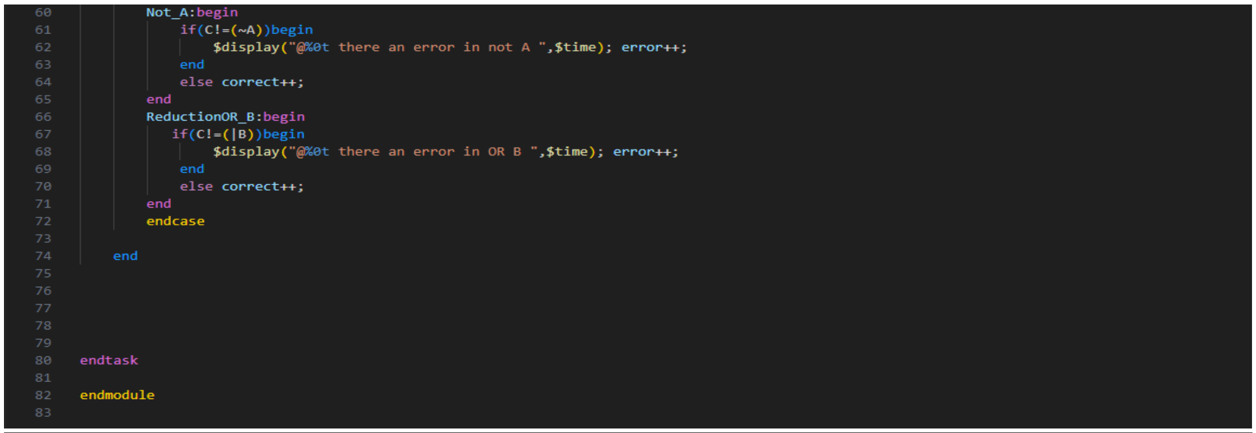
Test Bench



```
import pack_file::*;

module ALU_tb ();

bit clk=0;
bit reset;
bit [1:0] Opcode;   // The opcode
bit signed [3:0] A; // Input data A in 2's complement
bit signed [3:0] B; // Input data B in 2's complement

bit signed [4:0] C,c_check; // ALU output in 2's complement

int correct=0,error=0;


ALU_4_bit tb(.*);

always #5 clk =!clk;
transaction tr=new();


initial begin

    reset=1;
    check_result();
    repeat(50) begin
        assert(tr.randomize());
        reset=tr.reset;Opcode=tr.Opcode;A=tr.A;B=tr.B;
        check_result();
    end
```

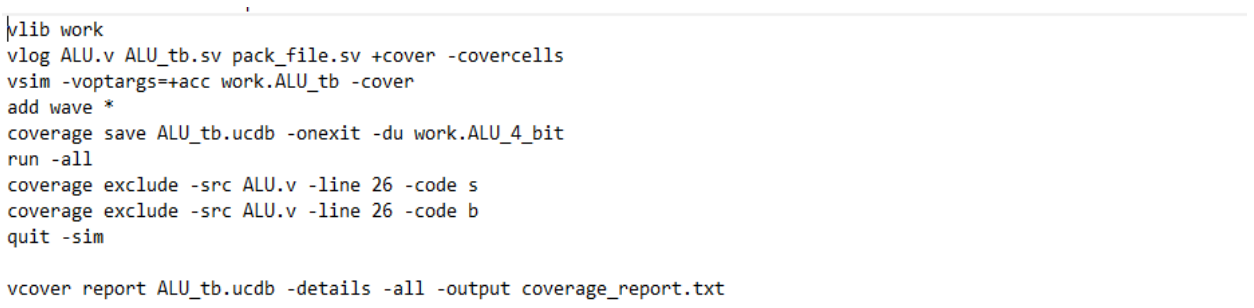


```
    $display("number of error=%0d , number of correct=%0d",error,correct);
$finish;
end


task  check_result();
    @(negedge clk);
    if(reset)begin
        if(reset && C!=0)begin
            $display("@%0t there an error ",$time); error++;
        end
        else begin
        correct++;
        end
    end
    else begin
        case (Opcode)
        Add:begin
            if(C!=A+B)begin
                $display("@%0t there an error in Addition ",$time); error++;
            end
            else correct++;
        end
        Sub:begin
            if(C!=A-B)begin
                $display("@%0t there an error in subtraction ",$time); error++;
            end
            else correct++;
        end
```



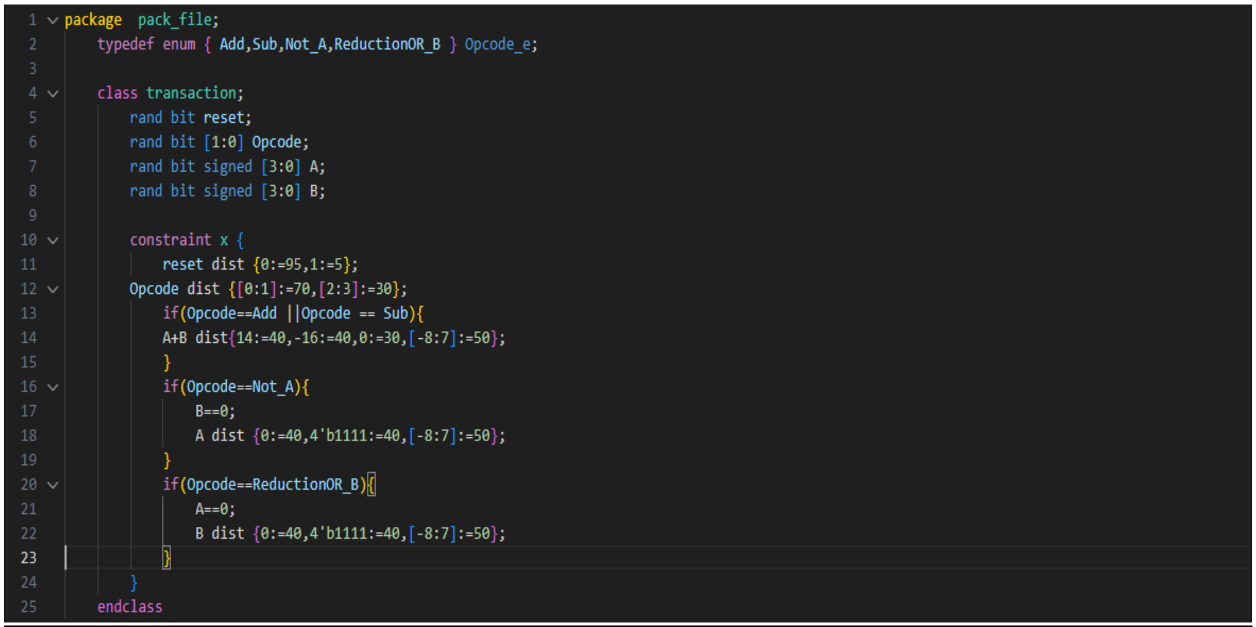
```
            Not_A:begin
                if(C!=(~A))begin
                    $display("@%0t there an error in not A ",$time); error++;
                end
                else correct++;
            end
            ReductionOR_B:begin
               if(C!=(|B))begin
                    $display("@%0t there an error in OR B ",$time); error++;
                end
                else correct++;
            end
            endcase

        end




endtask

endmodule
```

Do file



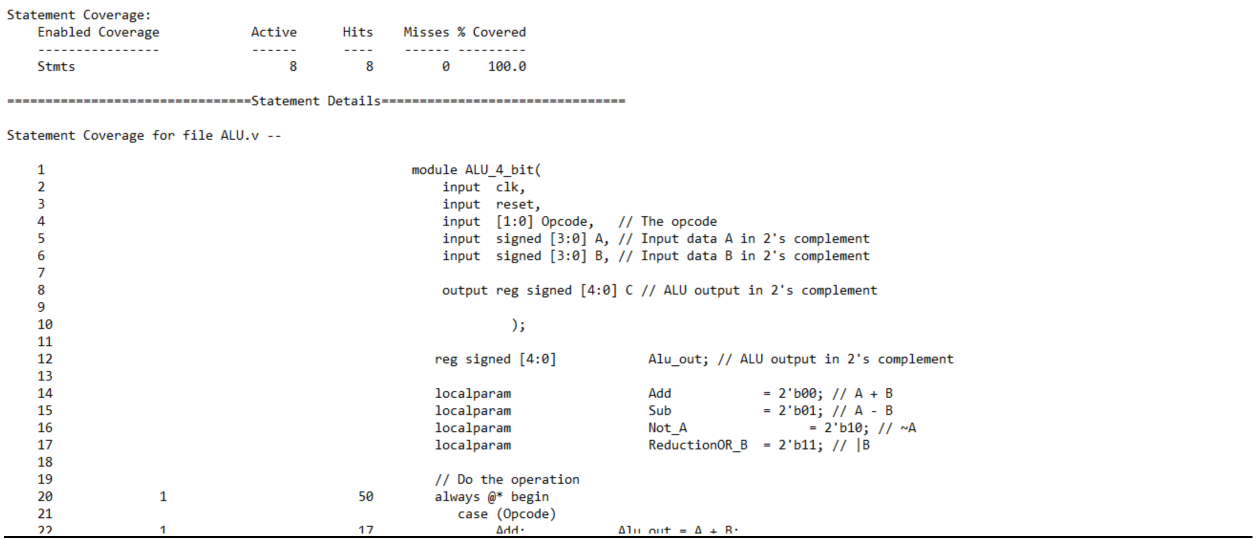
```
vlib work
vlog ALU.v ALU_tb.sv pack_file.sv +cover -covercells
vsim -voptargs=+acc work.ALU_tb -cover
add wave *
coverage save ALU_tb.ucdb -onexit -du work.ALU_4_bit
run -all
coverage exclude -src ALU.v -line 26 -code s
coverage exclude -src ALU.v -line 26 -code b
quit -sim

vcover report ALU_tb.ucdb -details -all -output coverage_report.txt
```

Package

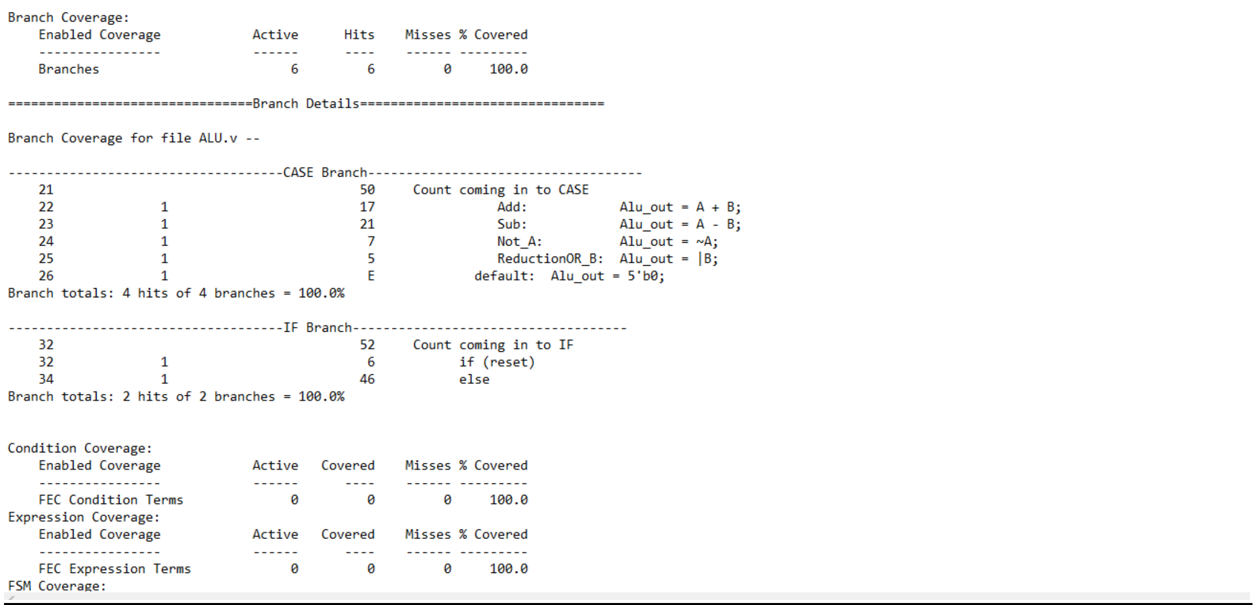


```
package  pack_file;
    typedef enum { Add,Sub,Not_A,ReductionOR_B } Opcode_e;

    class transaction;
        rand bit reset;
        rand bit [1:0] Opcode;
        rand bit signed [3:0] A;
        rand bit signed [3:0] B;

        constraint x {
            reset dist {0:=95,1:=5};
        Opcode dist {[0:1]:=70,[2:3]:=30};
            if(Opcode==Add ||Opcode == Sub){
            A+B dist{14:=40,-16:=40,0:=30,[-8:7]:=50};
            }
            if(Opcode==Not_A){
                B==0;
                A dist {0:=40,4'b1111:=40,[-8:7]:=50};
            }
            if(Opcode==ReductionOR_B){
                A==0;
                B dist {0:=40,4'b1111:=40,[-8:7]:=50};
            }
        }
    endclass
```

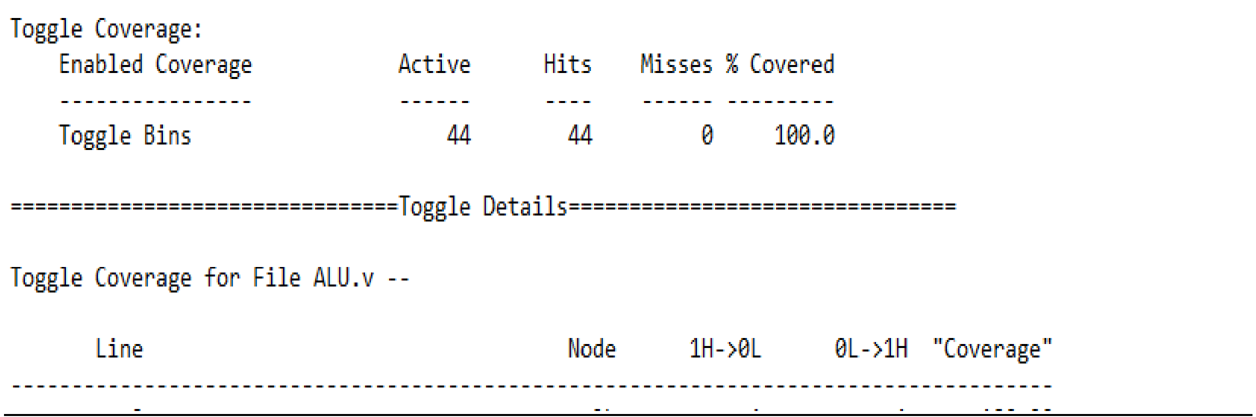
Coverage report



```
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         0     100.0

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

Statement Coverage for file 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                         50           always @* begin
    21                                                         case (Opcode)
    22               1                         17                   Add:           Alu_out = A + B;
```

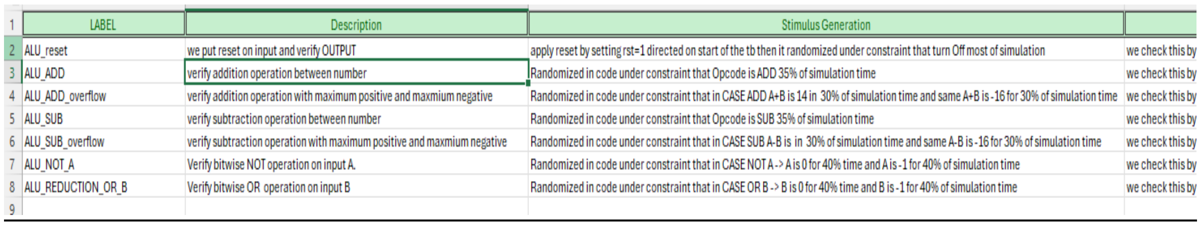


```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         6         6         0     100.0

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

Branch Coverage for file ALU.v --

------------------------------------CASE Branch------------------------------------
    21                                         50       Count coming in to CASE
    22               1                         17                 Add:           Alu_out = A + B;
    23               1                         21                 Sub:           Alu_out = A - B;
    24               1                          7                 Not_A:         Alu_out = ~A;
    25               1                          5                 ReductionOR_B:  Alu_out = |B;
    26               1                          E              default:  Alu_out = 5'b0;
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    32                                         52       Count coming in to IF
    32               1                          6             if (reset)
    34               1                         46             else
Branch totals: 2 hits of 2 branches = 100.0%


Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Condition Terms              0         0         0     100.0
Expression Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Expression Terms             0         0         0     100.0
FSM Coverage:
```

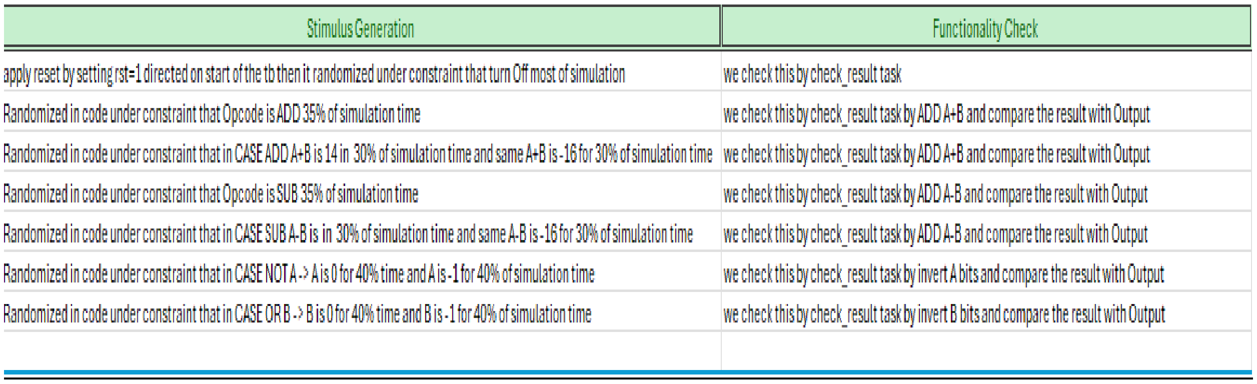


```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       44        44         0     100.0

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

Toggle Coverage for File ALU.v --

       Line                                   Node       1H->0L       0L->1H  "Coverage"
--------------------------------------------------------------------------------------
```

Verification plan

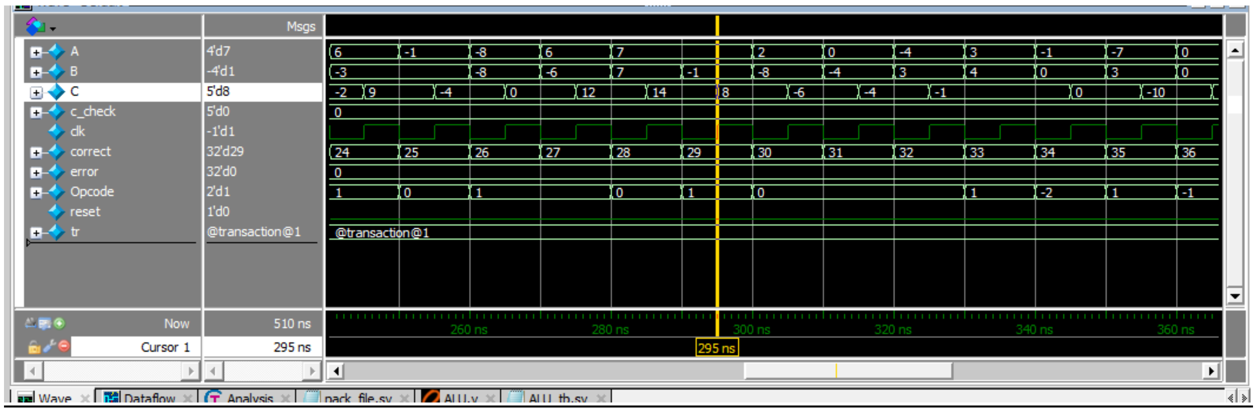


	LABEL	Description	Stimulus Generation	
2	ALU_reset	we put reset on input and verify OUTPUT	apply reset by setting rst=1 directed on start of the tb then it randomized under constraint that turn Off most of simulation	we check this by
3	ALU_ADD	verify addition operation between number	Randomized in code under constraint that Opcode is ADD 35% of simulation time	we check this by
4	ALU_ADD_overflow	verify addition operation with maximum positive and maxmium negative	Randomized in code under constraint that in CASE ADD A+B is 14 in 30% of simulation time and same A+B is -16 for 30% of simulation time	we check this by
5	ALU_SUB	verify subtraction operation between number	Randomized in code under constraint that Opcode is SUB 35% of simulation time	we check this by
6	ALU_SUB_overflow	verify subtraction operation with maximum positive and maxmium negative	Randomized in code under constraint that in CASE SUB A-B is in 30% of simulation time and same A-B is -16 for 30% of simulation time	we check this by
7	ALU_NOT_A	Verify bitwise NOT operation on input A.	Randomized in code under constraint that in CASE NOT A -> A is 0 for 40% time and A is -1 for 40% of simulation time	we check this by
8	ALU_REDUCTION_OR_B	Verify bitwise OR operation on input B	Randomized in code under constraint that in CASE OR B -> B is 0 for 40% time and B is -1 for 40% of simulation time	we check this by
9				

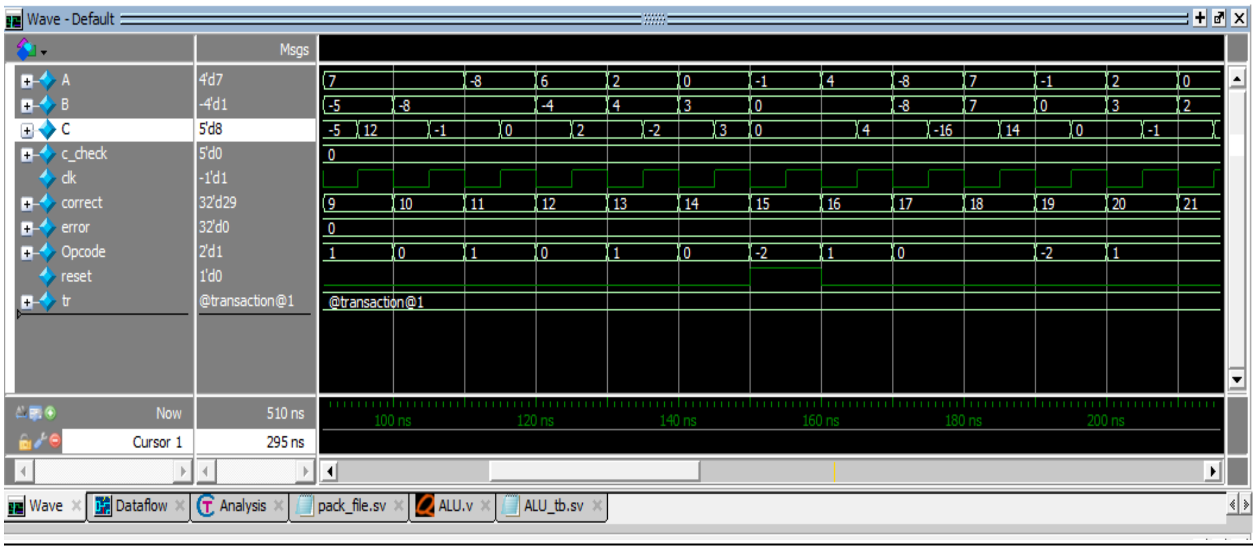


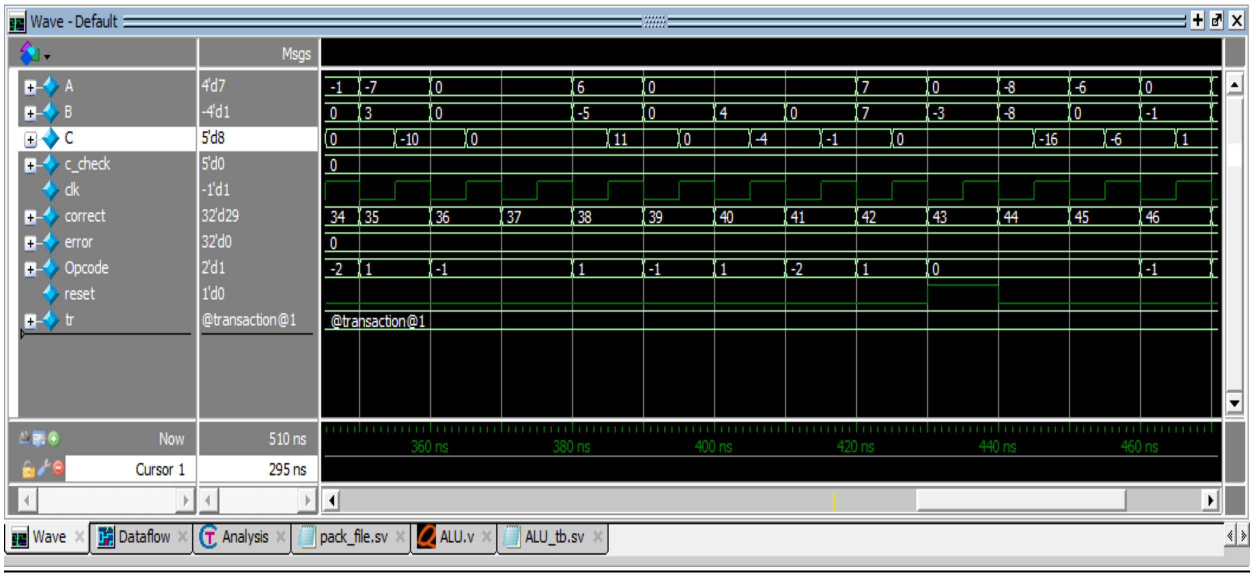
Stimulus Generation	Functionality Check
apply reset by setting rst=1 directed on start of the tb then it randomized under constraint that turn Off most of simulation	we check this by check_result task
Randomized in code under constraint that Opcode is ADD 35% of simulation time	we check this by check_result task by ADD A+B and compare the result with Output
Randomized in code under constraint that in CASE ADD A+B is 14 in 30% of simulation time and same A+B is -16 for 30% of simulation time	we check this by check_result task by ADD A+B and compare the result with Output
Randomized in code under constraint that Opcode is SUB 35% of simulation time	we check this by check_result task by ADD A-B and compare the result with Output
Randomized in code under constraint that in CASE SUB A-B is in 30% of simulation time and same A-B is -16 for 30% of simulation time	we check this by check_result task by ADD A-B and compare the result with Output
Randomized in code under constraint that in CASE NOT A -> A is 0 for 40% time and A is -1 for 40% of simulation time	we check this by check_result task by invert A bits and compare the result with Output
Randomized in code under constraint that in CASE OR B -> B is 0 for 40% time and B is -1 for 40% of simulation time	we check this by check_result task by invert B bits and compare the result with Output

simulation



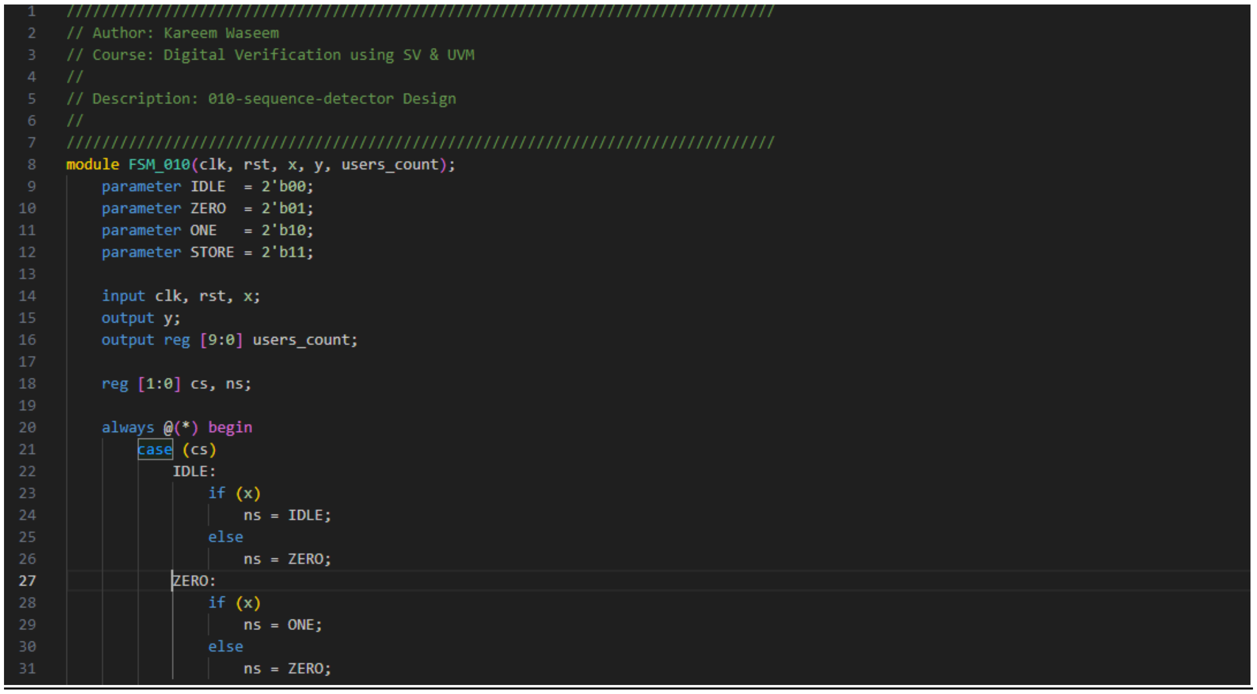
	Msgs
A	4'd7
B	-4'd1
C	5'd8
c_check	5'd0
clk	-1'd1
correct	32'd29
error	32'd0
Opcode	2'd1
reset	1'd0
tr	@transaction@1
Now	510 ns
Cursor 1	295 ns


Wave - Default
Msgs
A 4'd7
B -4'd1
C 5'd8
c_check 5'd0
clk -1'd1
correct 32'd29
error 32'd0
Opcode 2'd1
reset 1'd0
tr @transaction@1
Now 510 ns
Cursor 1 295 ns
100 ns
120 ns
140 ns
160 ns
180 ns
200 ns
Wave
Dataflow
Analysis
pack_file.sv
ALU.v
ALU_tb.sv

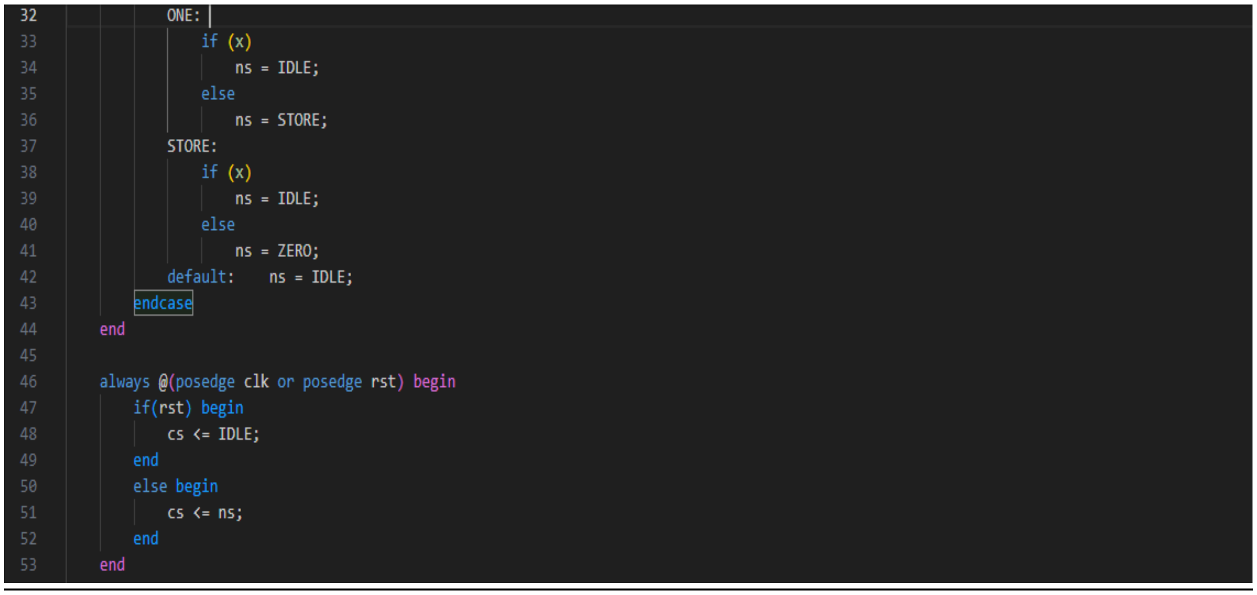

Wave - Default
Msgs
A 4'd7
B -4'd1
C 5'd8
c_check 5'd0
clk -1'd1
correct 32'd29
error 32'd0
Opcode 2'd1
reset 1'd0
tr @transaction@1
Now 510 ns
Cursor 1 295 ns
360 ns
380 ns
400 ns
420 ns
440 ns
460 ns
Wave
Dataflow
Analysis
pack_file.sv
ALU.v
ALU_tb.sv

FSM project

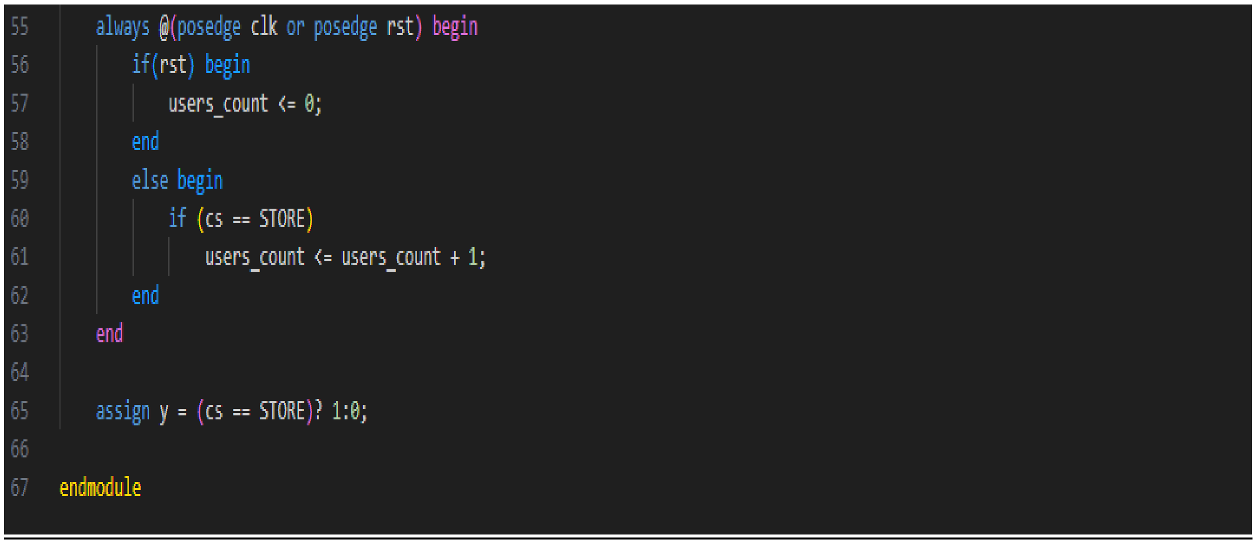
Design



```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: 010-sequence-detector Design
//
////////////////////////////////////////////////////////////////////////////////
module FSM_010(clk, rst, x, y, users_count);
    parameter IDLE  = 2'b00;
    parameter ZERO  = 2'b01;
    parameter ONE   = 2'b10;
    parameter STORE = 2'b11;

    input clk, rst, x;
    output y;
    output reg [9:0] users_count;

    reg [1:0] cs, ns;

    always @(*) begin
        case (cs)
            IDLE:
                if (x)
                    ns = IDLE;
                else
                    ns = ZERO;
            ZERO:
                if (x)
                    ns = ONE;
                else
                    ns = ZERO;
```

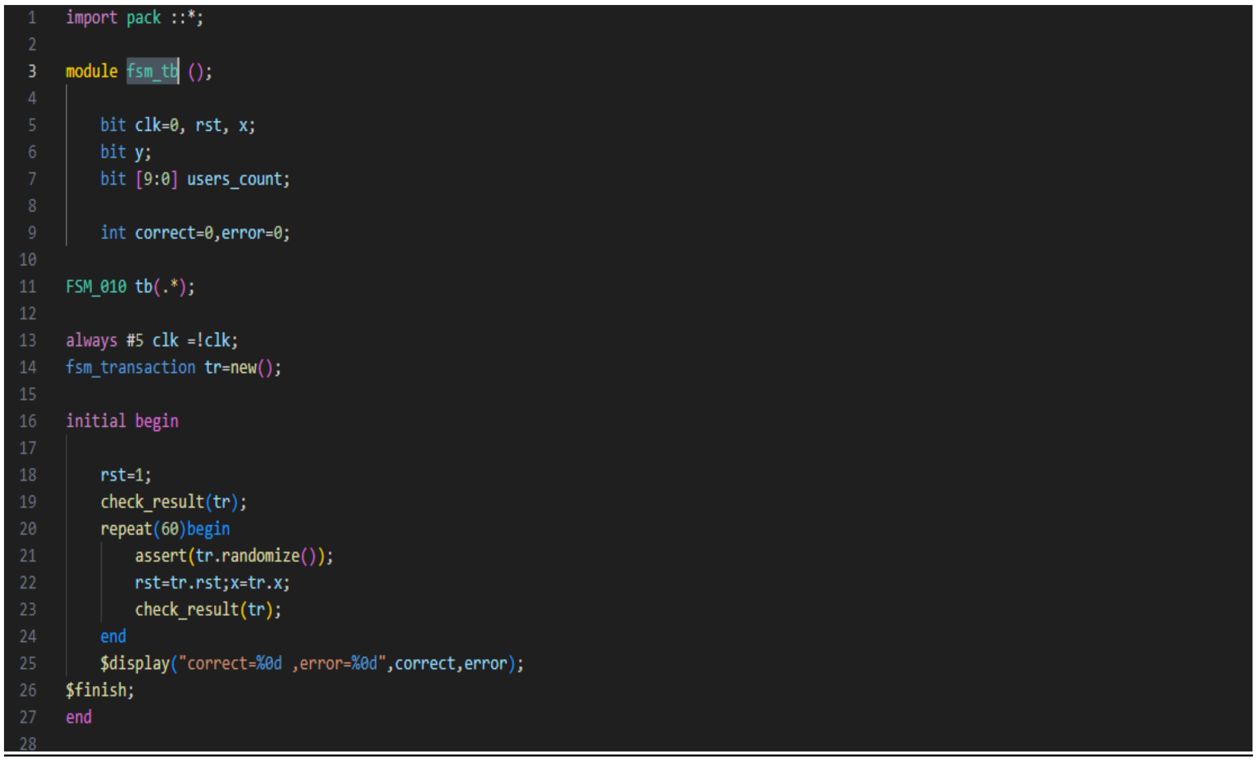


```
            ONE:
                if (x)
                    ns = IDLE;
                else
                    ns = STORE;
            STORE:
                if (x)
                    ns = IDLE;
                else
                    ns = ZERO;
            default:    ns = IDLE;
        endcase
    end

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            cs <= IDLE;
        end
        else begin
            cs <= ns;
        end
    end
```

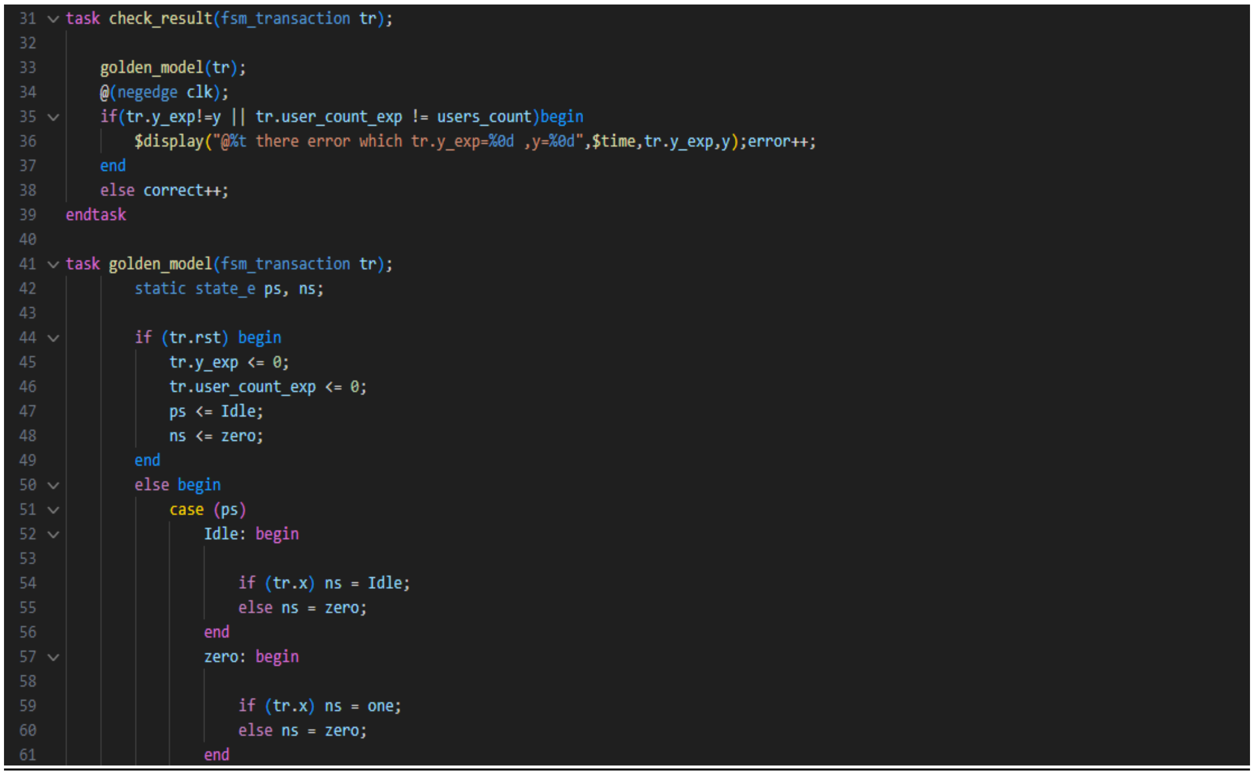


```
    always @(posedge clk or posedge rst) begin
        if(rst) begin
            users_count <= 0;
        end
        else begin
            if (cs == STORE)
                users_count <= users_count + 1;
        end
    end

    assign y = (cs == STORE)? 1:0;

endmodule
```

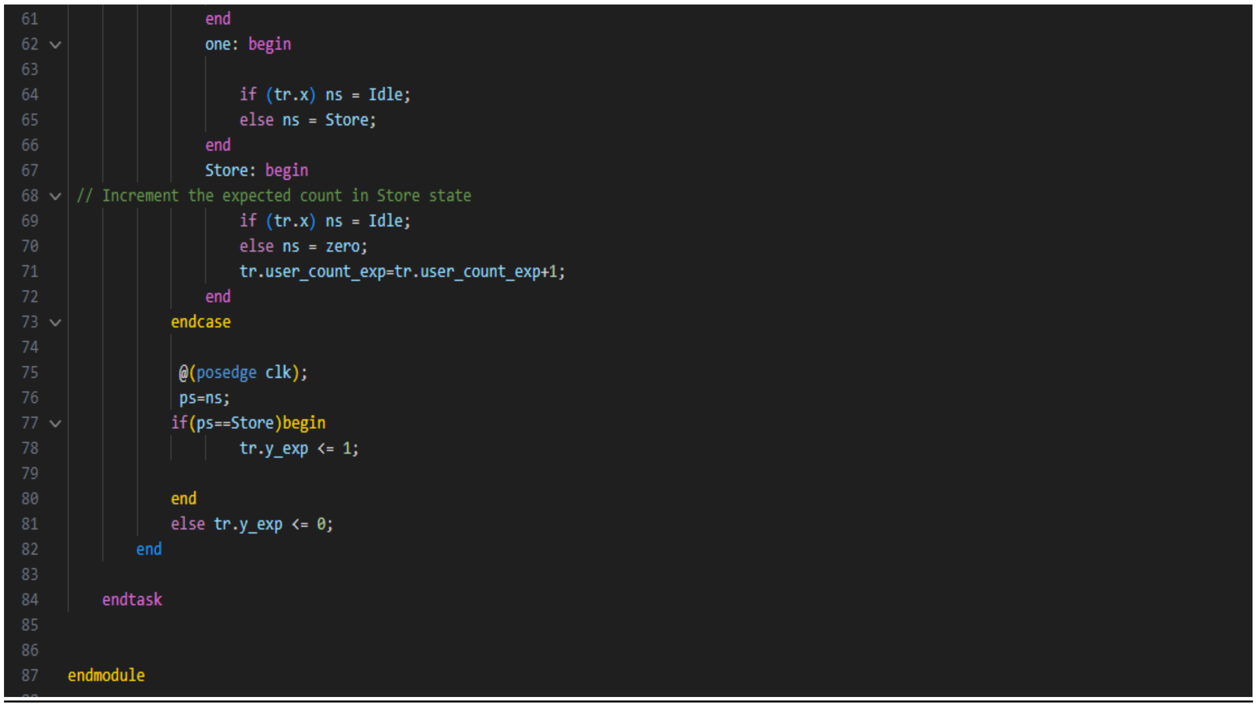
Test bench



```
import pack ::*;

module fsm_tb ();

    bit clk=0, rst, x;
    bit y;
    bit [9:0] users_count;

    int correct=0,error=0;

FSM_010 tb(.*);

always #5 clk =!clk;
fsm_transaction tr=new();

initial begin

    rst=1;
    check_result(tr);
    repeat(60)begin
        assert(tr.randomize());
        rst=tr.rst;x=tr.x;
        check_result(tr);
    end
    $display("correct=%0d ,error=%0d",correct,error);
$finish;
end

```

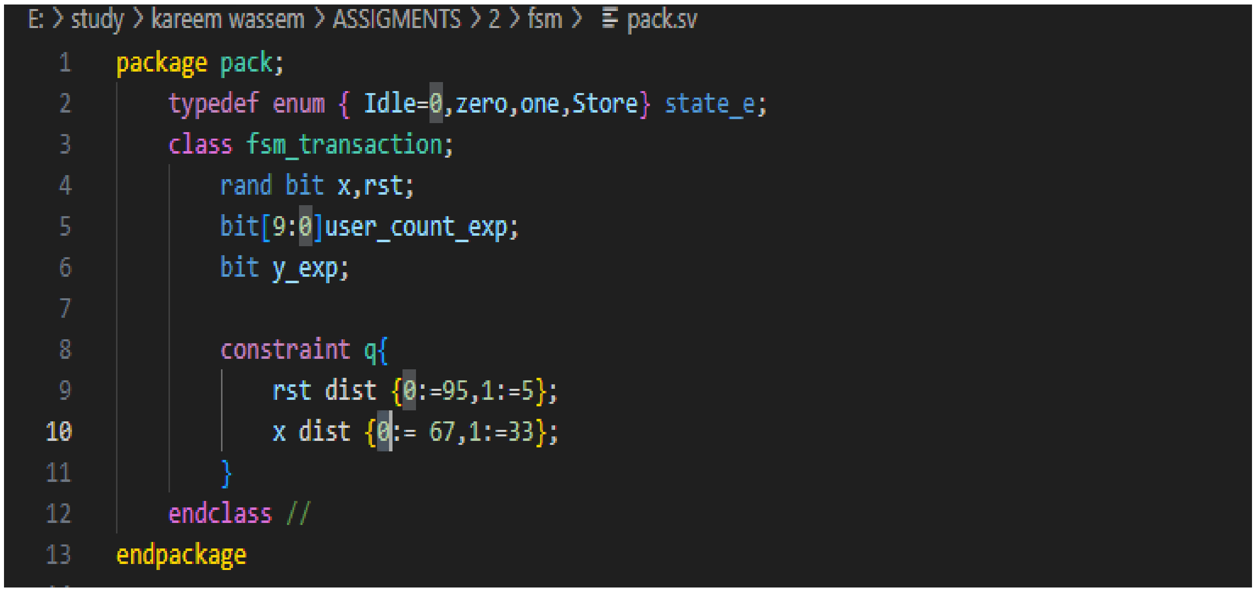


```
task check_result(fsm_transaction tr);

    golden_model(tr);
    @(negedge clk);
    if(tr.y_exp!=y || tr.user_count_exp != users_count)begin
        $display("@%t there error which tr.y_exp=%0d ,y=%0d",$time,tr.y_exp,y);error++;
    end
    else correct++;
endtask

task golden_model(fsm_transaction tr);
        static state_e ps, ns;

        if (tr.rst) begin
            tr.y_exp <= 0;
            tr.user_count_exp <= 0;
            ps <= Idle;
            ns <= zero;
        end
        else begin
            case (ps)
                Idle: begin

                    if (tr.x) ns = Idle;
                    else ns = zero;
                end
                zero: begin

                    if (tr.x) ns = one;
                    else ns = zero;
                end
```



```
                end
                one: begin

                    if (tr.x) ns = Idle;
                    else ns = Store;
                end
                Store: begin
 // Increment the expected count in Store state
                    if (tr.x) ns = Idle;
                    else ns = zero;
                    tr.user_count_exp=tr.user_count_exp+1;
                end
            endcase

             @(posedge clk);
             ps=ns;
            if(ps==Store)begin
                    tr.y_exp <= 1;

            end
            else tr.y_exp <= 0;
        end

    endtask


endmodule
```

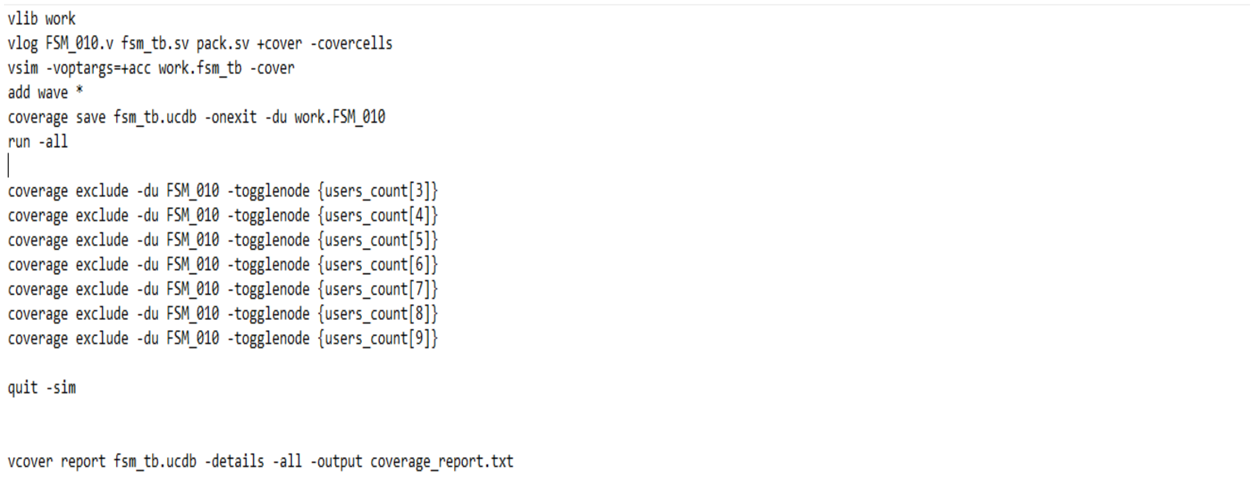
Package



E: > study > kareem wassem > ASSIGMENTS > 2 > fsm > pack.sv

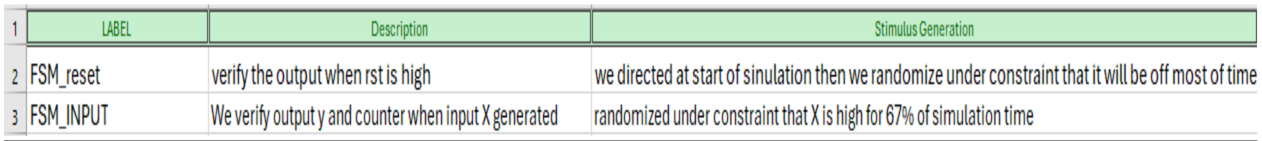
```
package pack;
    typedef enum { Idle=0,zero,one,Store} state_e;
    class fsm_transaction;
        rand bit x,rst;
        bit[9:0]user_count_exp;
        bit y_exp;

        constraint q{
            rst dist {0:=95,1:=5};
            x dist {0:= 67,1:=33};
        }
    endclass //
endpackage
```

Do_file

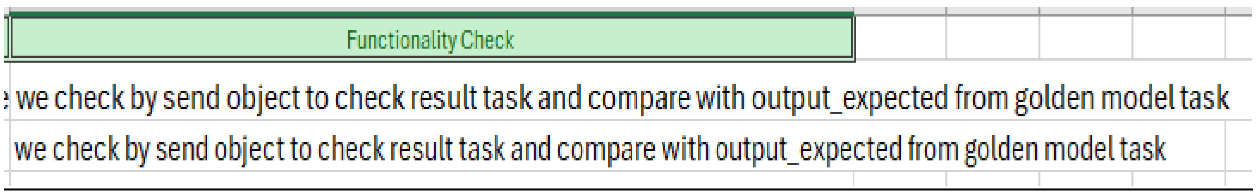


```
vlib work
vlog FSM_010.v fsm_tb.sv pack.sv +cover -covercells
vsim -voptargs=+acc work.fsm_tb -cover
add wave *
coverage save fsm_tb.ucdb -onexit -du work.FSM_010
run -all

coverage exclude -du FSM_010 -togglenode {users_count[3]}
coverage exclude -du FSM_010 -togglenode {users_count[4]}
coverage exclude -du FSM_010 -togglenode {users_count[5]}
coverage exclude -du FSM_010 -togglenode {users_count[6]}
coverage exclude -du FSM_010 -togglenode {users_count[7]}
coverage exclude -du FSM_010 -togglenode {users_count[8]}
coverage exclude -du FSM_010 -togglenode {users_count[9]}

quit -sim


vcover report fsm_tb.ucdb -details -all -output coverage_report.txt
```

verification plan



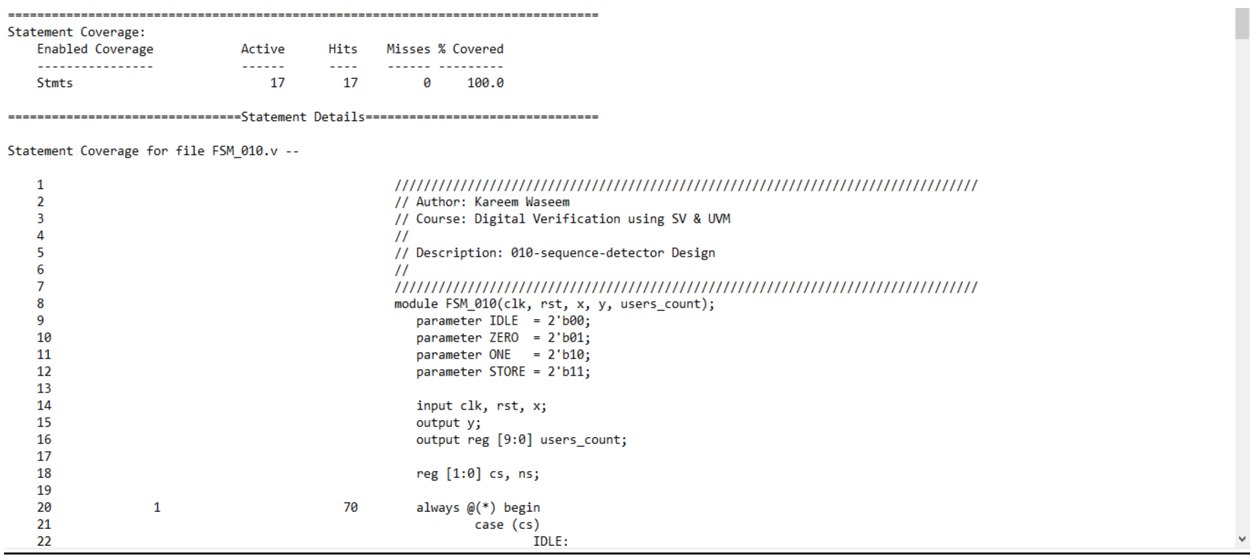
LABEL	Description	Stimulus Generation
FSM_reset	verify the output when rst is high	we directed at start of sinulation then we randomize under constraint that it will be off most of time
FSM_INPUT	We verify output y and counter when input X generated	randomized under constraint that X is high for 67% of simulation time


Functionality Check

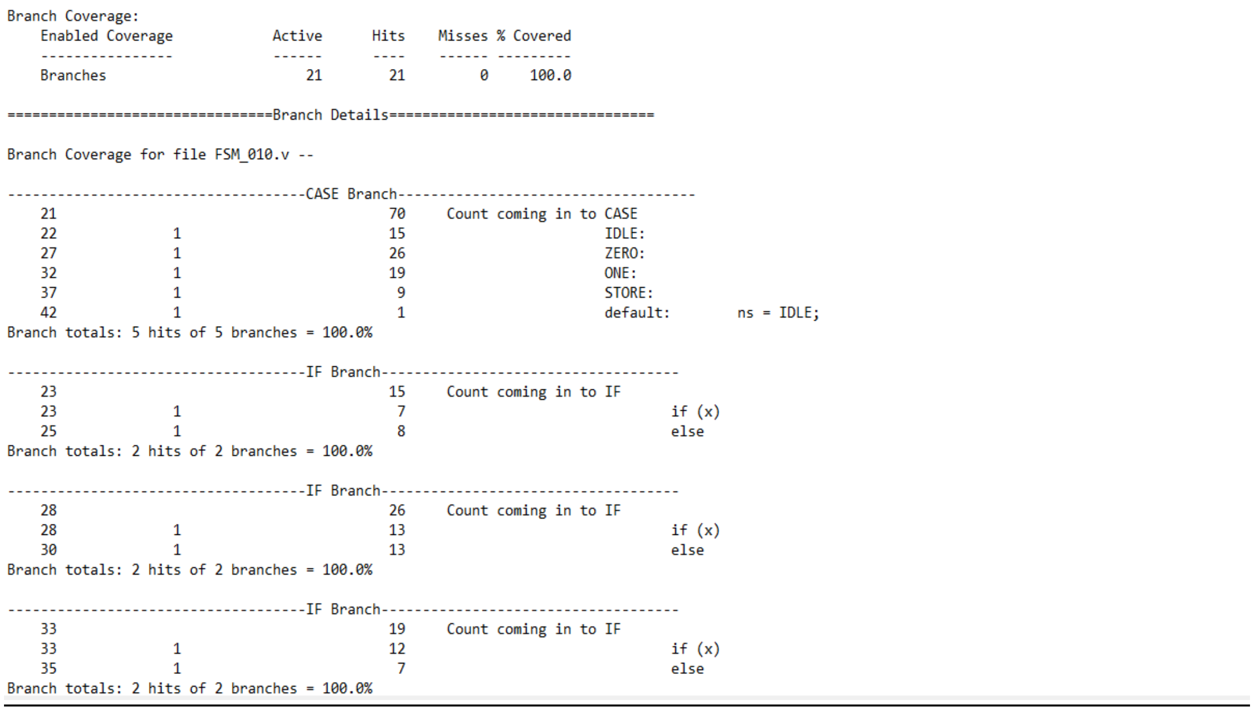
: we check by send object to check result task and compare with output_expected from golden model task

we check by send object to check result task and compare with output_expected from golden model task

Coverage report



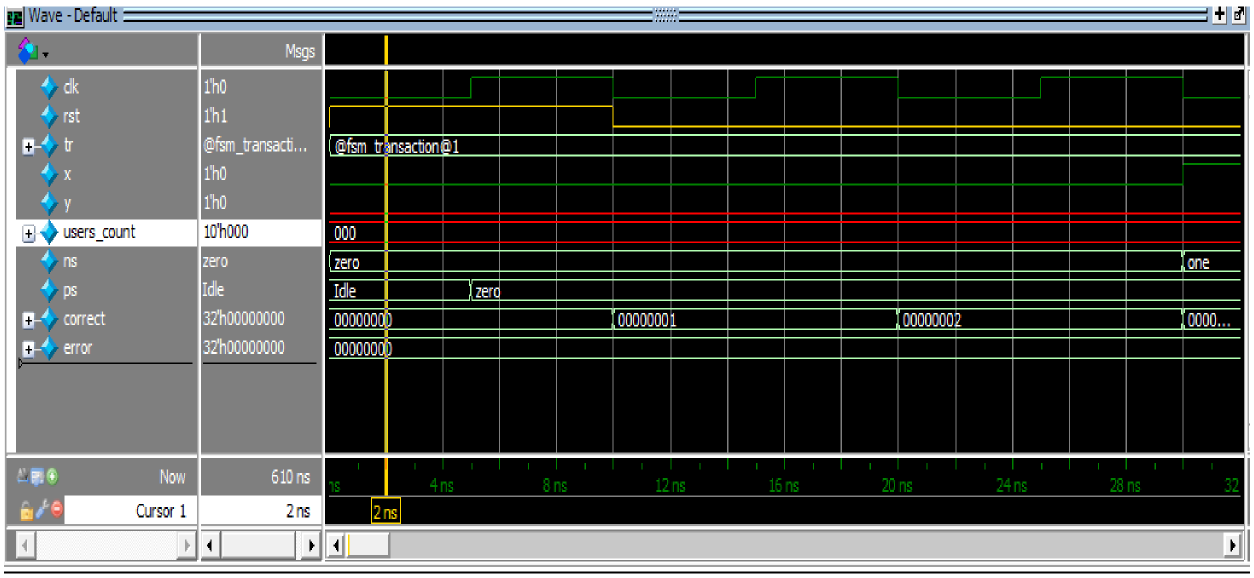
```
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           17        17         0     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                          70          always @(*) begin
    21                                                             case (cs)
    22                                                                     IDLE:
```

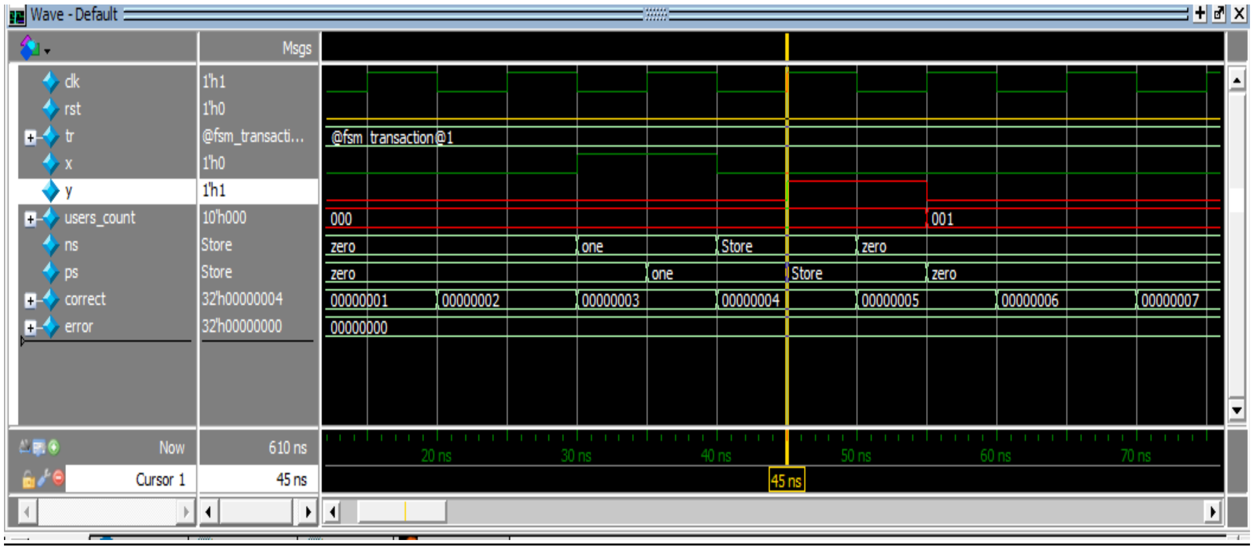


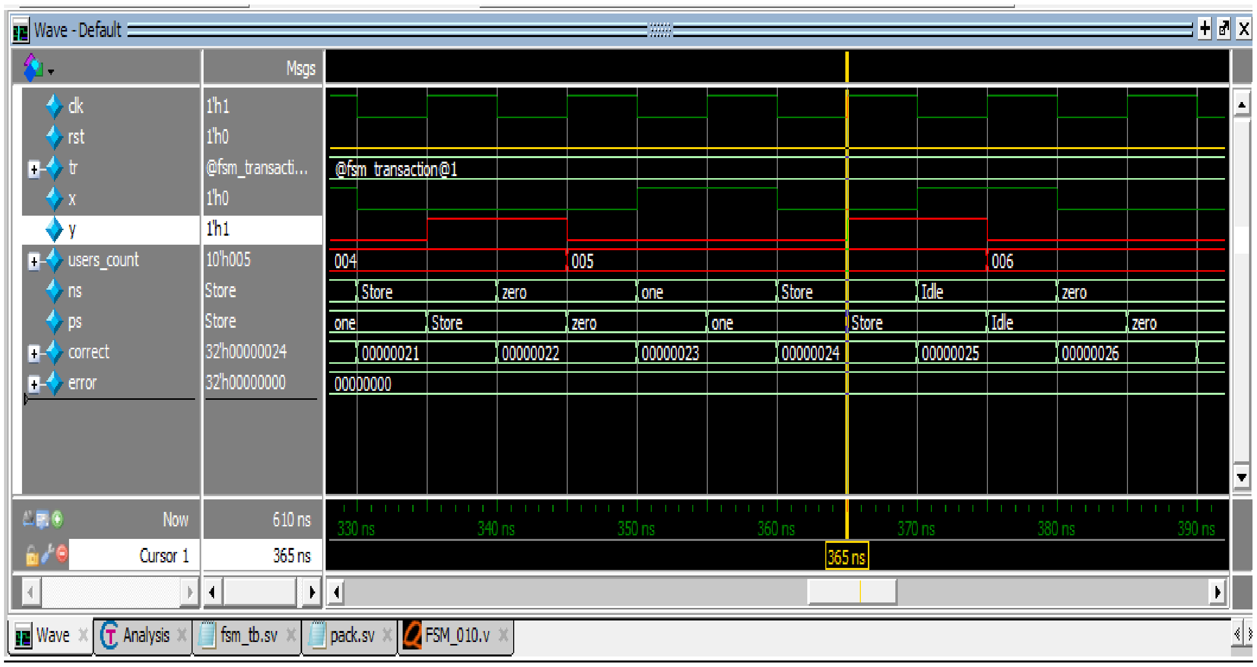
```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        21        21         0     100.0

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

Branch Coverage for file FSM_010.v --

------------------------------------CASE Branch----------------------------------
    21                                        70     Count coming in to CASE
    22              1                         15                         IDLE:
    27              1                         26                         ZERO:
    32              1                         19                         ONE:
    37              1                          9                         STORE:
    42              1                          1                         default:        ns = IDLE;
Branch totals: 5 hits of 5 branches = 100.0%

------------------------------------IF Branch------------------------------------
    23                                        15     Count coming in to IF
    23              1                          7                                 if (x)
    25              1                          8                                 else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    28                                        26     Count coming in to IF
    28              1                         13                                 if (x)
    30              1                         13                                 else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    33                                        19     Count coming in to IF
    33              1                         12                                 if (x)
    35              1                          7                                 else
Branch totals: 2 hits of 2 branches = 100.0%
```

```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       22        22         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[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

Total Node Count     =          11
Toggled Node Count   =          11
Untoggled Node Count =           0

Toggle Coverage      =      100.0% (22 of 22 bins)


Total Coverage By File (code coverage only, filtered view): 100.0%
```

Simulation




Wave - Default
Msgs
clk 1'h1
rst 1'h0
tr @fsm_transacti...
x 1'h0
y 1'h1
users_count 10'h000
ns Store
ps Store
correct 32'h00000004
error 32'h00000000
@fsm_transaction@1
000
001
zero
one
Store
zero
zero
one
Store
zero
00000001
00000002
00000003
00000004
00000005
00000006
00000007
00000000
Now 610 ns
Cursor 1 45 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
45 ns


Wave - Default
Msgs
clk 1'h1
rst 1'h0
tr @fsm_transacti...
x 1'h0
y 1'h1
users_count 10'h005
ns Store
ps Store
correct 32'h00000024
error 32'h00000000
@fsm_transaction@1
004
005
006
Store
zero
one
Store
Idle
zero
one
Store
zero
one
Store
Idle
zero
00000021
00000022
00000023
00000024
00000025
00000026
00000000
Now 610 ns
Cursor 1 365 ns
330 ns
340 ns
350 ns
360 ns
370 ns
380 ns
390 ns
365 ns
Wave
Analysis
fsm_tb.sv
pack.sv
FSM_010.v