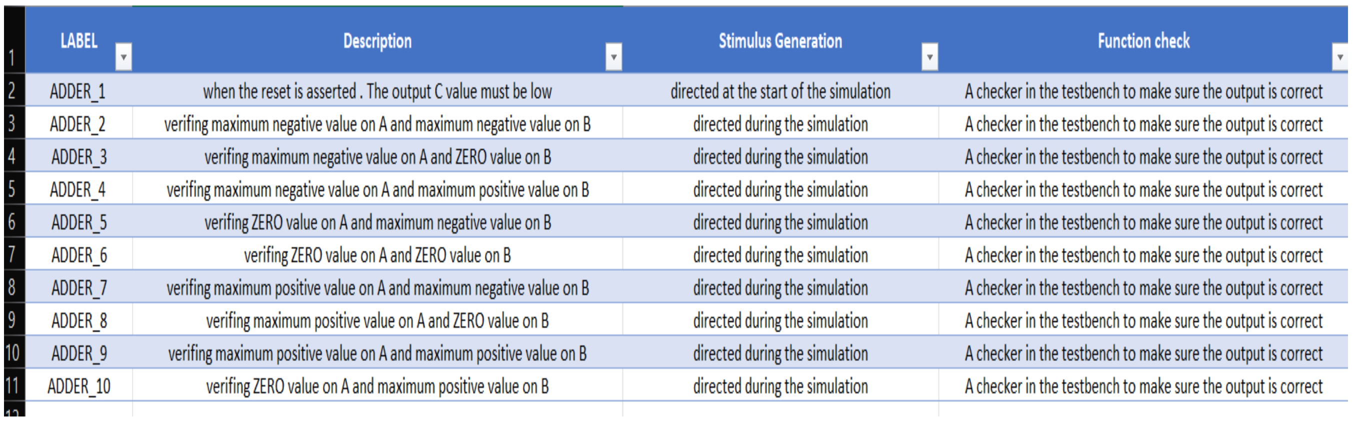
Assignment 1

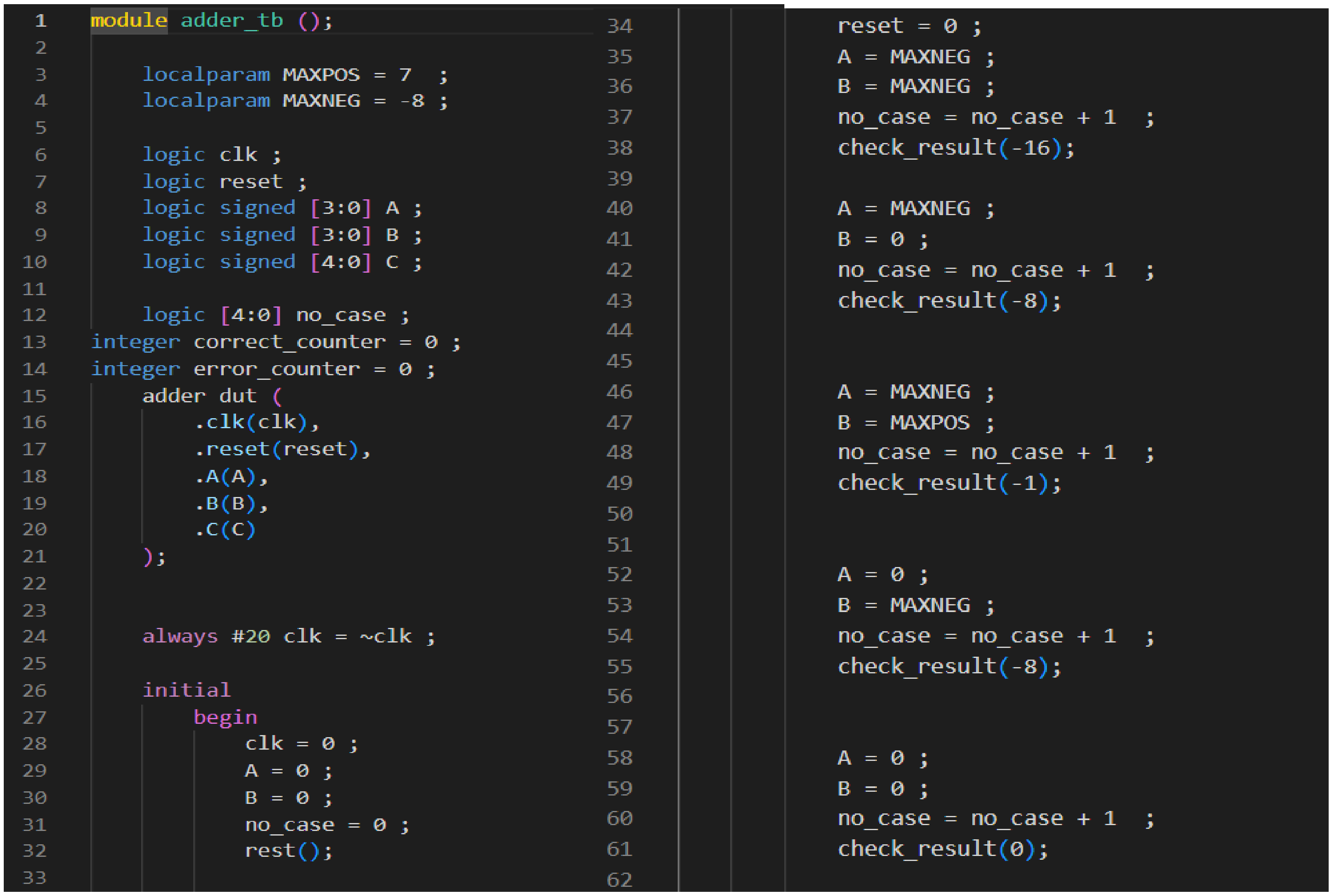
1- Adder

I. Verification plan

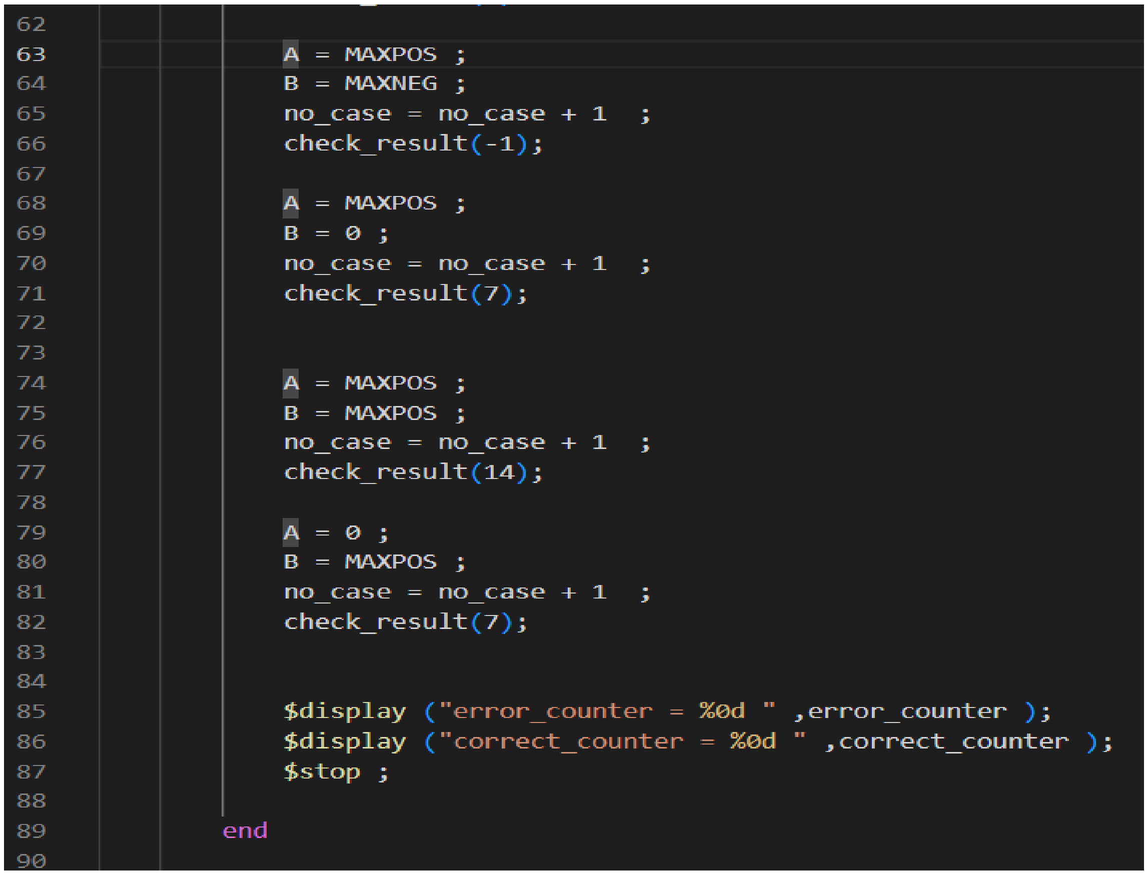


LABEL	Description	Stimulus Generation	Function check
ADDER_1	when the reset is asserted . The output C value must be low	directed at the start of the simulation	A checker in the testbench to make sure the output is correct
ADDER_2	verifing maximum negative value on A and maximum negative value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_3	verifing maximum negative value on A and ZERO value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_4	verifing maximum negative value on A and maximum positive value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_5	verifing ZERO value on A and maximum negative value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_6	verifing ZERO value on A and ZERO value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_7	verifing maximum positive value on A and maximum negative value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_8	verifing maximum positive value on A and ZERO value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_9	verifing maximum positive value on A and maximum positive value on B	directed during the simulation	A checker in the testbench to make sure the output is correct
ADDER_10	verifing ZERO value on A and maximum positive value on B	directed during the simulation	A checker in the testbench to make sure the output is correct

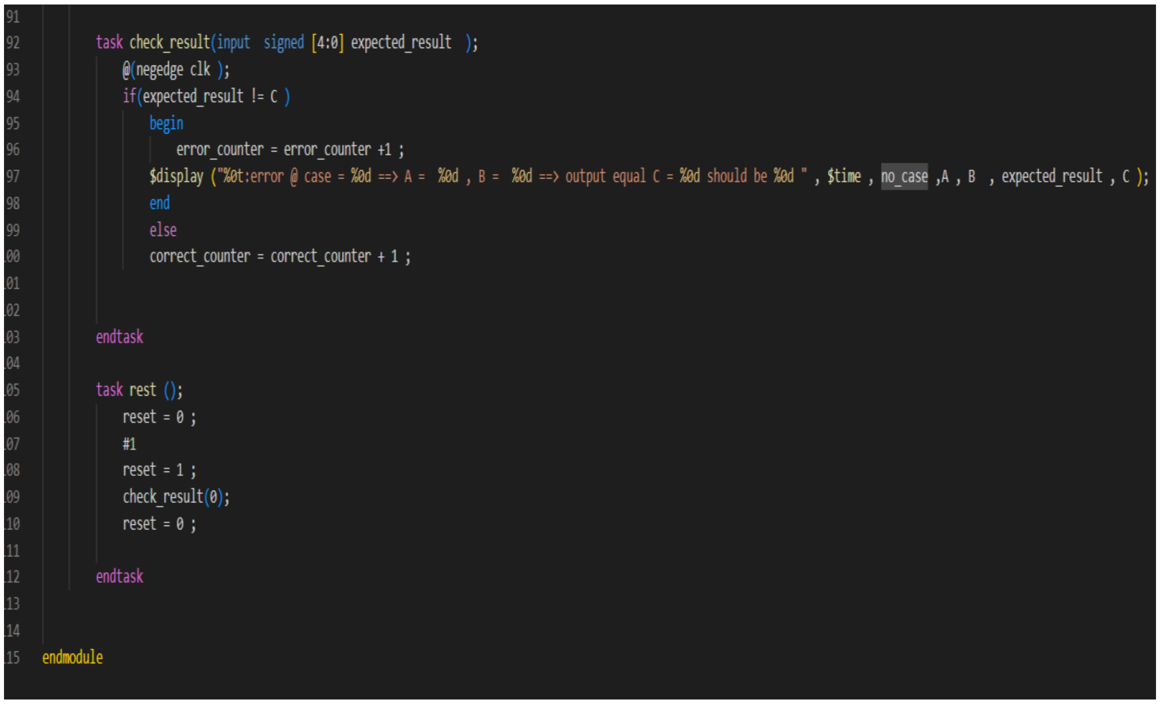
II. Testbench



```
module adder_tb ();

    localparam MAXPOS = 7  ;
    localparam MAXNEG = -8 ;

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

    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)
    );


    always #20 clk = ~clk ;

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

            reset = 0 ;
            A = MAXNEG ;
            B = MAXNEG ;
            no_case = no_case + 1  ;
            check_result(-16);

            A = MAXNEG ;
            B = 0 ;
            no_case = no_case + 1  ;
            check_result(-8);


            A = MAXNEG ;
            B = MAXPOS ;
            no_case = no_case + 1  ;
            check_result(-1);


            A = 0 ;
            B = MAXNEG ;
            no_case = no_case + 1  ;
            check_result(-8);


            A = 0 ;
            B = 0 ;
            no_case = no_case + 1  ;
            check_result(0);

```

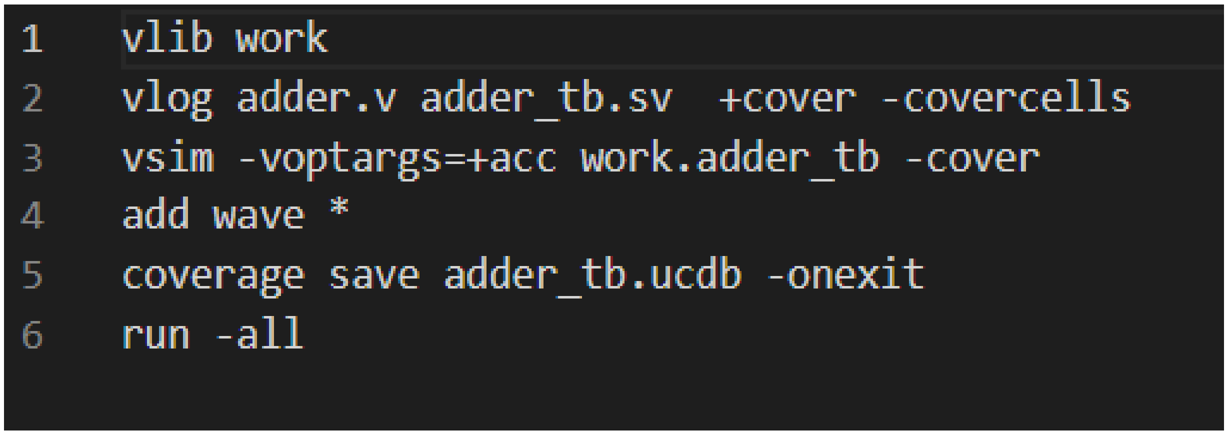


```
            A = MAXPOS ;
            B = MAXNEG ;
            no_case = no_case + 1   ;
            check_result(-1);

            A = MAXPOS ;
            B = 0 ;
            no_case = no_case + 1   ;
            check_result(7);


            A = MAXPOS ;
            B = MAXPOS ;
            no_case = no_case + 1   ;
            check_result(14);

            A = 0 ;
            B = MAXPOS ;
            no_case = no_case + 1   ;
            check_result(7);


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

        end

```



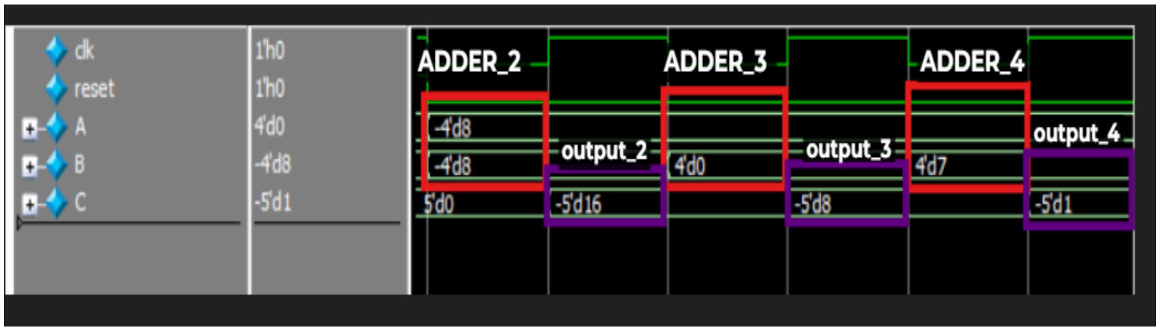
```

    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 rest ();
        reset = 0 ;
        #1
        reset = 1 ;
        check_result(0);
        reset = 0 ;

    endtask


endmodule
```

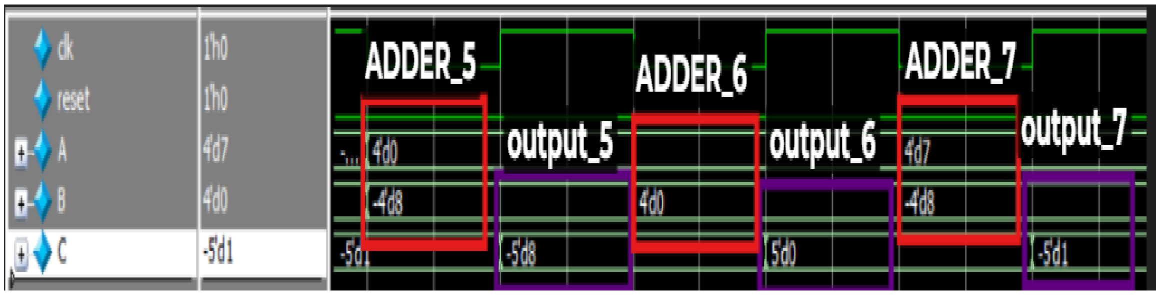
III. Do file



```
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
```

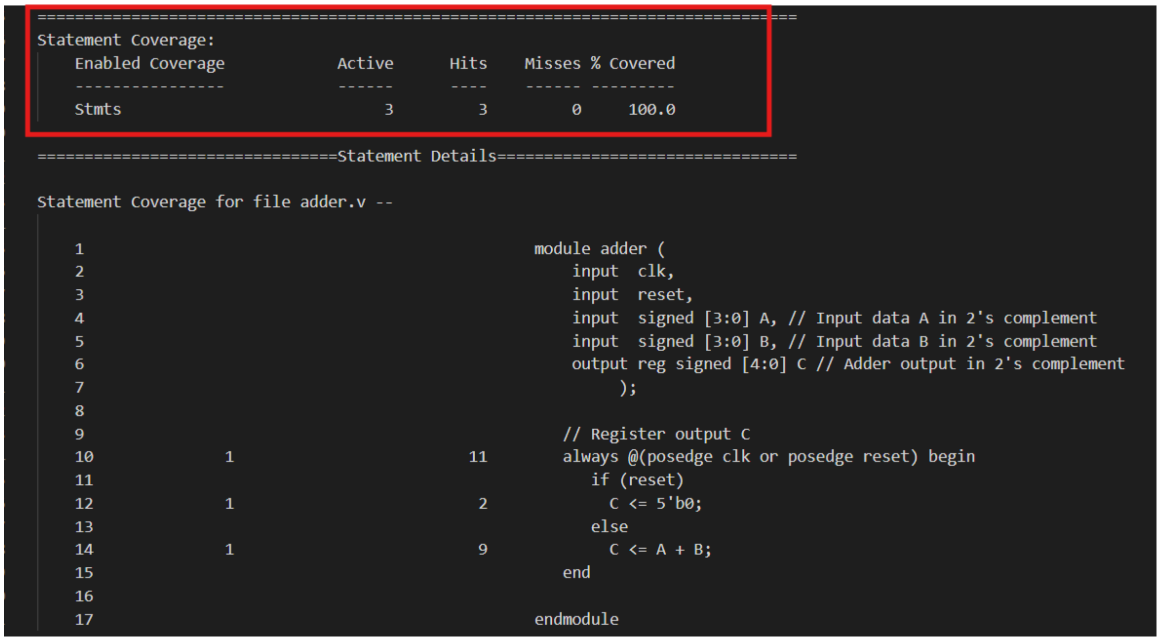
IV. Waveform





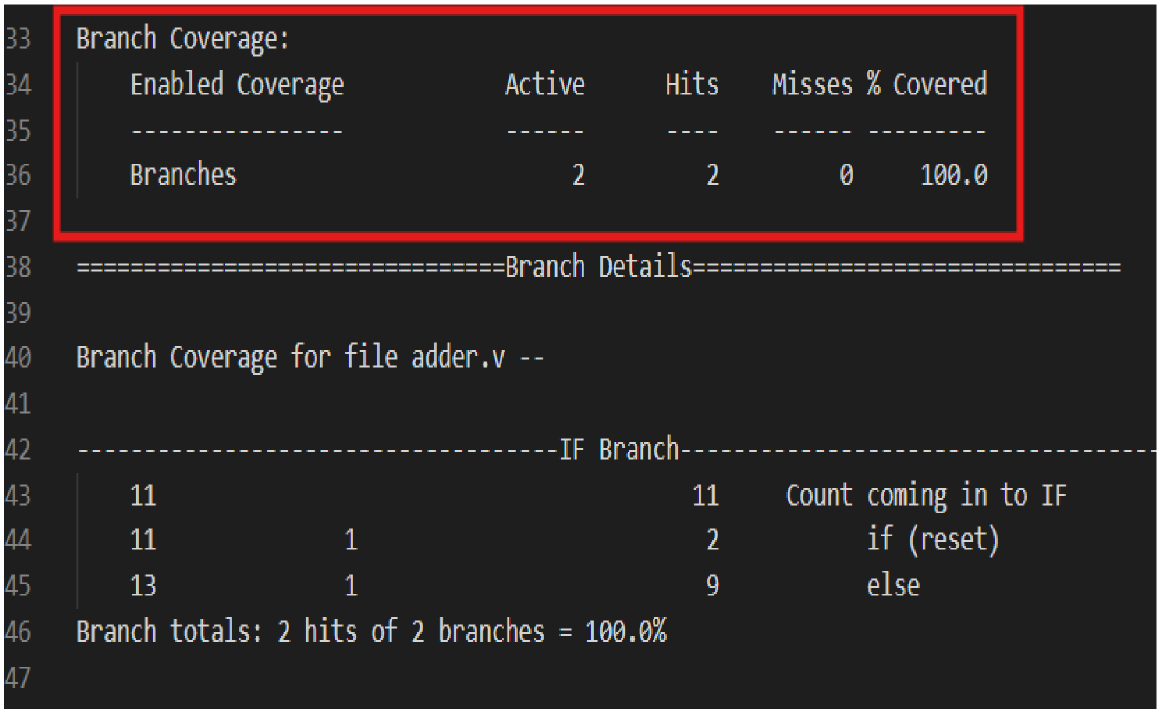
V. Coverage

1- Branch



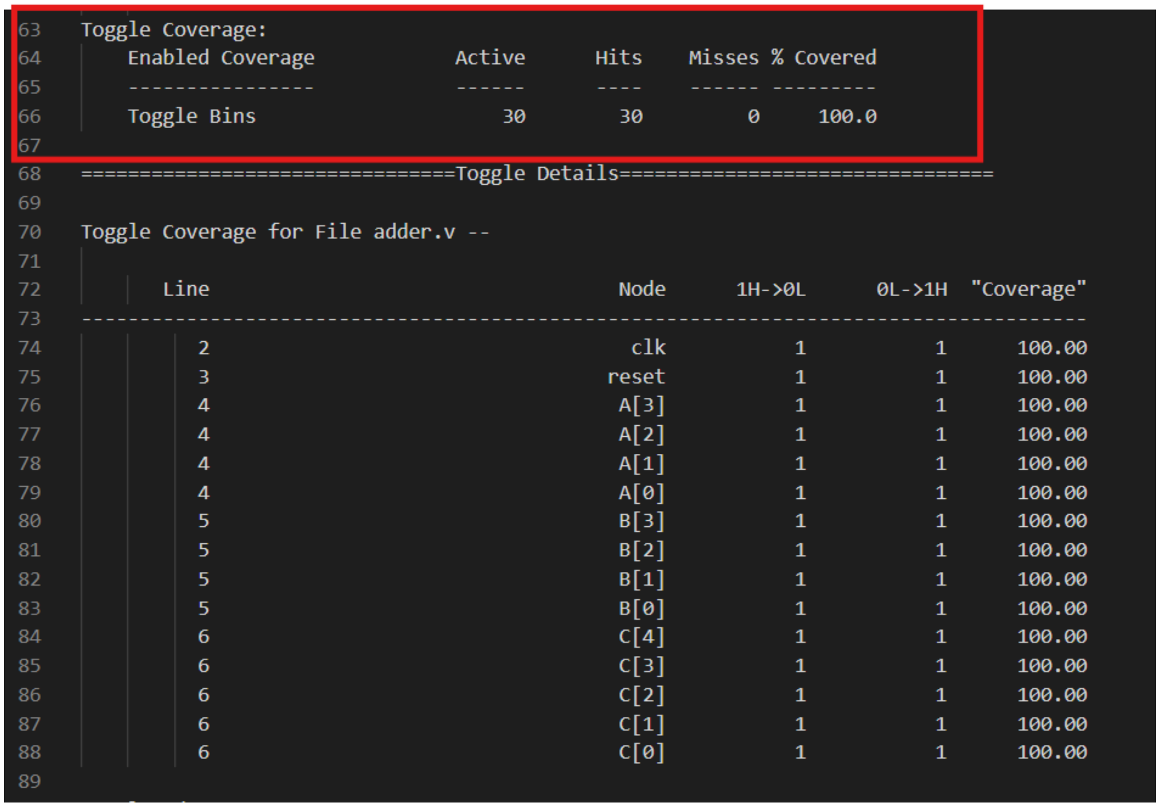
```
=================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            3         3         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                          11           always @(posedge clk or posedge reset) begin
    11                                                         if (reset)
    12              1                           2                C <= 5'b0;
    13                                                         else
    14              1                           9                C <= A + B;
    15                                                      end
    16
    17                                                   endmodule
```

2- Statement



```
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                                          11     Count coming in to IF
    11              1                            2           if (reset)
    13              1                            9           else
Branch totals: 2 hits of 2 branches = 100.0%
```

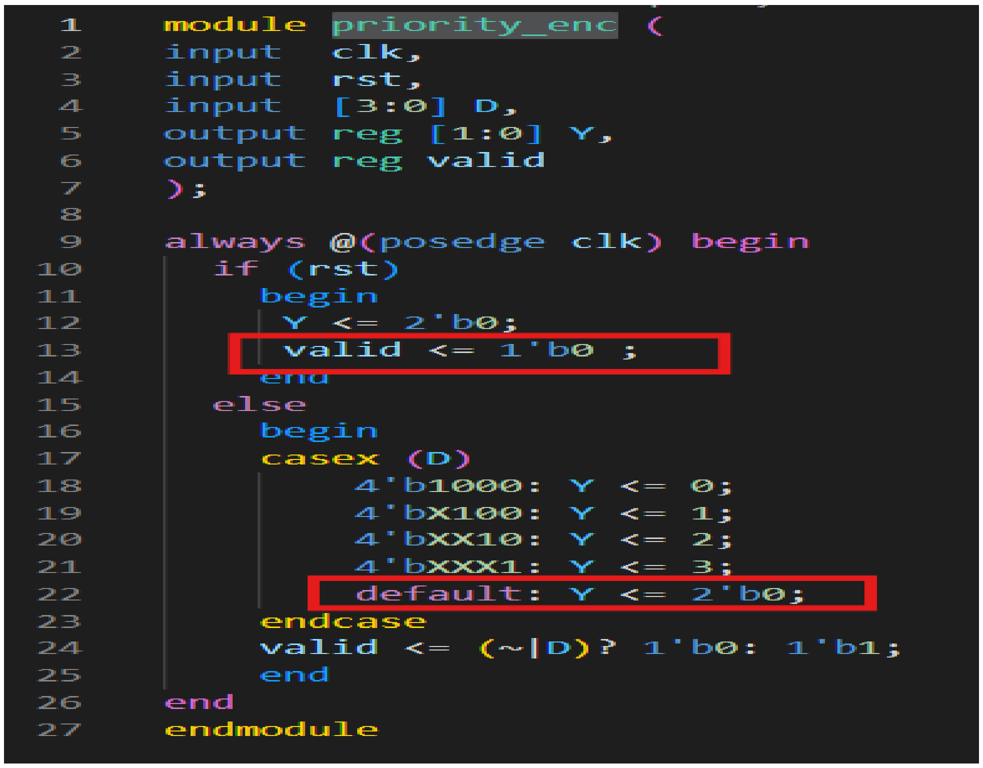
3- Toggle coverage



```
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
```

2- Priority encoder

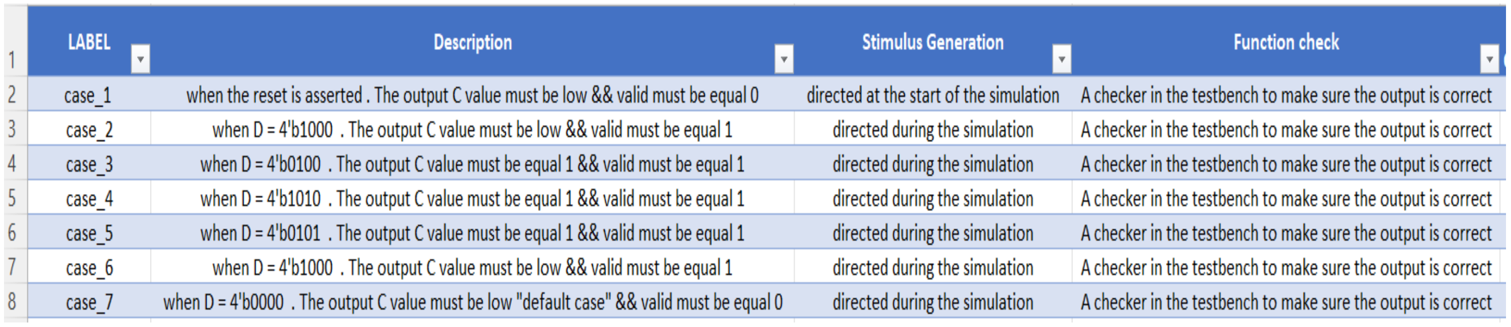
I. Design



```
module priority_enc (
input   clk,
input   rst,
input   [3:0] D,
output reg [1:0] Y,
output reg valid
);

always @(posedge clk) begin
    if (rst)
        begin
         Y <= 2'b0;
         valid <= 1'b0 ;
        end
    else
        begin
        casex (D)
            4'b1000: Y <= 0;
            4'bX100: Y <= 1;
            4'bXX10: Y <= 2;
            4'bXXX1: Y <= 3;
            default: Y <= 2'b0;
        endcase
        valid <= (~|D)? 1'b0: 1'b1;
        end
end
endmodule
```

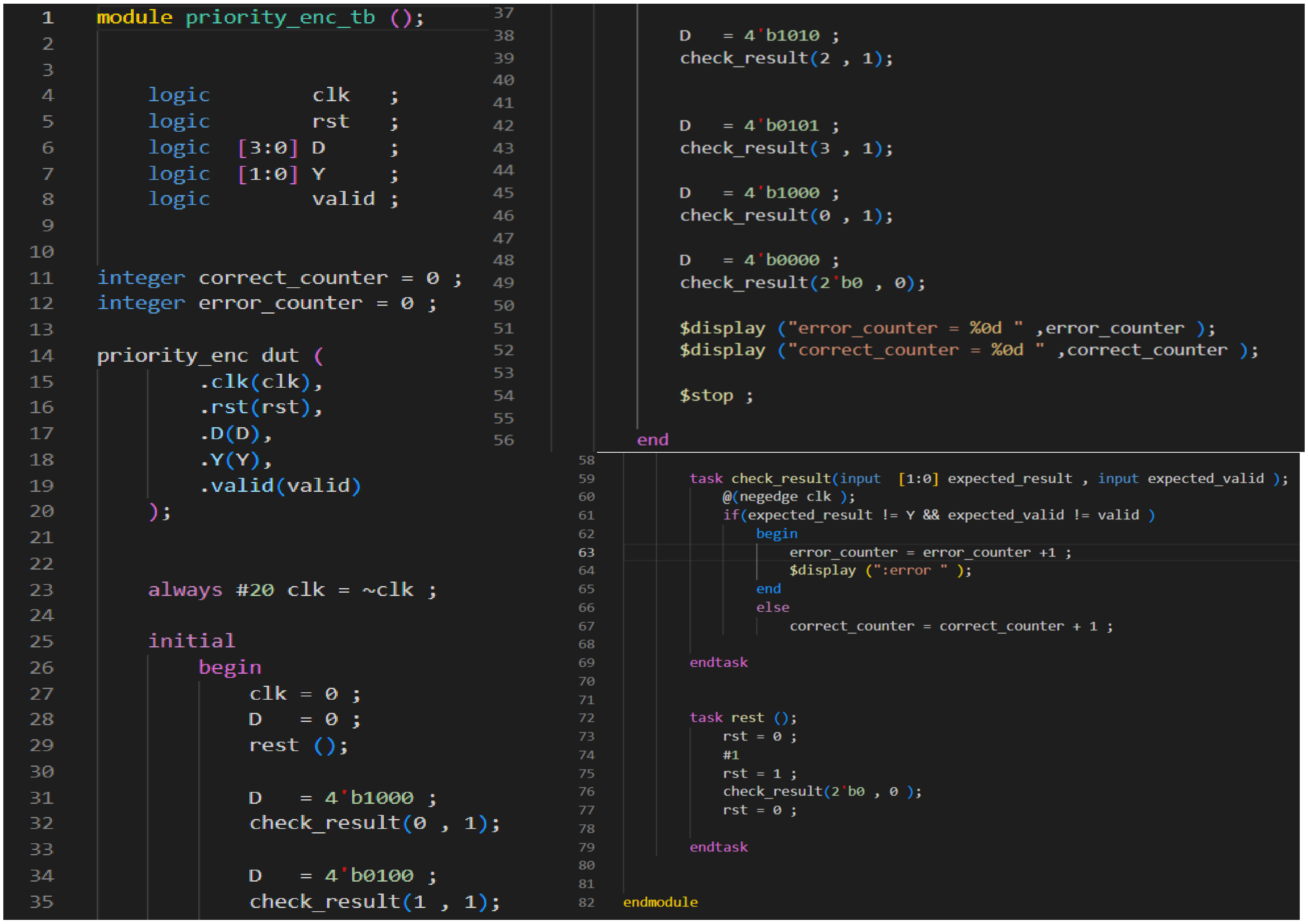
Add the Valid signal is equal to zero if rst is high. And add default case in the case when the input “ D = 4’b0000 “

II. Verification plan



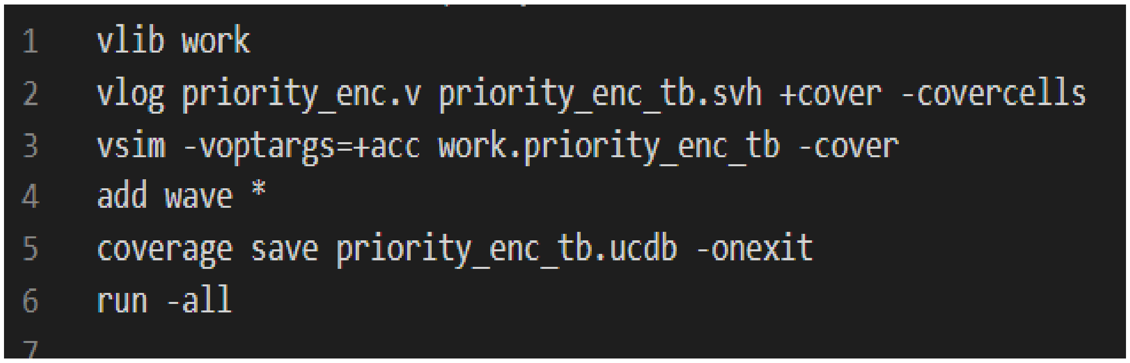
LABEL	Description	Stimulus Generation	Function check
case_1	when the reset is asserted . The output C value must be low && valid must be equal 0	directed at the start of the simulation	A checker in the testbench to make sure the output is correct
case_2	when D = 4'b1000 . The output C value must be low && valid must be equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_3	when D = 4'b0100 . The output C value must be equal 1 && valid must be equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_4	when D = 4'b1010 . The output C value must be equal 1 && valid must be equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_5	when D = 4'b0101 . The output C value must be equal 1 && valid must be equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_6	when D = 4'b1000 . The output C value must be low && valid must be equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_7	when D = 4'b0000 . The output C value must be low "default case" && valid must be equal 0	directed during the simulation	A checker in the testbench to make sure the output is correct

III. Testbench



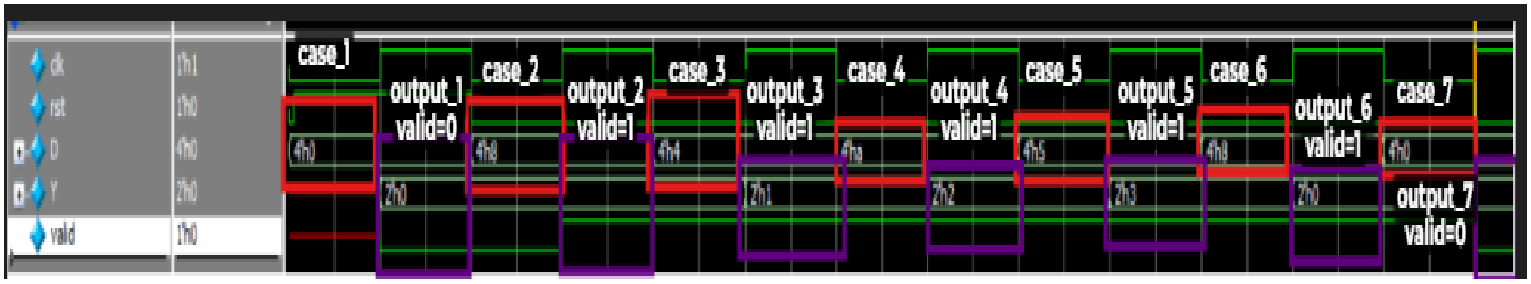
```
module priority_enc_tb ();


    logic         clk   ;
    logic         rst   ;
    logic   [3:0] D     ;
    logic   [1:0] Y     ;
    logic         valid ;


integer correct_counter = 0 ;
integer error_counter = 0 ;

priority_enc dut (
        .clk(clk),
        .rst(rst),
        .D(D),
        .Y(Y),
        .valid(valid)
    );


    always #20 clk = ~clk ;

    initial
        begin
            clk = 0 ;
            D   = 0 ;
            rest ();

            D   = 4'b1000 ;
            check_result(0 , 1);

            D   = 4'b0100 ;
            check_result(1 , 1);

            D   = 4'b1010 ;
            check_result(2 , 1);


            D   = 4'b0101 ;
            check_result(3 , 1);

            D   = 4'b1000 ;
            check_result(0 , 1);

            D   = 4'b0000 ;
            check_result(2'b0 , 0);

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

            $stop ;

        end

        task check_result(input  [1:0] expected_result , input expected_valid );
            @(negedge clk );
            if(expected_result != Y && expected_valid != valid )
                begin
                    error_counter = error_counter +1 ;
                    $display (":error " );
                end
                else
                    correct_counter = correct_counter + 1 ;

        endtask


        task rest ();
            rst = 0 ;
            #1
            rst = 1 ;
            check_result(2'b0 , 0 );
            rst = 0 ;

        endtask

endmodule
```

IV. Do file



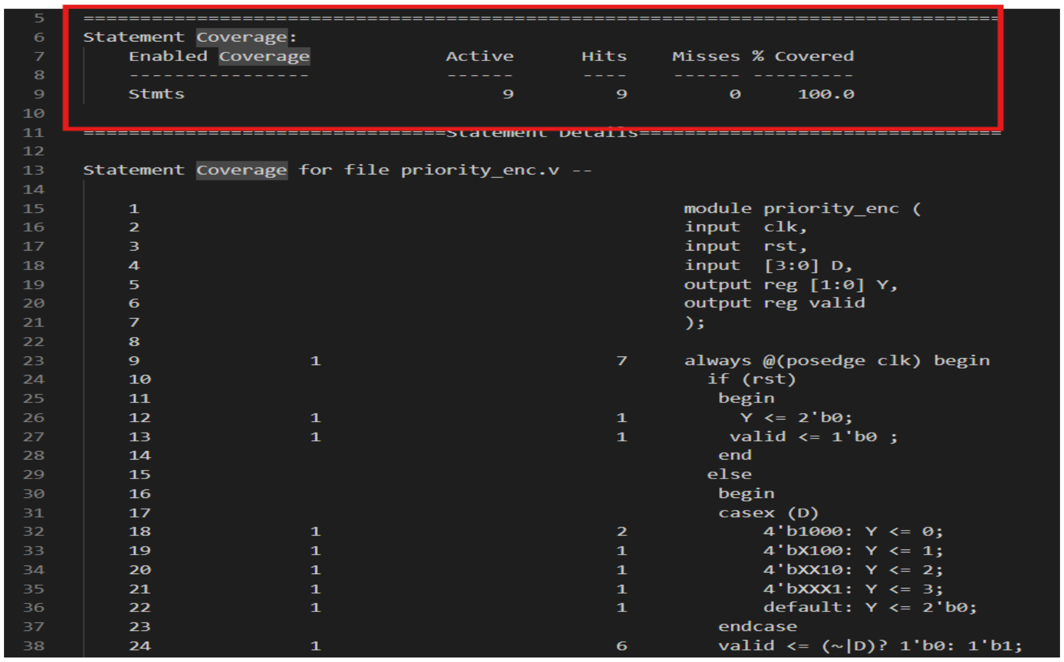
```
vlib work
vlog priority_enc.v priority_enc_tb.svh +cover -covercells
vsim -voptargs=+acc work.priority_enc_tb -cover
add wave *
coverage save priority_enc_tb.ucdb -onexit
run -all
```

V. Waveform



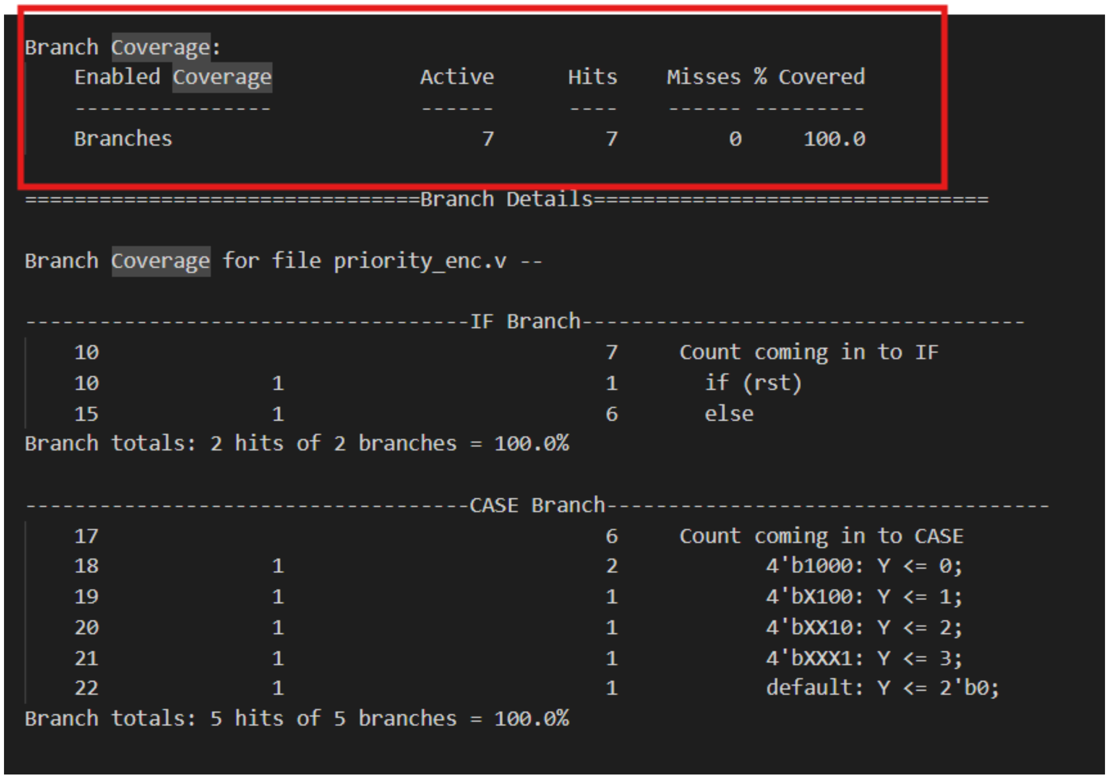
VI. Coverage

1- Branch



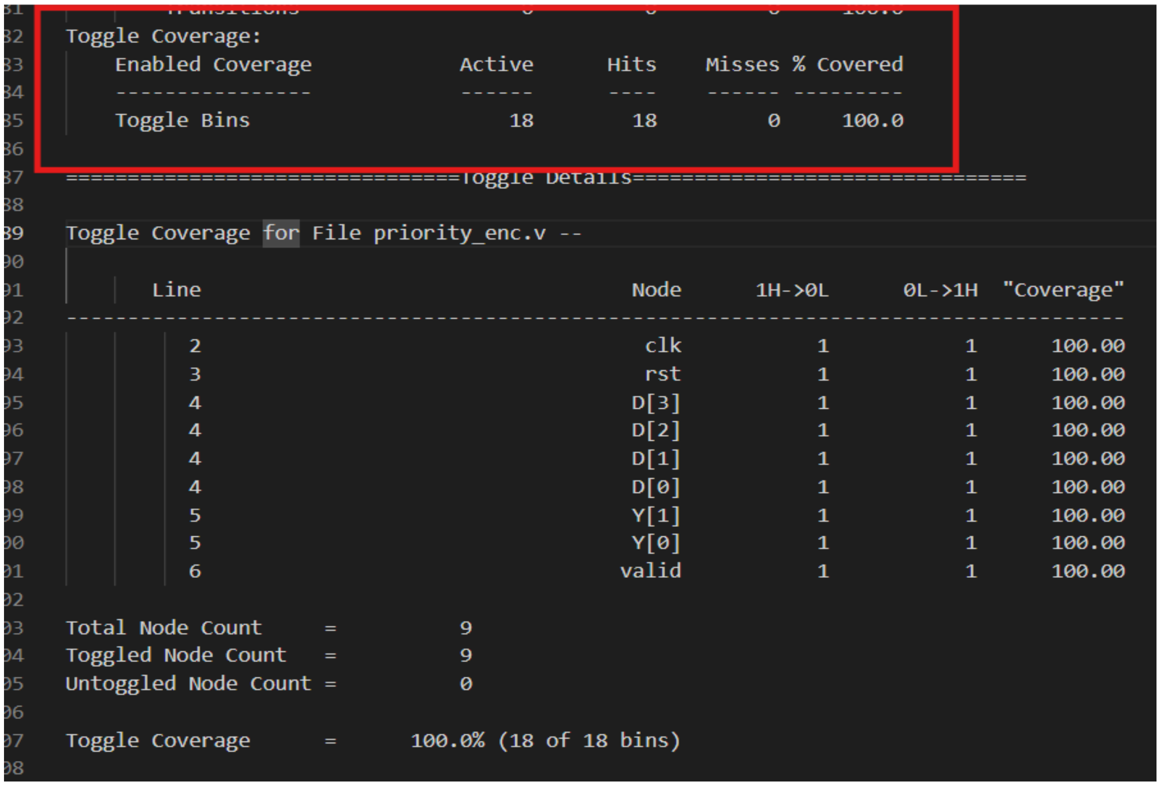
```
=====================================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                              9         9         0     100.0

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

Statement Coverage for file priority_enc.v --

    1                                                   module priority_enc (
    2                                                   input  clk,
    3                                                   input  rst,
    4                                                   input  [3:0] D,
    5                                                   output reg [1:0] Y,
    6                                                   output reg valid
    7                                                   );
    8
    9               1                          7        always @(posedge clk) begin
    10                                                    if (rst)
    11                                                     begin
    12              1                          1             Y <= 2'b0;
    13              1                          1            valid <= 1'b0 ;
    14                                                     end
    15                                                    else
    16                                                     begin
    17                                                     casex (D)
    18              1                          2               4'b1000: Y <= 0;
    19              1                          1               4'bX100: Y <= 1;
    20              1                          1               4'bXX10: Y <= 2;
    21              1                          1               4'bXXX1: Y <= 3;
    22              1                          1               default: Y <= 2'b0;
    23                                                     endcase
    24              1                          6           valid <= (~|D)? 1'b0: 1'b1;
```

2- Statement



```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         7         7         0     100.0

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

Branch Coverage for file priority_enc.v --

------------------------------------IF Branch------------------------------------
    10                                         7      Count coming in to IF
    10              1                          1        if (rst)
    15              1                          6        else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------CASE Branch------------------------------------
    17                                         6      Count coming in to CASE
    18              1                          2             4'b1000: Y <= 0;
    19              1                          1             4'bX100: Y <= 1;
    20              1                          1             4'bXX10: Y <= 2;
    21              1                          1             4'bXXX1: Y <= 3;
    22              1                          1             default: Y <= 2'b0;
Branch totals: 5 hits of 5 branches = 100.0%
```

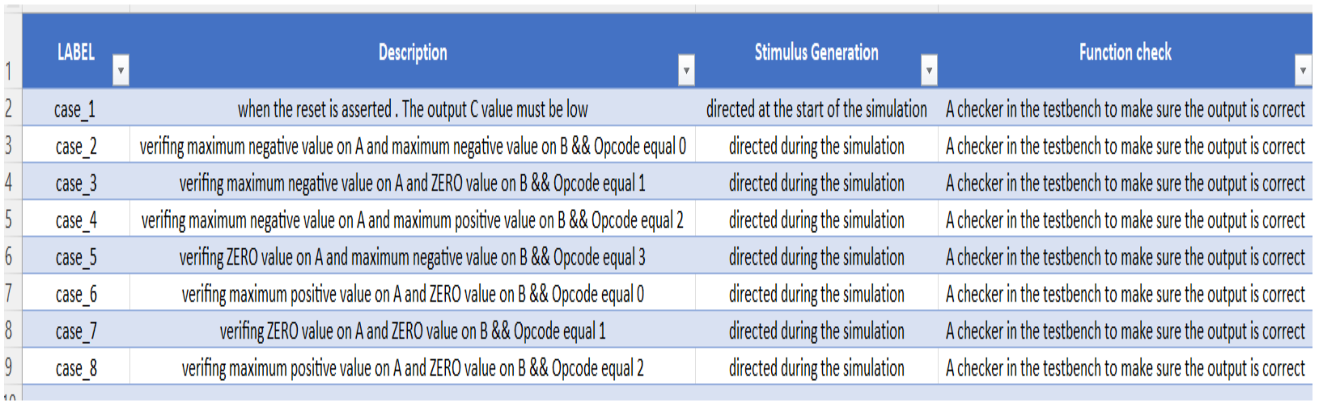
3- Toggle coverage



```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     18        18         0     100.0

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

Toggle Coverage for File priority_enc.v --

       Line                                   Node      1H->0L      0L->1H  "Coverage"
---------------------------------------------------------------------------------------
          2                                    clk           1           1      100.00
          3                                    rst           1           1      100.00
          4                                   D[3]           1           1      100.00
          4                                   D[2]           1           1      100.00
          4                                   D[1]           1           1      100.00
          4                                   D[0]           1           1      100.00
          5                                   Y[1]           1           1      100.00
          5                                   Y[0]           1           1      100.00
          6                                  valid           1           1      100.00

Total Node Count     =          9
Toggled Node Count   =          9
Untoggled Node Count =          0

Toggle Coverage      =      100.0% (18 of 18 bins)
```

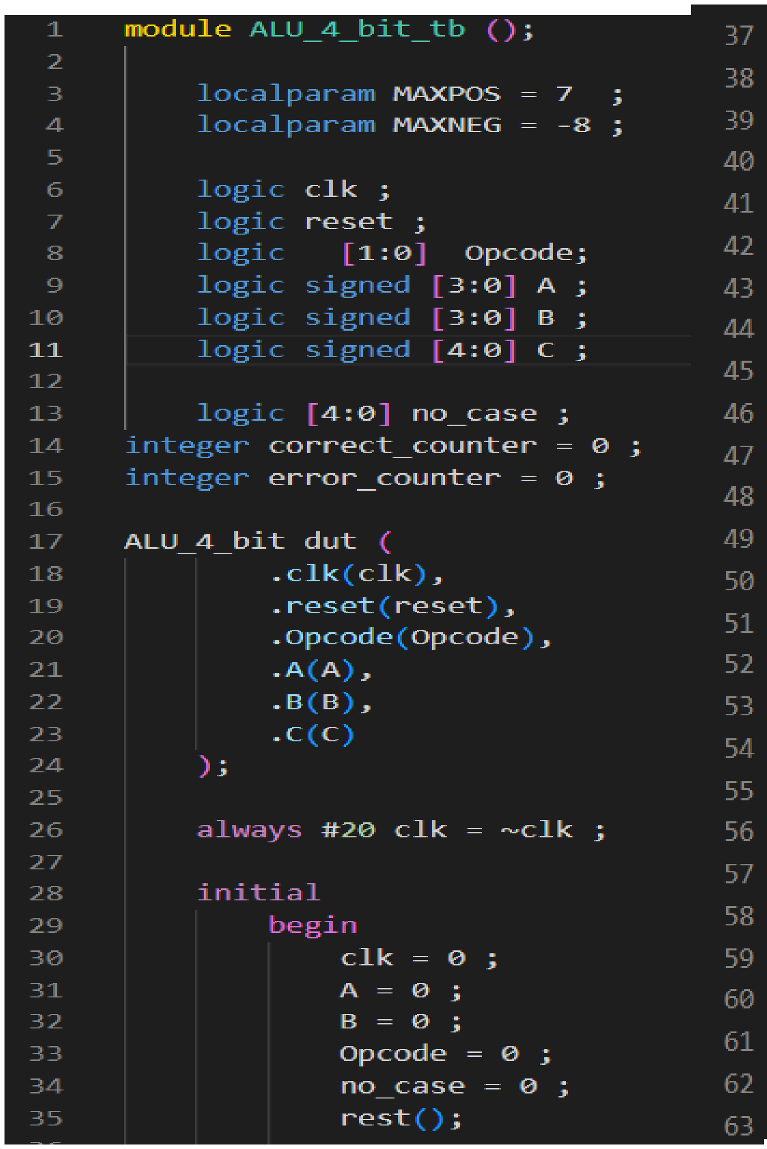
3- Alu

I. Verification plan

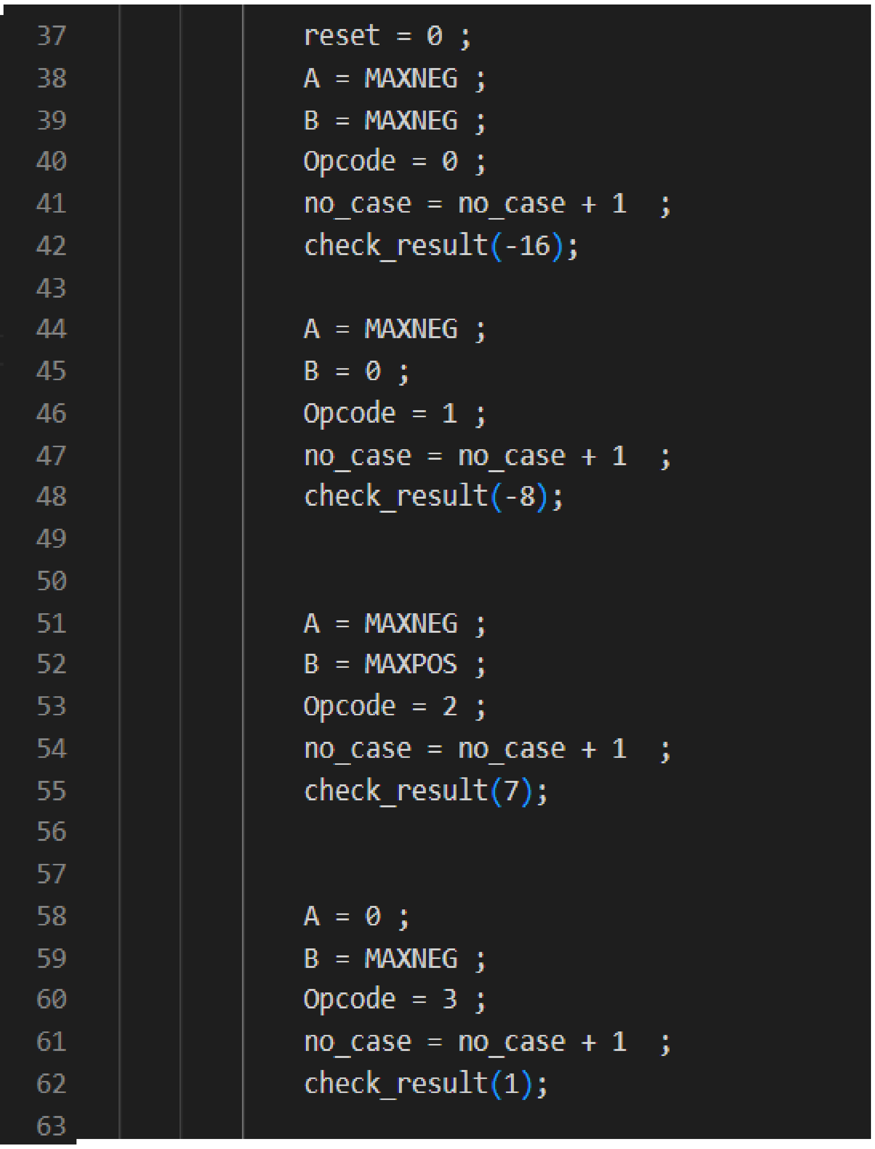


LABEL	Description	Stimulus Generation	Function check
case_1	when the reset is asserted . The output C value must be low	directed at the start of the simulation	A checker in the testbench to make sure the output is correct
case_2	verifing maximum negative value on A and maximum negative value on B && Opcode equal 0	directed during the simulation	A checker in the testbench to make sure the output is correct
case_3	verifing maximum negative value on A and ZERO value on B && Opcode equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_4	verifing maximum negative value on A and maximum positive value on B && Opcode equal 2	directed during the simulation	A checker in the testbench to make sure the output is correct
case_5	verifing ZERO value on A and maximum negative value on B && Opcode equal 3	directed during the simulation	A checker in the testbench to make sure the output is correct
case_6	verifing maximum positive value on A and ZERO value on B && Opcode equal 0	directed during the simulation	A checker in the testbench to make sure the output is correct
case_7	verifing ZERO value on A and ZERO value on B && Opcode equal 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_8	verifing maximum positive value on A and ZERO value on B && Opcode equal 2	directed during the simulation	A checker in the testbench to make sure the output is correct

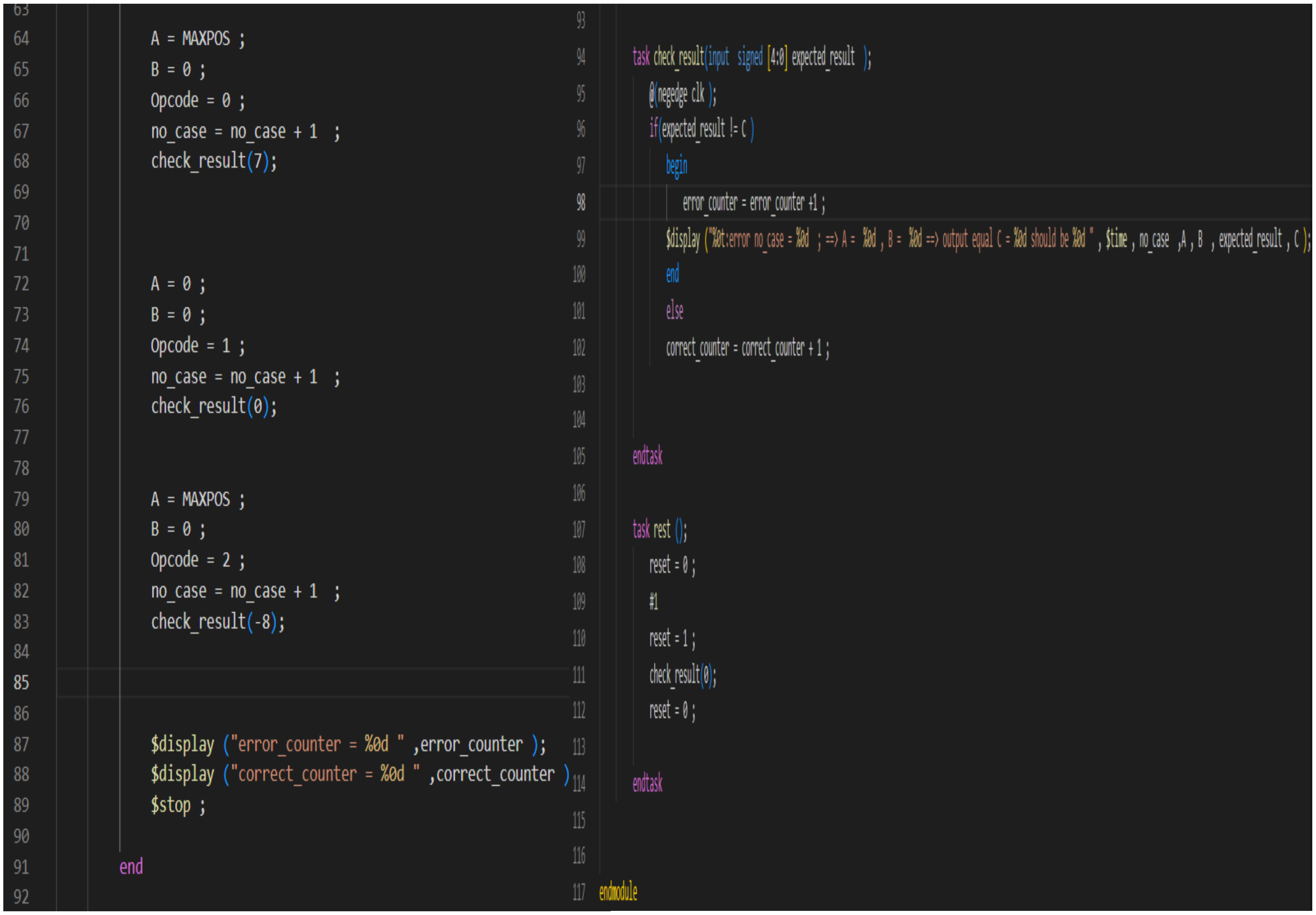
II. Testbench



```
module ALU_4_bit_tb ();

    localparam MAXPOS  = 7   ;
    localparam MAXNEG  = -8  ;

    logic clk ;
    logic reset ;
    logic   [1:0]  Opcode;
    logic signed [3:0] A ;
    logic signed [3:0] B ;
    logic signed [4:0] C ;

    logic [4:0] no_case ;
integer correct_counter = 0 ;
integer error_counter  = 0 ;

ALU_4_bit dut (
        .clk(clk),
        .reset(reset),
        .Opcode(Opcode),
        .A(A),
        .B(B),
        .C(C)
    );

    always #20 clk = ~clk ;

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



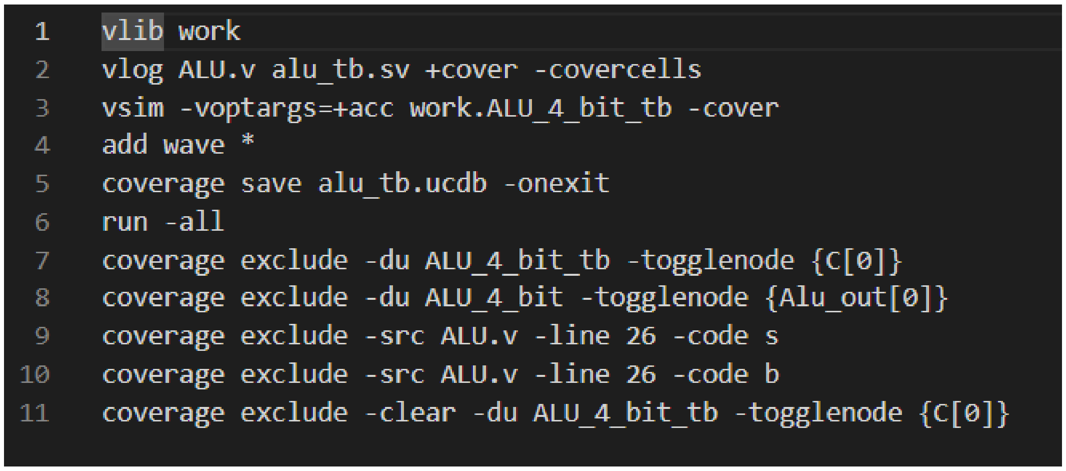
```
        reset = 0 ;
        A = MAXNEG ;
        B = MAXNEG ;
        Opcode = 0 ;
        no_case = no_case + 1  ;
        check_result(-16);

        A = MAXNEG ;
        B = 0 ;
        Opcode = 1 ;
        no_case = no_case + 1  ;
        check_result(-8);


        A = MAXNEG ;
        B = MAXPOS ;
        Opcode = 2 ;
        no_case = no_case + 1  ;
        check_result(7);


        A = 0 ;
        B = MAXNEG ;
        Opcode = 3 ;
        no_case = no_case + 1  ;
        check_result(1);

```



```
            A = MAXPOS ;
            B = 0 ;
            Opcode = 0 ;
            no_case = no_case + 1  ;
            check_result(7);



            A = 0 ;
            B = 0 ;
            Opcode = 1 ;
            no_case = no_case + 1  ;
            check_result(0);


            A = MAXPOS ;
            B = 0 ;
            Opcode = 2 ;
            no_case = no_case + 1  ;
            check_result(-8);




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

        end

```

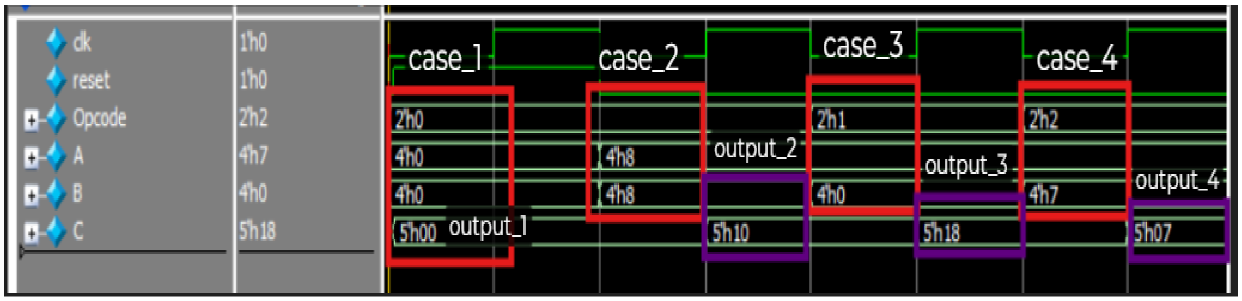
```

    task check_result(input  signed [4:0] expected_result  );
      @(negedge clk );
      if(expected_result != C )
        begin
          error_counter = error_counter +1 ;
        $display ("%0t:error no_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 rest ();
      reset = 0 ;
      #1
      reset = 1 ;
      check_result(0);
      reset = 0 ;

    endtask


endmodule
```

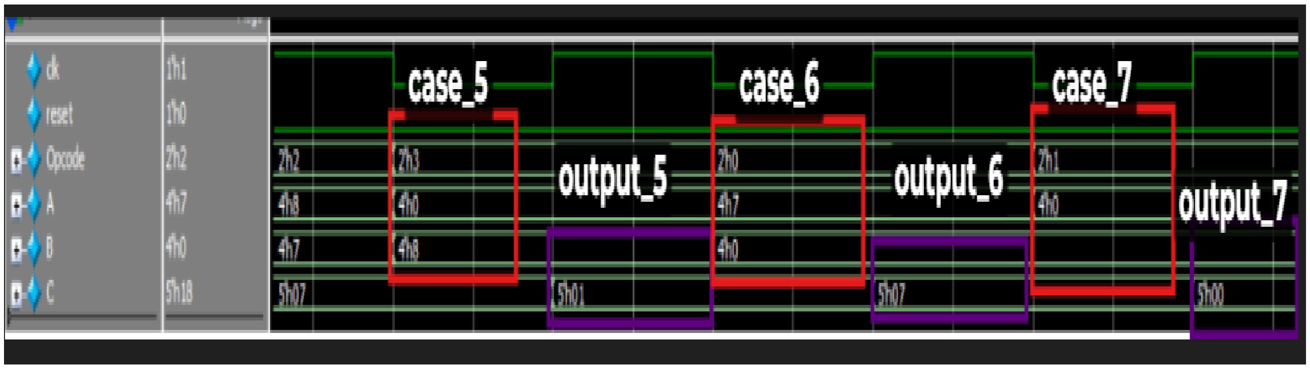
III. Do file



```
vlib work
vlog ALU.v alu_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALU_4_bit_tb -cover
add wave *
coverage save alu_tb.ucdb -onexit
run -all
coverage exclude -du ALU_4_bit_tb -togglenode {C[0]}
coverage exclude -du ALU_4_bit -togglenode {Alu_out[0]}
coverage exclude -src ALU.v -line 26 -code s
coverage exclude -src ALU.v -line 26 -code b
coverage exclude -clear -du ALU_4_bit_tb -togglenode {C[0]}
```

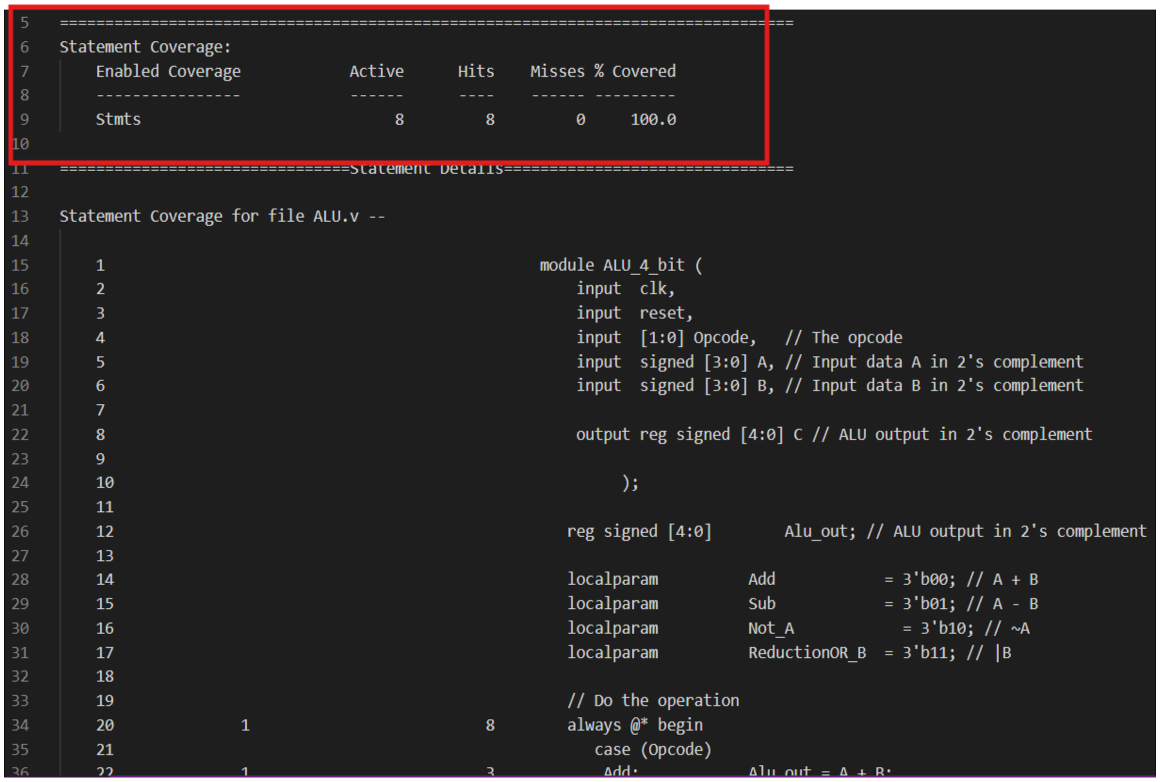
IV. Waveform





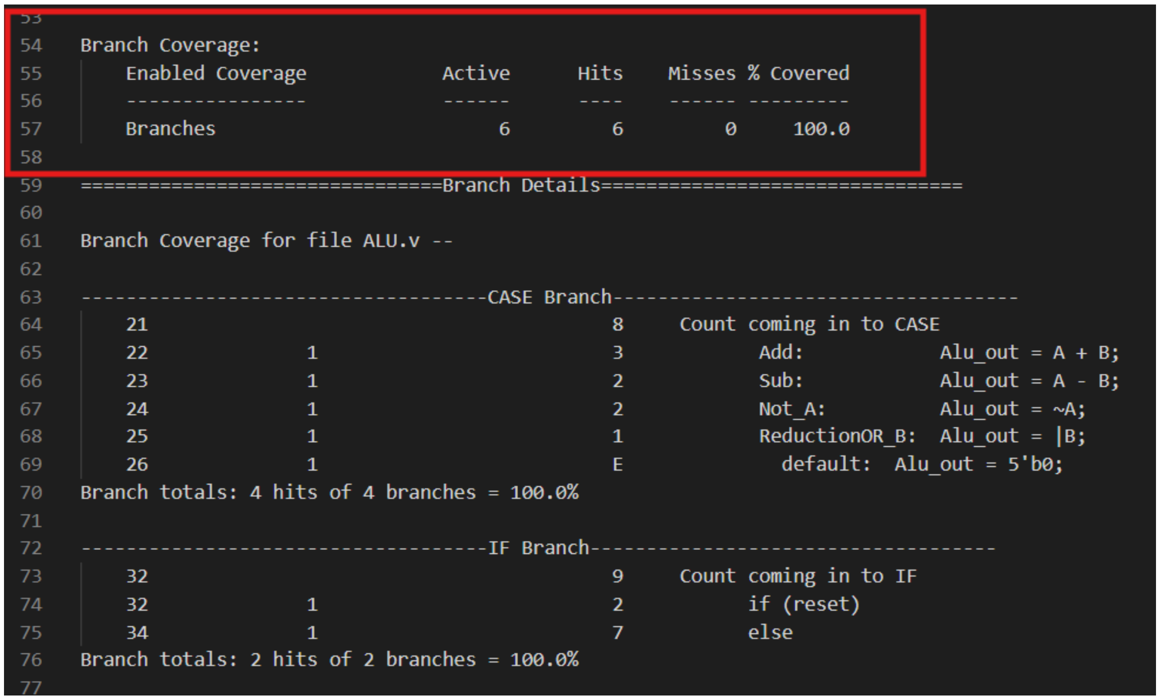
V. Coverage

1- Branch



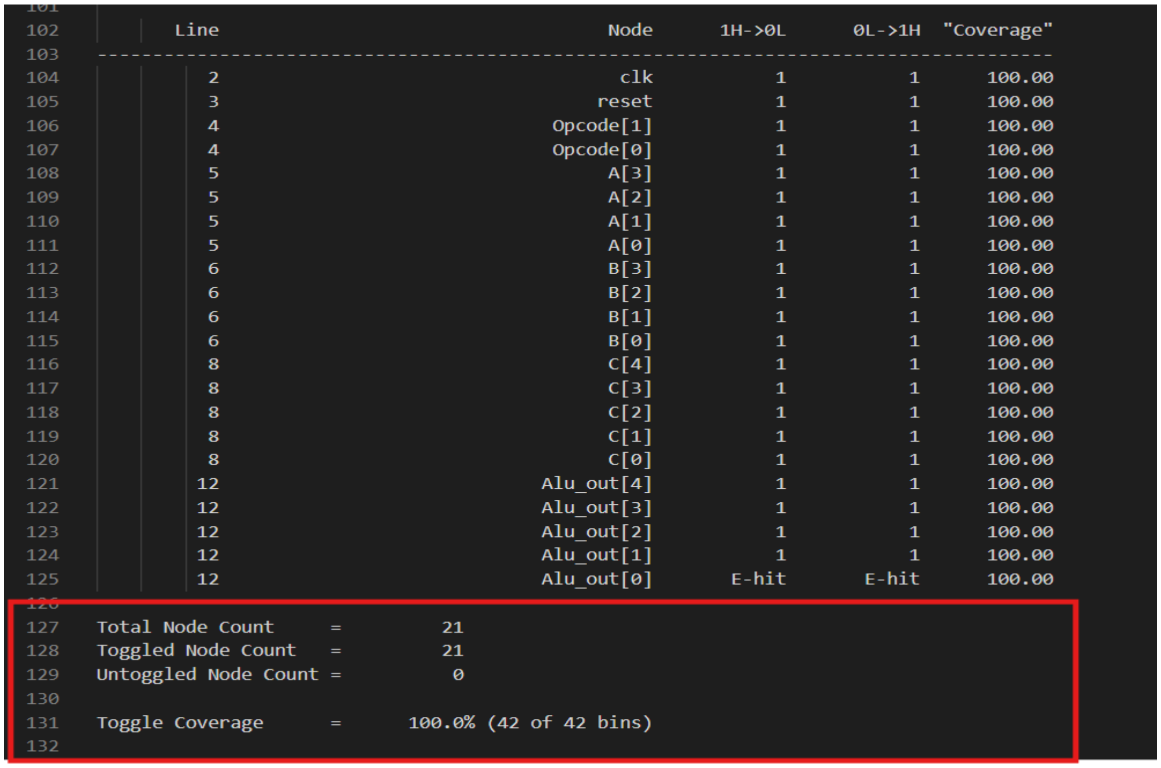
```
==================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         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            = 3'b00; // A + B
    15                                                   localparam          Sub            = 3'b01; // A - B
    16                                                   localparam          Not_A            = 3'b10; // ~A
    17                                                   localparam          ReductionOR_B  = 3'b11; // |B
    18
    19                                                   // Do the operation
    20              1                          8         always @* begin
    21                                                      case (Opcode)
    22              1                          3              Add:              Alu_out = A + B;
```

2- Statement



```
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                                         8     Count coming in to CASE
    22              1                          3            Add:            Alu_out = A + B;
    23              1                          2            Sub:            Alu_out = A - B;
    24              1                          2            Not_A:          Alu_out = ~A;
    25              1                          1            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                                         9     Count coming in to IF
    32              1                          2           if (reset)
    34              1                          7           else
Branch totals: 2 hits of 2 branches = 100.0%
```

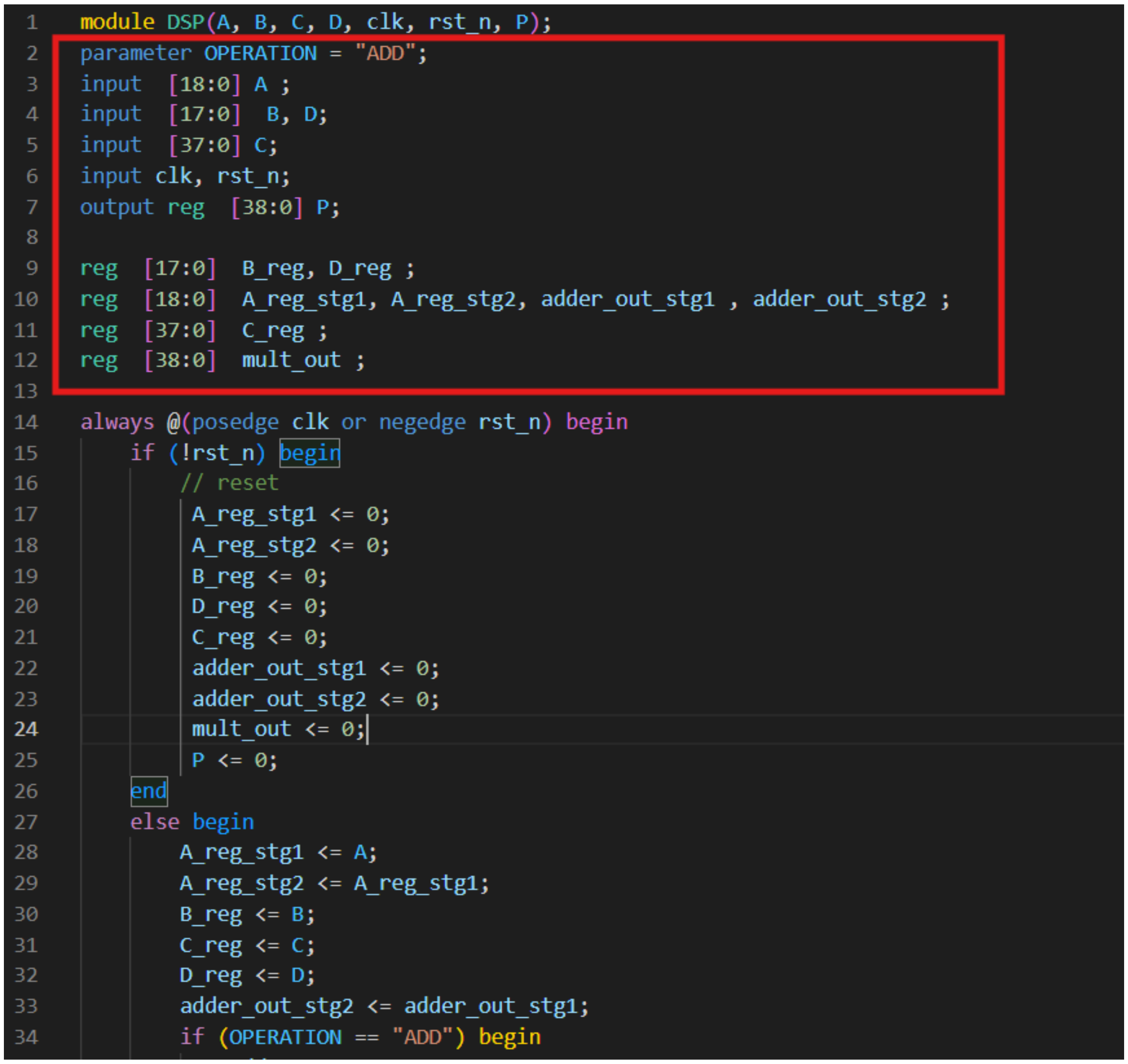
3- Toggle coverage



```
       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
         12                              Alu_out[2]           1           1      100.00
         12                              Alu_out[1]           1           1      100.00
         12                              Alu_out[0]       E-hit       E-hit      100.00

Total Node Count     =          21
Toggled Node Count   =          21
Untoggled Node Count =           0

Toggle Coverage      =      100.0% (42 of 42 bins)
```

4- DSP

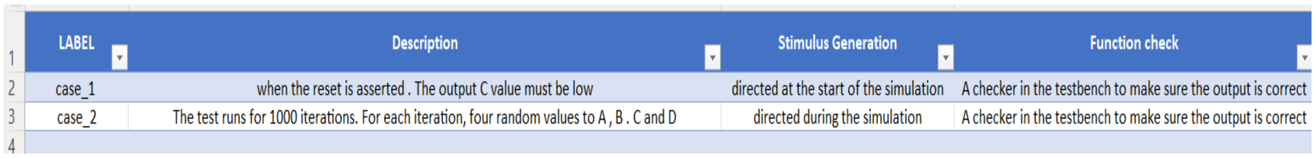
I. Design



```
module DSP(A, B, C, D, clk, rst_n, P);
parameter OPERATION = "ADD";
input  [18:0] A ;
input  [17:0]  B, D;
input  [37:0] C;
input clk, rst_n;
output reg  [38:0] P;

reg  [17:0]  B_reg, D_reg ;
reg  [18:0]  A_reg_stg1, A_reg_stg2, adder_out_stg1 , adder_out_stg2 ;
reg  [37:0]  C_reg ;
reg  [38:0]  mult_out ;

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        // reset
         A_reg_stg1 <= 0;
         A_reg_stg2 <= 0;
         B_reg <= 0;
         D_reg <= 0;
         C_reg <= 0;
         adder_out_stg1 <= 0;
         adder_out_stg2 <= 0;
         mult_out <= 0;
         P <= 0;
    end
    else begin
        A_reg_stg1 <= A;
        A_reg_stg2 <= A_reg_stg1;
        B_reg <= B;
        C_reg <= C;
        D_reg <= D;
        adder_out_stg2 <= adder_out_stg1;
        if (OPERATION == "ADD") begin
```

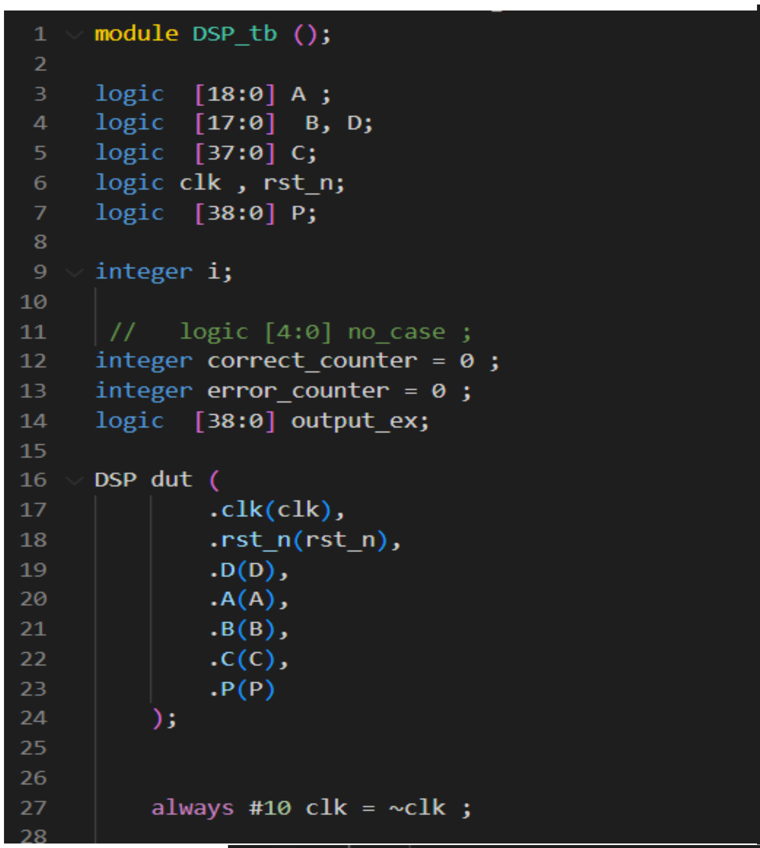
Change the width of A , P , A_reg_stg1, A_reg_stg2, adder_out_stg1 , adder_out_stg2 and mult_out

II. Verification plan

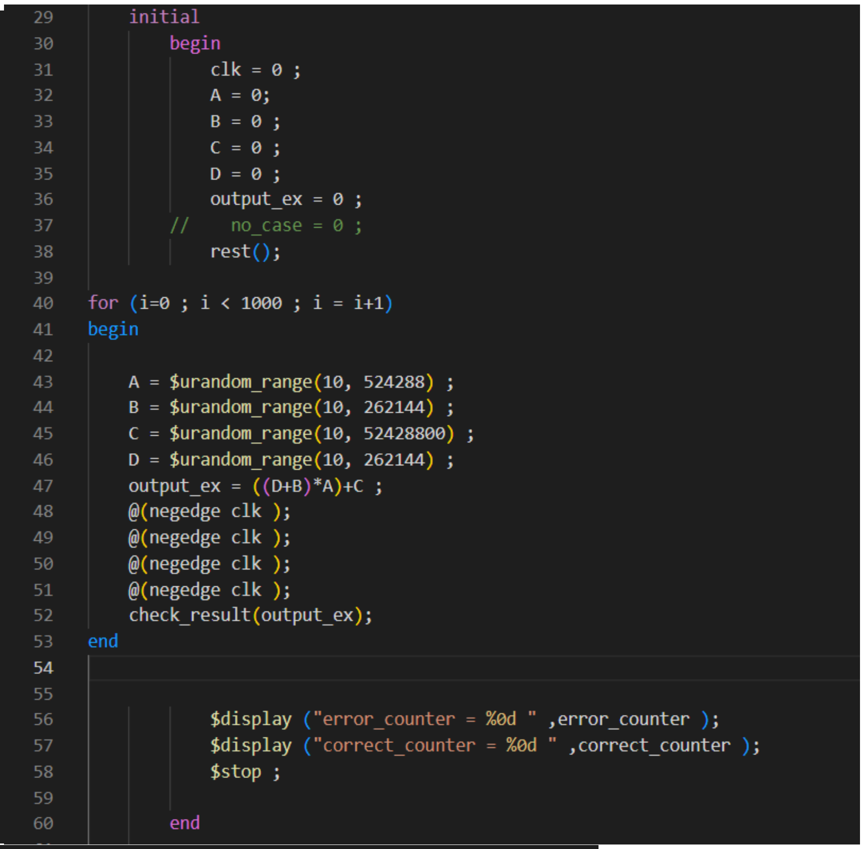


LABEL	Description	Stimulus Generation	Function check
case_1	when the reset is asserted . The output C value must be low	directed at the start of the simulation	A checker in the testbench to make sure the output is correct
case_2	The test runs for 1000 iterations. For each iteration, four random values to A , B . C and D	directed during the simulation	A checker in the testbench to make sure the output is correct

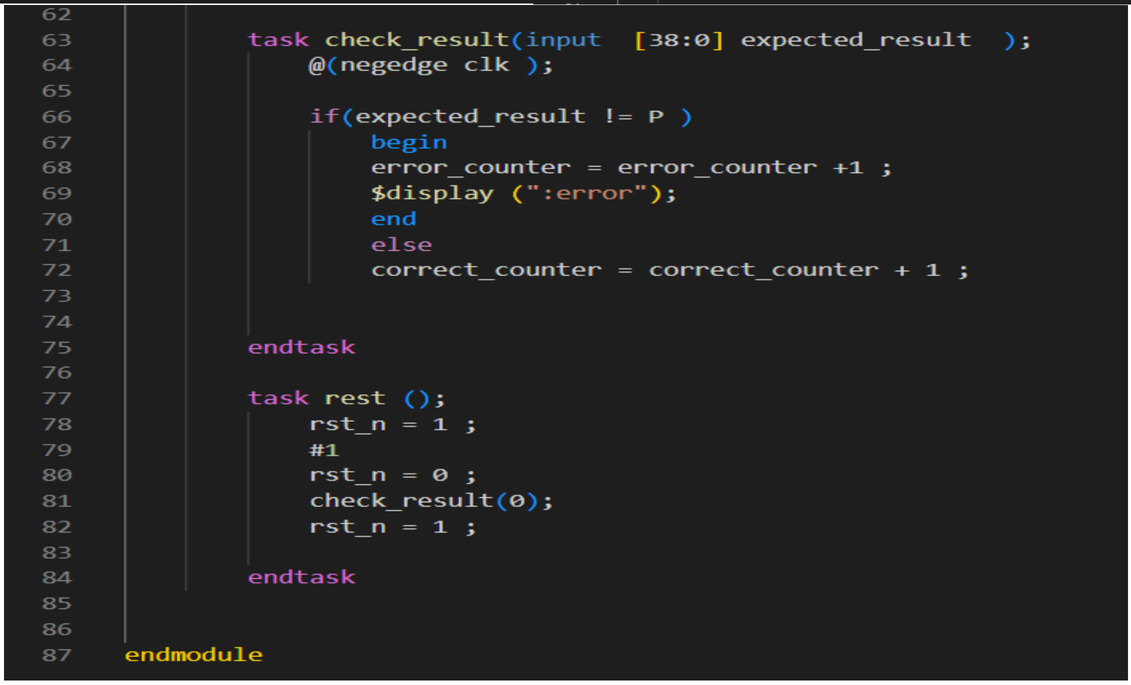
III. Testbench



```
module DSP_tb ();

logic  [18:0] A ;
logic  [17:0]  B, D;
logic  [37:0] C;
logic clk , rst_n;
logic  [38:0] P;

integer i;

 //   logic [4:0] no_case ;
integer correct_counter = 0 ;
integer error_counter = 0 ;
logic  [38:0] output_ex;

DSP dut (
        .clk(clk),
        .rst_n(rst_n),
        .D(D),
        .A(A),
        .B(B),
        .C(C),
        .P(P)
    );


    always #10 clk = ~clk ;
```

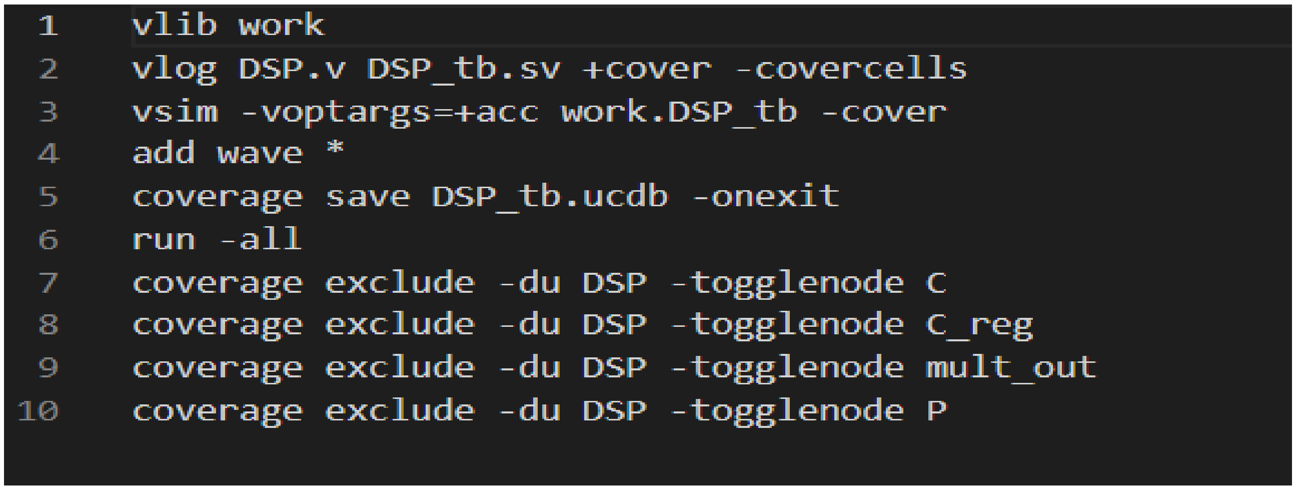


```
    initial
        begin
            clk = 0 ;
            A = 0;
            B = 0 ;
            C = 0 ;
            D = 0 ;
            output_ex = 0 ;
        //    no_case = 0 ;
            rest();

for (i=0 ; i < 1000 ; i = i+1)
begin

    A = $urandom_range(10, 524288) ;
    B = $urandom_range(10, 262144) ;
    C = $urandom_range(10, 52428800) ;
    D = $urandom_range(10, 262144) ;
    output_ex = ((D+B)*A)+C ;
    @(negedge clk );
    @(negedge clk );
    @(negedge clk );
    @(negedge clk );
    check_result(output_ex);
end


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

        end
```



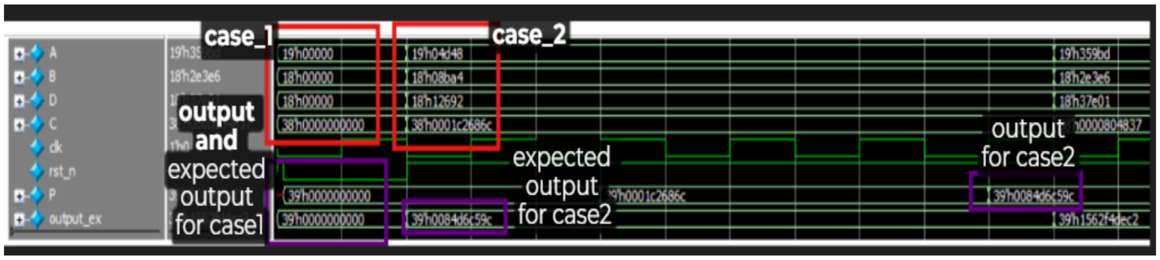
```
            task check_result(input  [38:0] expected_result  );
                @(negedge clk );

                if(expected_result != P )
                    begin
                    error_counter = error_counter +1 ;
                    $display (":error");
                    end
                    else
                    correct_counter = correct_counter + 1 ;


            endtask

            task rest ();
                rst_n = 1 ;
                #1
                rst_n = 0 ;
                check_result(0);
                rst_n = 1 ;

            endtask


    endmodule
```

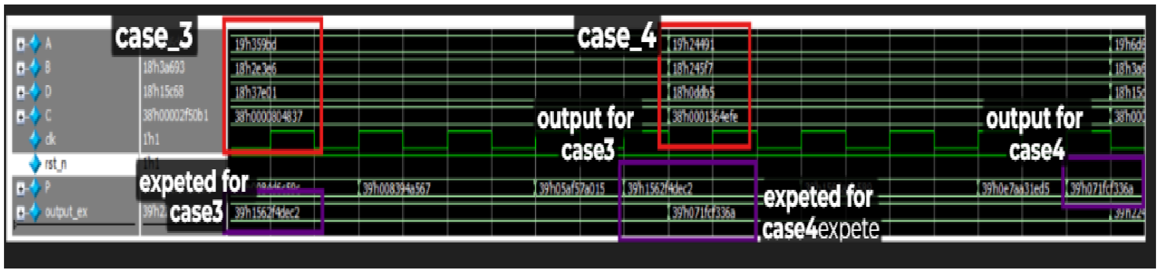
IV. Do file



```
vlib work
vlog DSP.v DSP_tb.sv +cover -covercells
vsim -voptargs=+acc work.DSP_tb -cover
add wave *
coverage save DSP_tb.ucdb -onexit
run -all
coverage exclude -du DSP -togglenode C
coverage exclude -du DSP -togglenode C_reg
coverage exclude -du DSP -togglenode mult_out
coverage exclude -du DSP -togglenode P
```

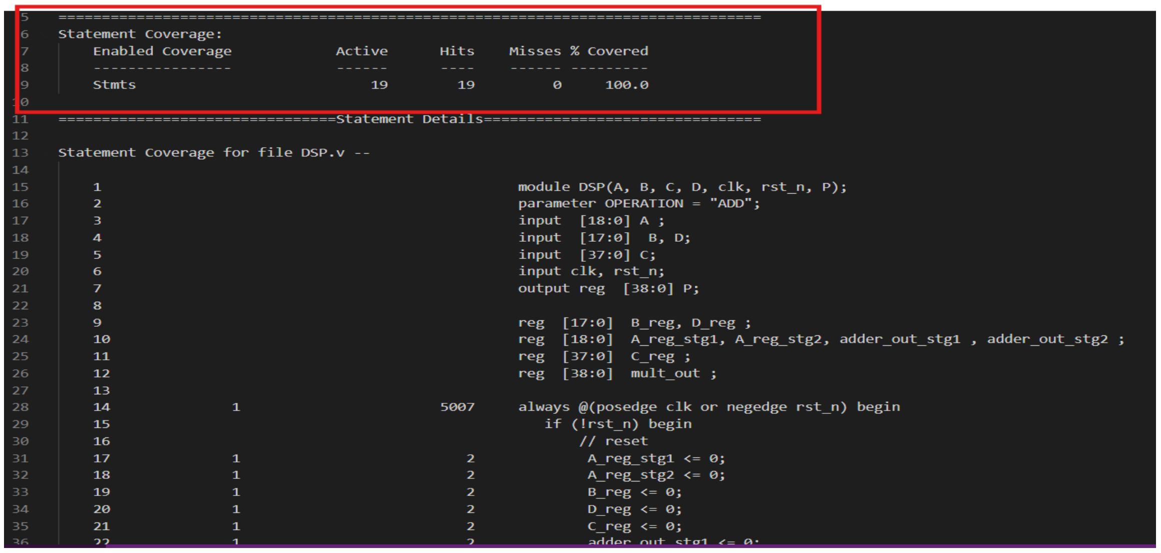
V. Waveform





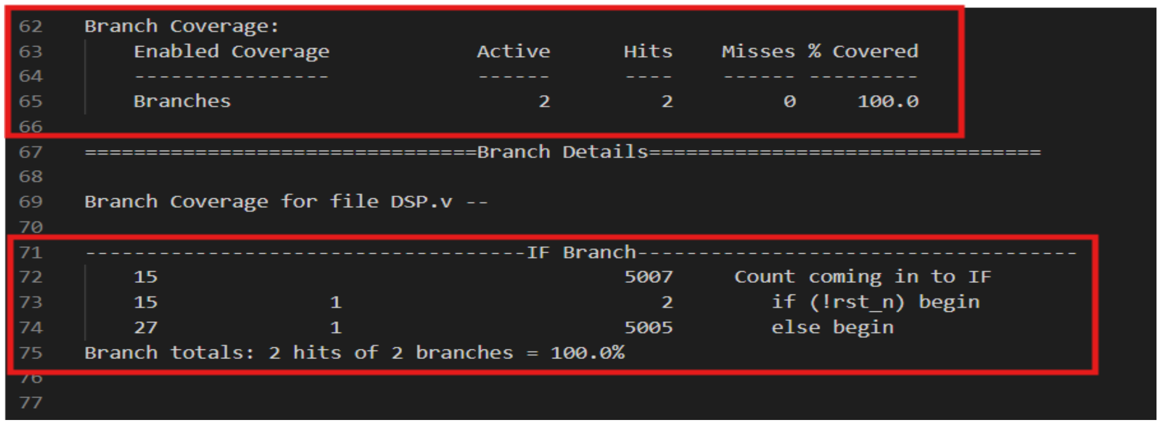
VI. Coverage

1- Branch



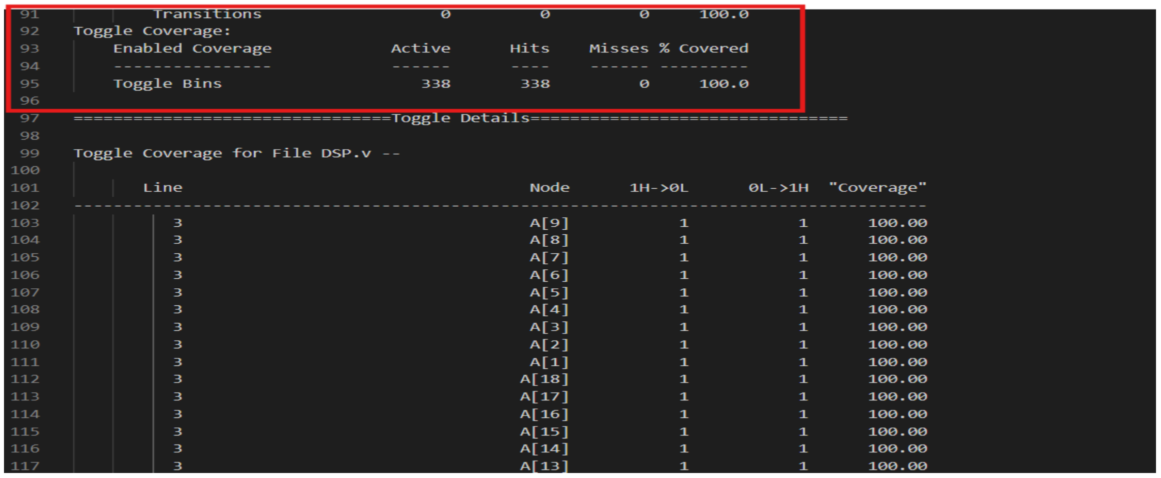
```
==============================================================================
Statement Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Stmts                             19        19         0     100.0

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

Statement Coverage for file DSP.v --

    1                                                module DSP(A, B, C, D, clk, rst_n, P);
    2                                                parameter OPERATION = "ADD";
    3                                                input  [18:0] A ;
    4                                                input  [17:0]  B, D;
    5                                                input  [37:0] C;
    6                                                input clk, rst_n;
    7                                                output reg  [38:0] P;
    8
    9                                                reg  [17:0]  B_reg, D_reg ;
    10                                               reg  [18:0]  A_reg_stg1, A_reg_stg2, adder_out_stg1 , adder_out_stg2 ;
    11                                               reg  [37:0]  C_reg ;
    12                                               reg  [38:0]  mult_out ;
    13
    14             1                        5007     always @(posedge clk or negedge rst_n) begin
    15                                                  if (!rst_n) begin
    16                                                      // reset
    17             1                           2             A_reg_stg1 <= 0;
    18             1                           2             A_reg_stg2 <= 0;
    19             1                           2             B_reg <= 0;
    20             1                           2             D_reg <= 0;
    21             1                           2             C_reg <= 0;
    22             1                           2             adder_out_stg1 <= 0;
```

2- Statement



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

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

Branch Coverage for file DSP.v --

------------------------------------IF Branch------------------------------------
    15                                      5007     Count coming in to IF
    15              1                          2        if (!rst_n) begin
    27              1                       5005        else begin
Branch totals: 2 hits of 2 branches = 100.0%
```

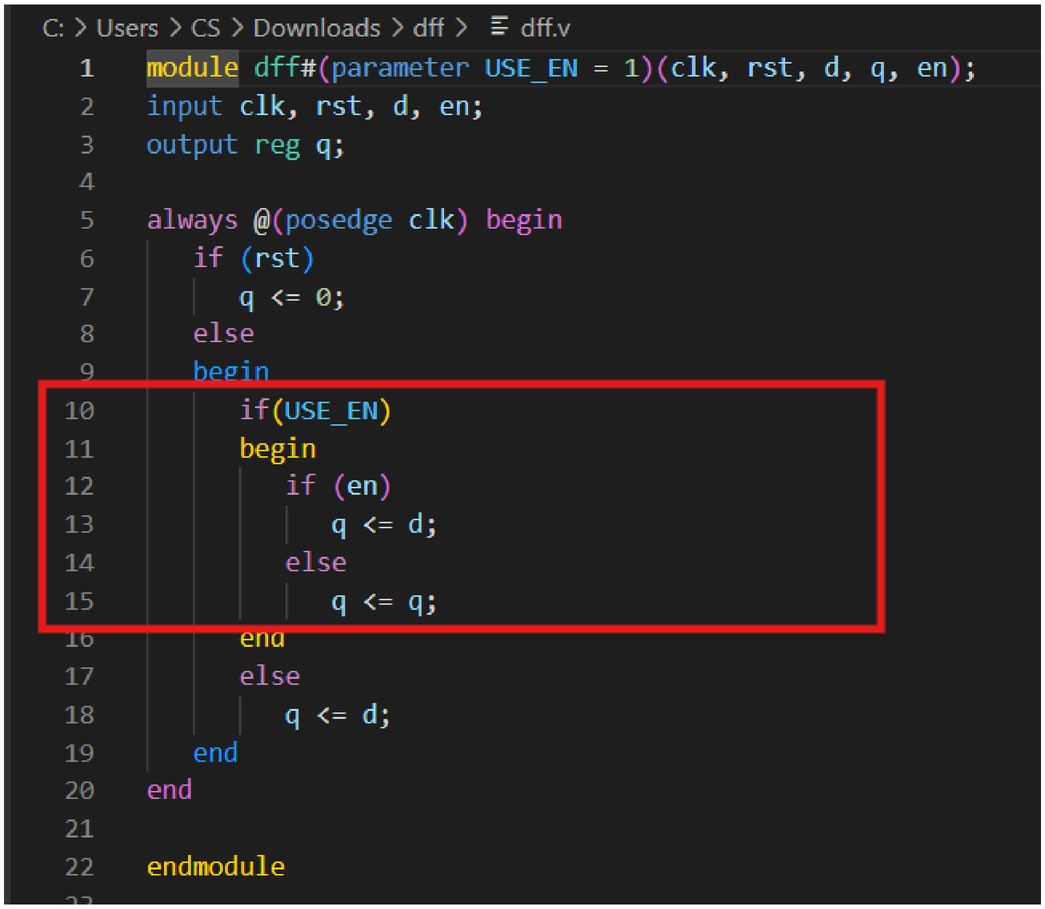
3- Toggle coverage



```
            Transitions                    0           0           0     100.0
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                      338       338         0     100.0

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

Toggle Coverage for File DSP.v --

          Line                                   Node      1H->0L      0L->1H  "Coverage"
-----------------------------------------------------------------------------------------
             3                                   A[9]           1           1      100.00
             3                                   A[8]           1           1      100.00
             3                                   A[7]           1           1      100.00
             3                                   A[6]           1           1      100.00
             3                                   A[5]           1           1      100.00
             3                                   A[4]           1           1      100.00
             3                                   A[3]           1           1      100.00
             3                                   A[2]           1           1      100.00
             3                                   A[1]           1           1      100.00
             3                                  A[18]           1           1      100.00
             3                                  A[17]           1           1      100.00
             3                                  A[16]           1           1      100.00
             3                                  A[15]           1           1      100.00
             3                                  A[14]           1           1      100.00
             3                                  A[13]           1           1      100.00
```

5- D_FF

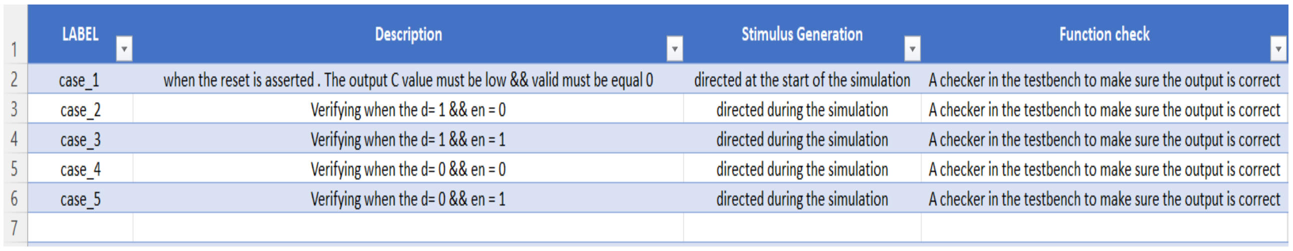
I. Design



C: > Users > CS > Downloads > dff > dff.v

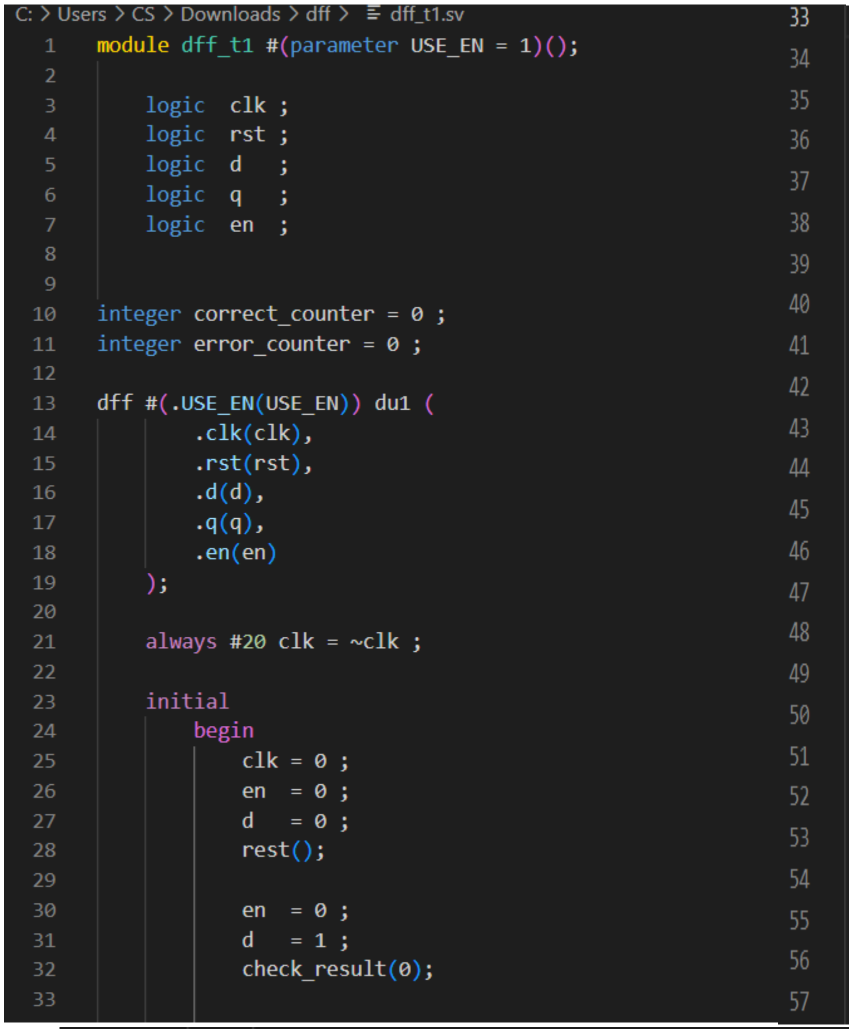
```verilog
module dff#(parameter USE_EN = 1)(clk, rst, d, q, en);
input clk, rst, d, en;
output reg q;

always @(posedge clk) begin
    if (rst)
        q <= 0;
    else
    begin
        if(USE_EN)
        begin
            if (en)
                q <= d;
            else
                q <= q;
        end
        else
            q <= d;
    end
end

endmodule
```

II. Verification plan for the two testbench



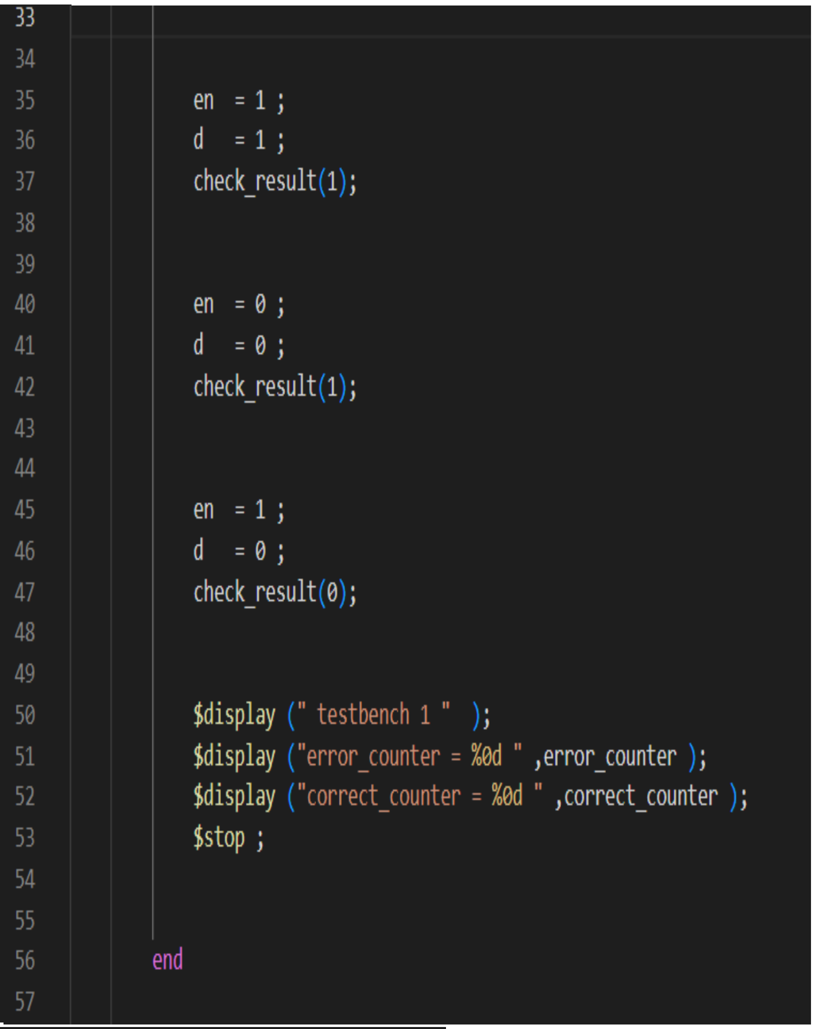
LABEL	Description	Stimulus Generation	Function check
case_1	when the reset is asserted . The output C value must be low && valid must be equal 0	directed at the start of the simulation	A checker in the testbench to make sure the output is correct
case_2	Verifying when the d= 1 && en = 0	directed during the simulation	A checker in the testbench to make sure the output is correct
case_3	Verifying when the d= 1 && en = 1	directed during the simulation	A checker in the testbench to make sure the output is correct
case_4	Verifying when the d= 0 && en = 0	directed during the simulation	A checker in the testbench to make sure the output is correct
case_5	Verifying when the d= 0 && en = 1	directed during the simulation	A checker in the testbench to make sure the output is correct

III. Testbench1

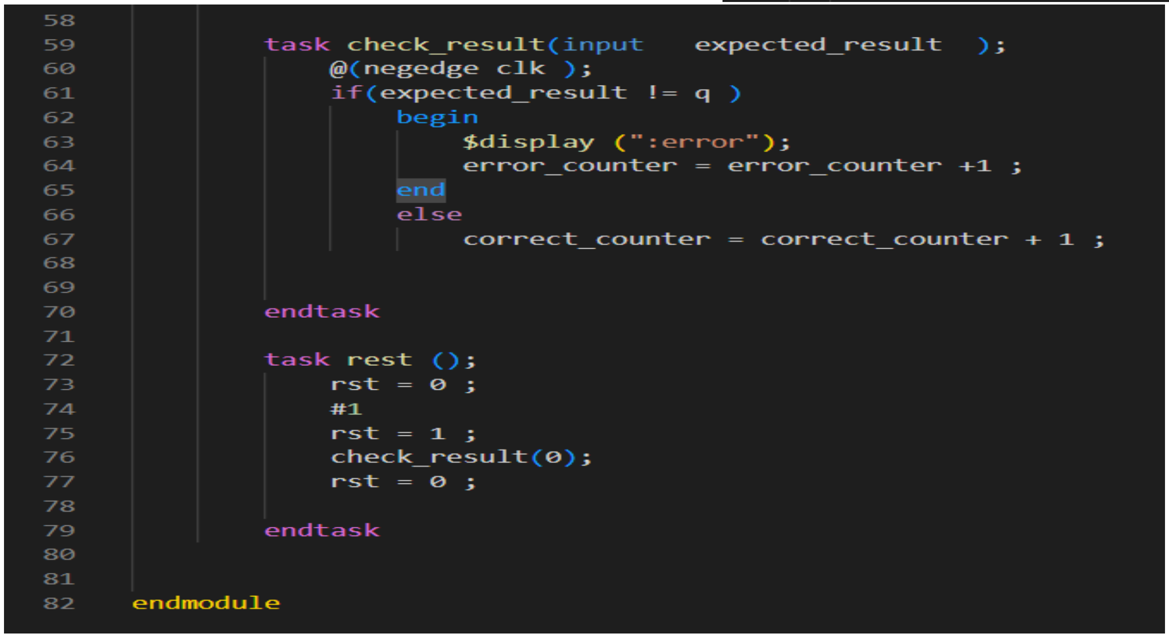


C: > Users > CS > Downloads > dff > dff_t1.sv

```
module dff_t1 #(parameter USE_EN = 1)();

    logic  clk ;
    logic  rst ;
    logic  d   ;
    logic  q   ;
    logic  en  ;


integer correct_counter = 0 ;
integer error_counter = 0 ;

dff #(.USE_EN(USE_EN)) du1 (
        .clk(clk),
        .rst(rst),
        .d(d),
        .q(q),
        .en(en)
    );

    always #20 clk = ~clk ;

    initial
        begin
            clk = 0 ;
            en  = 0 ;
            d   = 0 ;
            rest();

            en  = 0 ;
            d   = 1 ;
            check_result(0);

```

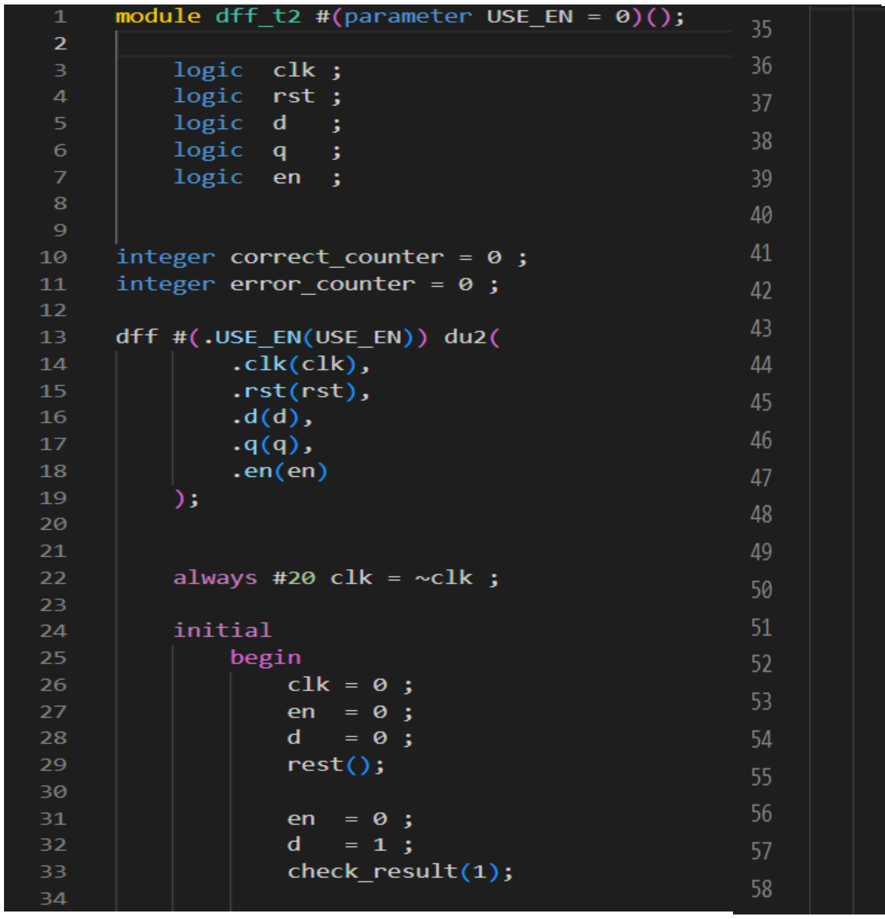


```


            en  = 1 ;
            d   = 1 ;
            check_result(1);


            en  = 0 ;
            d   = 0 ;
            check_result(1);


            en  = 1 ;
            d   = 0 ;
            check_result(0);


            $display (" testbench 1 "  );
            $display ("error_counter = %0d " ,error_counter );
            $display ("correct_counter = %0d " ,correct_counter );
            $stop ;


        end

```

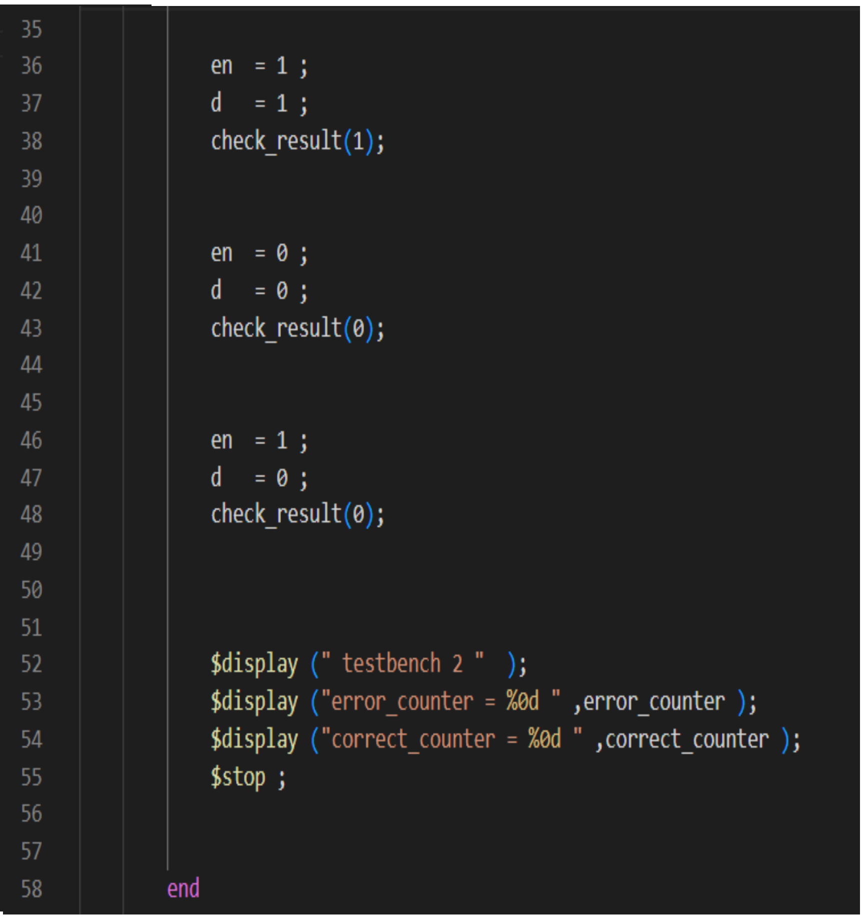


```

        task check_result(input   expected_result   );
            @(negedge clk );
            if(expected_result != q )
                begin
                    $display (":error");
                    error_counter = error_counter +1 ;
                end
                else
                    correct_counter = correct_counter + 1 ;


        endtask

        task rest ();
            rst = 0 ;
            #1
            rst = 1 ;
            check_result(0);
            rst = 0 ;

        endtask


endmodule
```

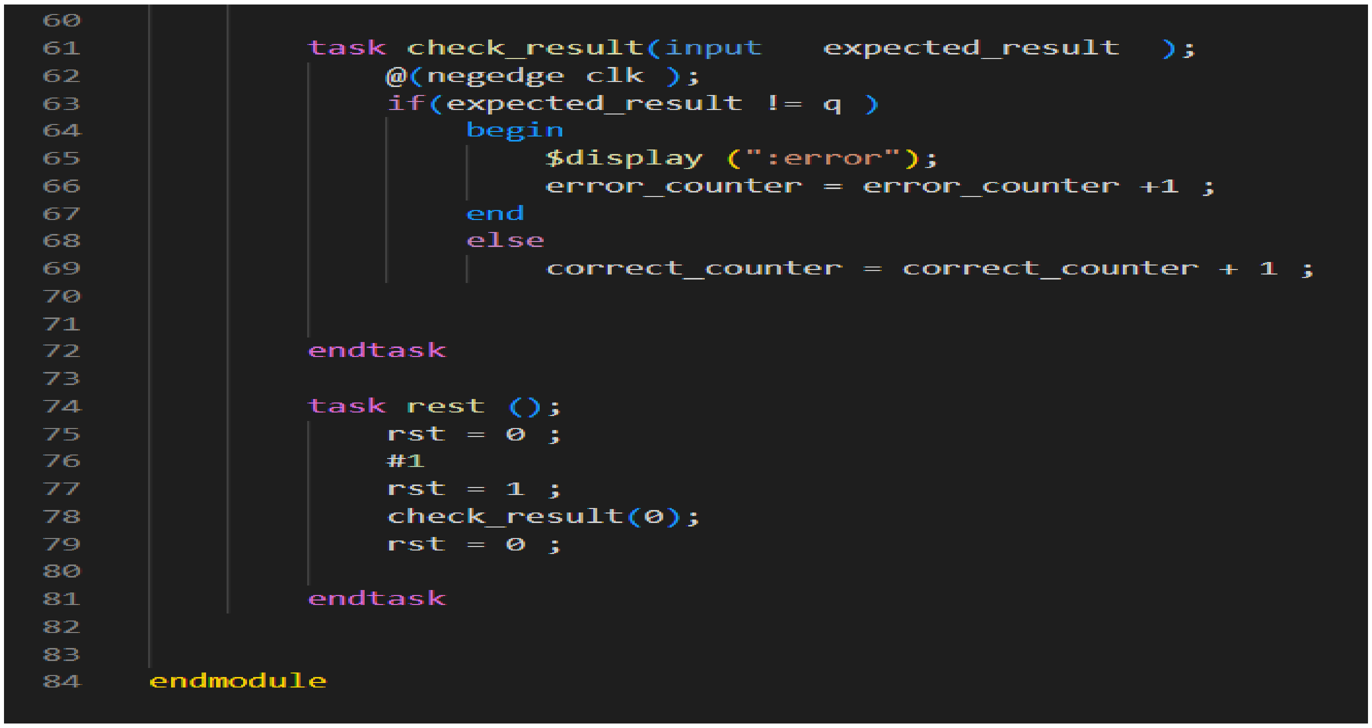
IV. Testbench2



```
module dff_t2 #(parameter USE_EN = 0)();

    logic  clk ;
    logic  rst ;
    logic  d   ;
    logic  q   ;
    logic  en  ;


integer correct_counter = 0 ;
integer error_counter = 0 ;

dff #(.USE_EN(USE_EN)) du2(
        .clk(clk),
        .rst(rst),
        .d(d),
        .q(q),
        .en(en)
    );


    always #20 clk = ~clk ;

    initial
        begin
            clk = 0 ;
            en  = 0 ;
            d   = 0 ;
            rest();

            en  = 0 ;
            d   = 1 ;
            check_result(1);

```

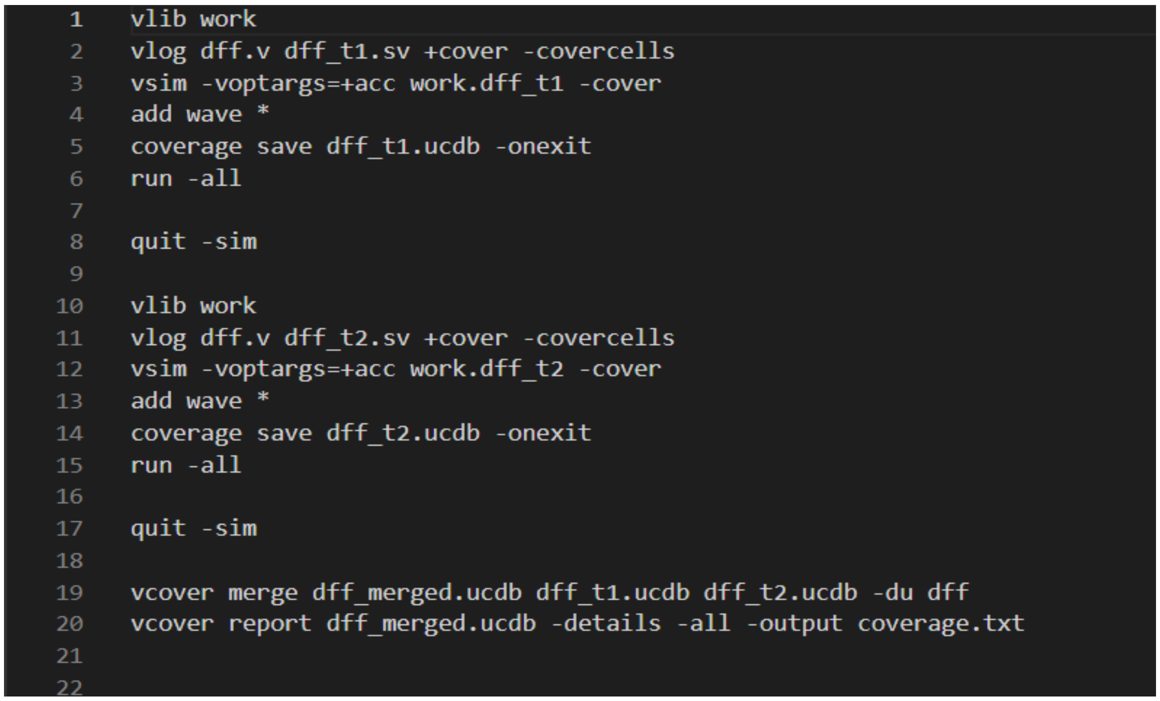


```

            en  = 1 ;
            d   = 1 ;
            check_result(1);


            en  = 0 ;
            d   = 0 ;
            check_result(0);


            en  = 1 ;
            d   = 0 ;
            check_result(0);



            $display (" testbench 2 "  );
            $display ("error_counter = %0d " ,error_counter );
            $display ("correct_counter = %0d " ,correct_counter );
            $stop ;


        end
```



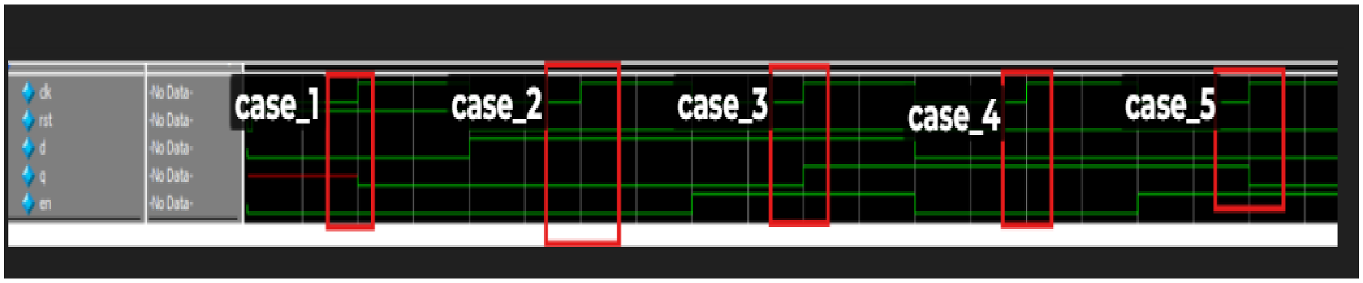
```

        task check_result(input   expected_result  );
            @(negedge clk );
            if(expected_result != q )
                begin
                    $display (":error");
                    error_counter = error_counter +1 ;
                end
                else
                    correct_counter = correct_counter + 1 ;


        endtask

        task rest ();
            rst = 0 ;
            #1
            rst = 1 ;
            check_result(0);
            rst = 0 ;

        endtask


endmodule
```

V. Do file

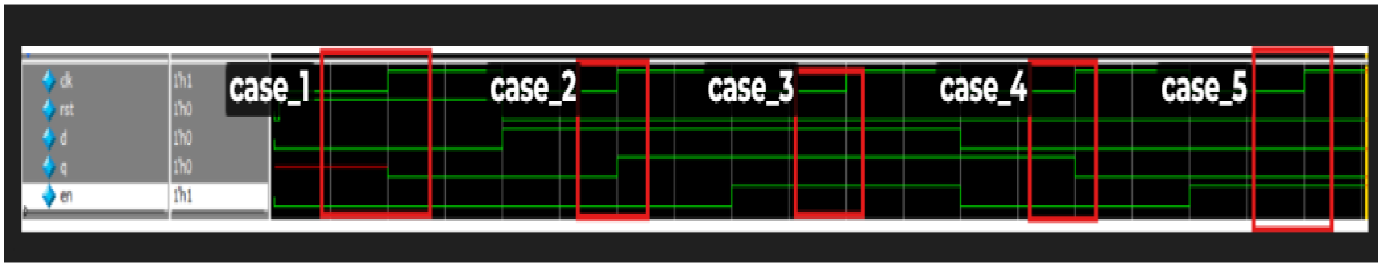


```
vlib work
vlog dff.v dff_t1.sv +cover -covercells
vsim -voptargs=+acc work.dff_t1 -cover
add wave *
coverage save dff_t1.ucdb -onexit
run -all

quit -sim

vlib work
vlog dff.v dff_t2.sv +cover -covercells
vsim -voptargs=+acc work.dff_t2 -cover
add wave *
coverage save dff_t2.ucdb -onexit
run -all

quit -sim

vcover merge dff_merged.ucdb dff_t1.ucdb dff_t2.ucdb -du dff
vcover report dff_merged.ucdb -details -all -output coverage.txt

```

VI. Waveform for testbench1

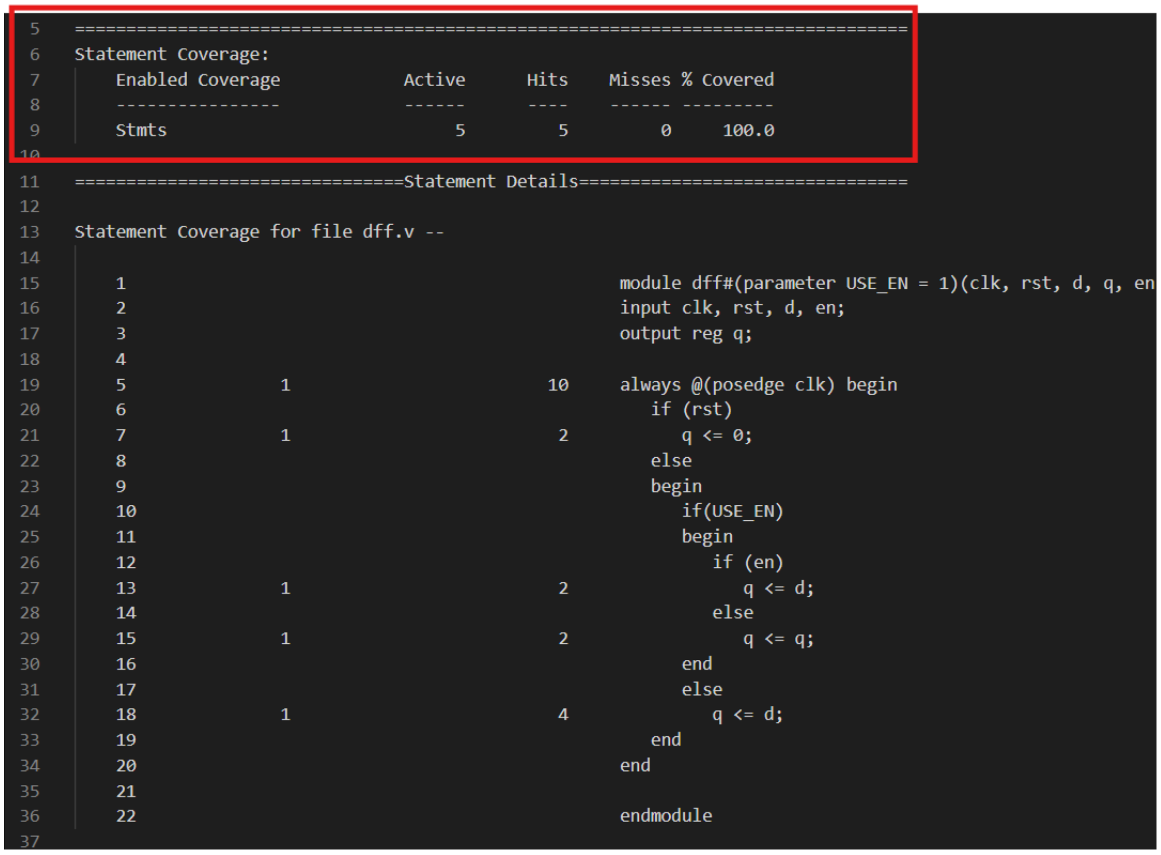


VII. Waveform for testbench2



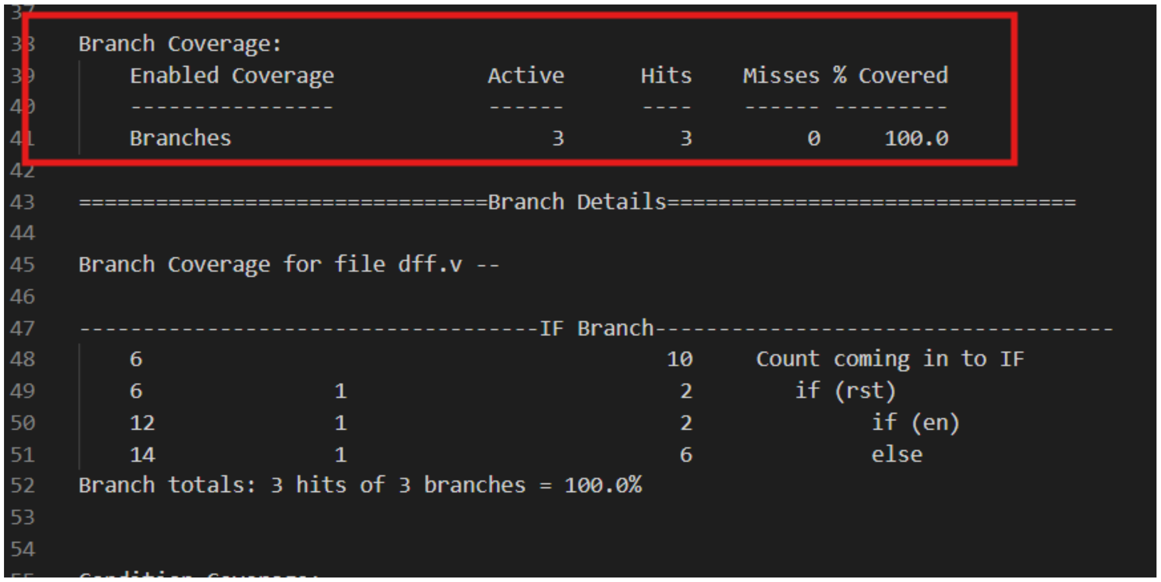
VIII. Total Coverage

1- Branch



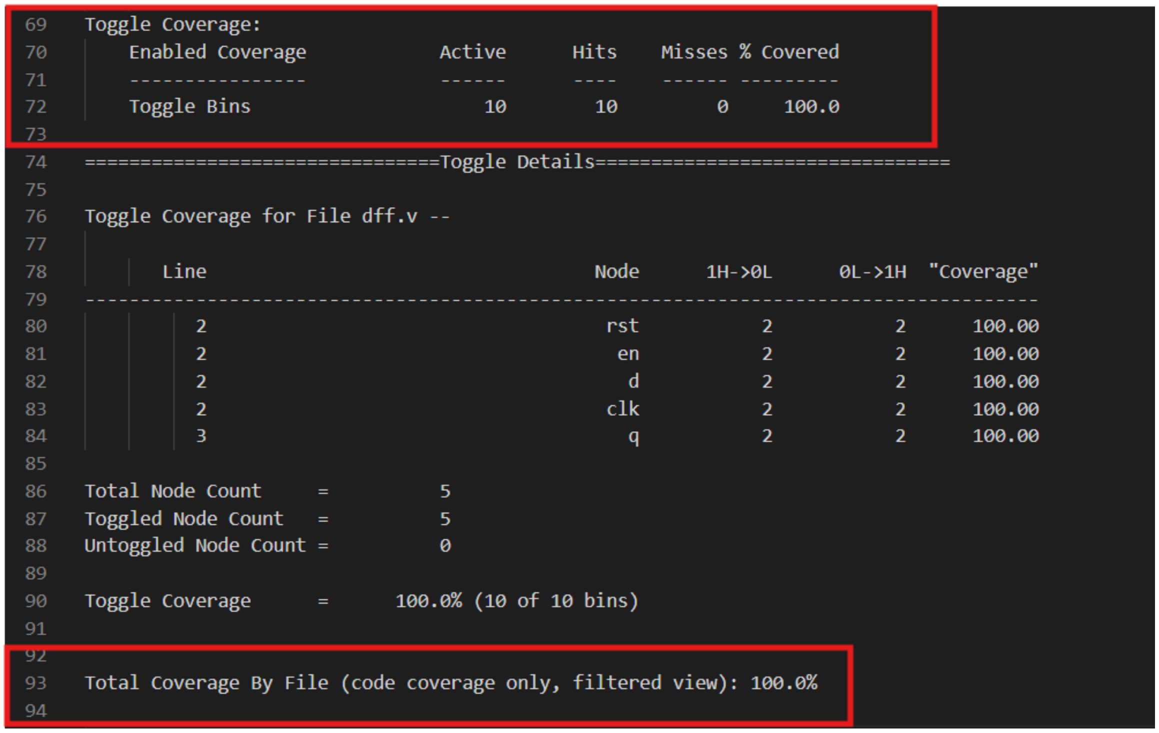
```
=================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            5         5         0     100.0

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

Statement Coverage for file dff.v --

    1                                                    module dff#(parameter USE_EN = 1)(clk, rst, d, q, en
    2                                                    input clk, rst, d, en;
    3                                                    output reg q;
    4
    5               1                           10       always @(posedge clk) begin
    6                                                       if (rst)
    7               1                            2             q <= 0;
    8                                                       else
    9                                                       begin
    10                                                         if(USE_EN)
    11                                                         begin
    12                                                            if (en)
    13              1                            2                   q <= d;
    14                                                            else
    15              1                            2                   q <= q;
    16                                                         end
    17                                                         else
    18              1                            4                q <= d;
    19                                                      end
    20                                                   end
    21
    22                                                   endmodule
```

2- Statement



```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         3         3         0     100.0

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

Branch Coverage for file dff.v --

------------------------------------IF Branch------------------------------------
    6                                   10     Count coming in to IF
    6               1                    2        if (rst)
    12              1                    2              if (en)
    14              1                    6              else
Branch totals: 3 hits of 3 branches = 100.0%
```

3- Toggle coverage



```
Toggle Coverage:
    Enabled Coverage              Active      Hits    Misses % Covered
    ----------------              ------      ----    ------ ---------
    Toggle Bins                       10        10         0     100.0

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

Toggle Coverage for File dff.v --

       Line                                       Node      1H->0L      0L->1H  "Coverage"
---------------------------------------------------------------------------------------
          2                                        rst           2           2      100.00
          2                                         en           2           2      100.00
          2                                          d           2           2      100.00
          2                                        clk           2           2      100.00
          3                                          q           2           2      100.00

Total Node Count     =          5
Toggled Node Count   =          5
Untoggled Node Count =          0

Toggle Coverage      =      100.0% (10 of 10 bins)


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