Digital Verification using SV and UVM
Assignment-1

Name: Fares Khalaf Sultan

Q1)

Before modification:

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        29         1    96.66%

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

Toggle Coverage for instance /\TB#a1  --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                        A[0-3]           1           1      100.00
                                        B[0-3]           1           1      100.00
                                        C[4-0]           1           1      100.00
                                           clk           1           1      100.00
                                         reset           1           0       50.00

Total Node Count      =         15
Toggled Node Count    =         14
Untoggled Node Count =          1

Toggle Coverage       =      96.66% (29 of 30 bins)


Total Coverage By Instance (filtered view): 98.88%
```

Issue: reset signal didn't change from 0 to 1 during simulation.

Fix:

```
    reset = 1;
    check_result(0);

    $display("Error Count = %0d\n",Error_Count);
    $display("Correct Count = %0d\n",Correct_Count);

    $stop;
end
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        30         0   100.00%

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

Toggle Coverage for instance /\TB#a1  --

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

Total Node Count      =         15
Toggled Node Count    =         15
Untoggled Node Count =          0

Toggle Coverage       =     100.00% (30 of 30 bins)


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

Q2) Priority Encoder:

❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
ENC_1	at posedge clk, if rst is asserted all outputs should be low	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
ENC_2	When D = ‘b0000, Valid should be low	Directed during the simulation	-	A Checker in the testbench to make sure the output is correct
ENC_3	exhaustively verify all possible inputs combination.	Directed during the simulation	-	A checker in the testbench to make sure the output is correct
ENC_4	Some values of the inputs to complete the toggle coverage	Directed at the end of the simulation	-	A checker in the testbench to make sure the output is correct

❖ TestBench:

```
module enc_tb ();

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

    integer Error_Count, Correct_Count;

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

    initial begin
        clk = 0;

        forever begin
            #1 clk = ~clk;
        end
    end


    initial begin

    Error_Count = 0;
    Correct_Count = 0;
    D = 'b0001;

    //ENC_1
    assert_reset;
```

```
    // ENC_2
    D = 'b0000;
    @(negedge clk);
    if (valid == 'b0) begin
        $display("Passed for D = 'b%0b",D);
        Correct_Count++;
    end
    else begin
        $display("Passed for D = 'b%0b",D);
        Error_Count++;
    end

    //ENC_3
    D = 'b0001;
    check_result('b11,'b1);

    D = 'b0010;
    check_result('b10,'b1);

    D = 'b0011;
    check_result('b11,'b1);

    D = 'b0100;
    check_result('b01,'b1);

    D = 'b0101;
    check_result('b11,'b1);

    D = 'b0110;
    check_result('b10,'b1);

    D = 'b0111;
    check_result('b11,'b1);

    D = 'b1000;
    check_result('b00,'b1);

    D = 'b1001;
    check_result('b11,'b1);

    D = 'b1010;
    check_result('b10,'b1);

    D = 'b1011;
    check_result('b11,'b1);

    D = 'b1100;
    check_result('b01,'b1);

    D = 'b1101;
    check_result('b11,'b1);

    D = 'b1110;
    check_result('b10,'b1);

    D = 'b1111;
    check_result('b11,'b1);

    //ENC_4

    assert_reset; // rst : L->H & valid: H:L

    D = 'b0001;    // D[3]: H->L
    check_result('b11,'b1);

    $display("Error Count: %0d", Error_Count);
    $display("Correct Count: %0d", Correct_Count);

    $stop;
    end
```

```
102         task assert_reset ;
103
104         rst = 1;
105         @(negedge clk);
106
107         if ((Y != 'b0)|| (valid != 'b0))
108          begin
109           $display("Reset failed");
110           Error_Count++;
111         end
112         else begin
113          $display("Reset Passed");
114          Correct_Count++;
115         end
116         rst = 0;
117
118     endtask
119
120     task check_result(logic [1:0] expected_Y, logic excpected_valid);
121
122         @(negedge clk);
123
124         if ((Y == expected_Y) && (valid == excpected_valid)) begin
125             $display("Passed for D = 'b%0b",D);
126             Correct_Count++;
127         end
128         else begin
129             $display("Passed for D = 'b%0b",D);
130              Error_Count++;
131         end
132     endtask
133
134     endmodule
```

❖ Bugs found:

Label	Bug Description	Wrong code	Correction
	Output (Y) was defined as wire, not reg.	`output [1:0] Y,`	Defined Y as output reg

ENC1	rst signal only resets the value of Y but does nothing to valid	`if (rst)`
		`Y <= 2'b0;`
		`else`
			(rst) signal must reset the value of valid too, and valid signal should take its value inside the else statement, to ensure the priority of rst signal

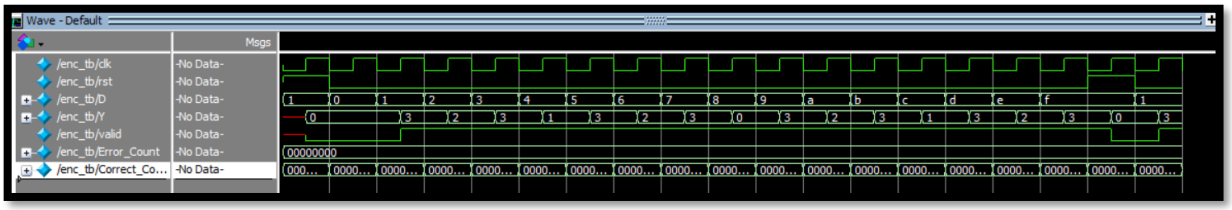
❖ **Fixed design:**

```
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   always @(posedge clk) begin
10    if (rst) begin
11      Y <= 2'b0;
12      valid <= 1'b0;
13    end
14    else begin
15      casex (D)
16          4'b1000: Y <= 0;
17          4'bX100: Y <= 1;
18          4'bXX10: Y <= 2;
19          4'bXXX1: Y <= 3;
20      endcase
21
22      valid <= (~|D)? 1'b0: 1'b1;
23    end
24  end
25  endmodule
26
```

❖ Result:

```
# vsim -voptargs="+acc" work.enc_tb -coverage
# Start time: 01:16:37 on Mar 06,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading sv_std.std
# Loading work.enc_tb(fast)
# Loading work.priority_enc(fast)
# Reset Passed
# Passed for D = 'b0
# Passed for D = 'b1
# Passed for D = 'b10
# Passed for D = 'b11
# Passed for D = 'b100
# Passed for D = 'b101
# Passed for D = 'b110
# Passed for D = 'b111
# Passed for D = 'b1000
# Passed for D = 'b1001
# Passed for D = 'b1010
# Passed for D = 'b1011
# Passed for D = 'b1100
# Passed for D = 'b1101
# Passed for D = 'b1110
# Passed for D = 'b1111
# Reset Passed
# Passed for D = 'b1
# Error Count: 0
# Correct Count: 19
# ** Note: $stop    : priority_enc_TB.sv(99)
#    Time: 38 ns  Iteration: 1  Instance: /enc_tb
# Break in Module enc_tb at priority_enc_TB.sv line 99
```

Waveform:



❖ Coverage Report:

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         7         0   100.00%

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

Branch Coverage for instance /\enc_tb#DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File priority_enc.v
-----------------------------------IF Branch-----------------------------------
    10                                        19     Count coming in to IF
    10             1                           2       if (rst) begin

    14             1                          17       else begin

Branch totals: 2 hits of 2 branches = 100.00%

---------------------------------CASE Branch---------------------------------
    15                                        17     Count coming in to CASE
    16             1                           1            4'b1000: Y <= 0;

    17             1                           2            4'bX100: Y <= 1;

    18             1                           4            4'bXX10: Y <= 2;

    19             1                           9            4'bXXX1: Y <= 3;

                                               1     All False Count
Branch totals: 5 hits of 5 branches = 100.00%
```

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

================================Statement Details================================
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         18        18         0   100.00%

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

Toggle Coverage for instance /\enc_tb#DUT  --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                       D[0-3]           1           1      100.00
                                       Y[1-0]           1           1      100.00
                                          clk           1           1      100.00
                                          rst           1           1      100.00
                                        valid           1           1      100.00

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

Toggle Coverage       =     100.00% (18 of 18 bins)


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

Q3) ALU

❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
ALU1	when reset is asserted, Output (C) should be Low	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
ALU2	Hitting the bounary cases of A and B and try addition then subtraction	Directed during the simulation	-	A checker in the testbench to make sure the output is correct
ALU3	verify bitwise invert input A	Random during the simulation	-	A checker in the testbench to make sure the output is correct
ALU4	verify reduction OR input B	Random during the simulation	-	A checker in the testbench to make sure the output is correct
ALU5	overall functionality of the ALU by select two values for A,B and the apply all opcodes on them	Random during the simulation	-	A checker in the testbench to make sure the output is correct

❖ TestBench:

```
1   module ALU_tb ();
2
3   logic signed [3:0] A,B ;
4   logic clk, reset;
5   logic [1:0] Opcode;
6   logic signed [4:0] C;
7
8   localparam MAXPOS = 7, MAXNEG = -8 ;
9   localparam          Add            = 2'b00; // A + B
10  localparam          Sub            = 2'b01; // A - B
11  localparam          Not_A          = 2'b10; // ~A
12  localparam          ReductionOR_B  = 2'b11; // |B
13
14  integer Error_Count, Correct_Count;
15
16  ALU_4_bit DUT (.*);
17
18  initial begin
19      clk = 0;
20      forever begin
21          #1 clk = ~clk;
22      end
23  end
24
25  initial begin
26      Error_Count = 0;
27      Correct_Count = 0;
28      A = 1;
29      B = 1;
30      Opcode = Add;
31
32  //ALU1
33      assert_reset;
34
```

```
// ALU2
    A = MAXNEG;B = MAXNEG; Opcode = Add;
    check_Add_result(-16);
    Opcode = Sub;
    check_Sub_result(0);

    A = MAXNEG;B = 0; Opcode = Add;
    check_Add_result(-8);
    Opcode = Sub;
    check_Sub_result(-8);

    A = MAXNEG;B = MAXPOS; Opcode = Add;
    check_Add_result(-1);
    Opcode = Sub;
    check_Sub_result(-15);

    A = 0;B = MAXPOS; Opcode = Add;
    check_Add_result(7);
    Opcode = Sub;
    check_Sub_result(-7);

    A = MAXPOS;B = MAXPOS; Opcode = Add;
    check_Add_result(14);
    Opcode = Sub;
    check_Sub_result(0);

    A = MAXPOS;B = 0; Opcode = Add;
    check_Add_result(7);
    Opcode = Sub;
    check_Sub_result(7);

    A = 0;B = 0; Opcode = Add;
    check_Add_result(0);
    Opcode = Sub;
    check_Sub_result(0);

    A = MAXNEG;B = 0;Opcode = Add;
    check_Add_result(-8);
    Opcode = Sub;
    check_Sub_result(-8);

```

```
//ALU3
    Opcode = Not_A;
    repeat(10) begin
        A = $random;
        check_Ainvert_result;
    end

//ALU4
    Opcode = ReductionOR_B;
    repeat(10) begin
        B = $random;
        check_RedOR_B_result;
    end

//ALU5

    A = 6; B = -3;
    Opcode = Add; check_Add_result(3);
    Opcode = Sub; check_Sub_result(9);
    Opcode = Not_A; check_Ainvert_result;
    Opcode = ReductionOR_B; check_RedOR_B_result;


assert_reset; // to complete toggle coverage

    $display("Error Count = %0d\n",Error_Count);
    $display("Correct Count = %0d\n",Correct_Count);

    $stop;
end
```

```
108     task assert_reset ;
109
110         reset = 1;
111         @(negedge clk);
112         if (C != 'b0)
113          begin
114           $display("Reset failed");
115           Error_Count++;
116         end
117         else begin
118          $display("Reset Passed");
119          Correct_Count++;
120         end
121         reset = 0;
122
123     endtask //
```

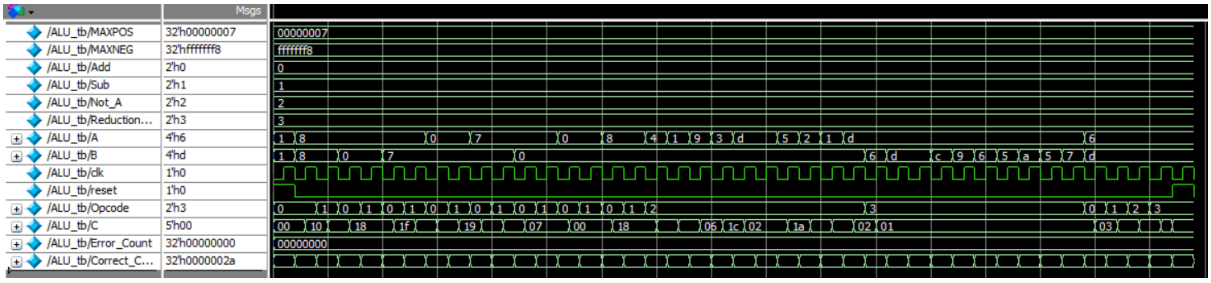
```
125     task check_Add_result(logic signed [4:0] expected_value);
126
127         @(negedge clk);
128
129         if (A + B == expected_value) begin
130             $display("Passed Adding for A = %0d, B = %0d excpected output: %0d",A,B,expected_value);
131             Correct_Count++;
132         end
133         else begin
134             $display("Failed Adding for A = %0d, B = %0d excpected output: %0d",A,B,expected_value);
135              Error_Count++;
136         end
137     endtask
138
139     task check_Sub_result(logic signed [4:0] expected_value);
140
141         @(negedge clk);
142
143         if (A - B == expected_value) begin
144             $display("Passed Subtracting for A = %0d, B = %0d excpected output: %0d",A,B,expected_value);
145             Correct_Count++;
146         end
147         else begin
148             $display("Failed Subtracting for A = %0d, B = %0d excpected output: %0d",A,B,expected_value);
149              Error_Count++;
150         end
151     endtask
152
```

```
153     task check_Ainvert_result;
154
155         @(negedge clk);
156
157         if (C == ~A) begin
158             $display("Passed Op1 Inverting for A = %0b,output: %0b",A,C);
159             Correct_Count++;
160         end
161         else begin
162             $display("Failed Op1 Inverting for A = %0b,output: %0b",A,C);
163              Error_Count++;
164         end
165     endtask
166
167     task check_RedOR_B_result;
168
169         @(negedge clk);
170
171         if (C == |B) begin
172             $display("Passed Op2 Reduction OR for B = %0b,output: %0b",B,C);
173             Correct_Count++;
174         end
175         else begin
176             $display("Failed Op2 Reduction OR for B = %0b,output: %0b",B,C);
177              Error_Count++;
178         end
179     endtask
180
181     endmodule
```

❖ Results:

```
# vsim -voptargs="+acc" ALU_tb -coverage
# Start time: 01:07:31 on Mar 06,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt1
# Loading sv_std.std
# Loading work.ALU_tb(fast)
# Loading work.ALU_4_bit(fast)
# Reset Passed
# Passed Adding for A = -8, B = -8 excpected output: -16
# Passed Subtracting for A = -8, B = -8 excpected output: 0
# Passed Adding for A = -8, B = 0 excpected output: -8
# Passed Subtracting for A = -8, B = 0 excpected output: -8
# Passed Adding for A = -8, B = 7 excpected output: -1
# Passed Subtracting for A = -8, B = 7 excpected output: -15
# Passed Adding for A = 0, B = 7 excpected output: 7
# Passed Subtracting for A = 0, B = 7 excpected output: -7
# Passed Adding for A = 7, B = 7 excpected output: 14
# Passed Subtracting for A = 7, B = 7 excpected output: 0
# Passed Adding for A = 7, B = 0 excpected output: 7
# Passed Subtracting for A = 7, B = 0 excpected output: 7
# Passed Adding for A = 0, B = 0 excpected output: 0
# Passed Subtracting for A = 0, B = 0 excpected output: 0
# Passed Adding for A = -8, B = 0 excpected output: -8
# Passed Subtracting for A = -8, B = 0 excpected output: -8
# Passed Op1 Inverting for A = 100,output: 11011
# Passed Op1 Inverting for A = 1,output: 11110
# Passed Op1 Inverting for A = 1001,output: 110
# Passed Op1 Inverting for A = 11,output: 11100
# Passed Op1 Inverting for A = 1101,output: 10
# Passed Op1 Inverting for A = 1101,output: 10
# Passed Op1 Inverting for A = 101,output: 11010
# Passed Op1 Inverting for A = 10,output: 11101
# Passed Op1 Inverting for A = 1,output: 11110
# Passed Op1 Inverting for A = 1101,output: 10
# Passed Op2 Reduction OR for B = 110,output: 1
# Passed Op2 Reduction OR for B = 1101,output: 1
# Passed Op2 Reduction OR for B = 1101,output: 1
# Passed Op2 Reduction OR for B = 1100,output: 1
# Passed Op2 Reduction OR for B = 1001,output: 1
# Passed Op2 Reduction OR for B = 110,output: 1
# Passed Op2 Reduction OR for B = 101,output: 1
# Passed Op2 Reduction OR for B = 1010,output: 1
# Passed Op2 Reduction OR for B = 101,output: 1
# Passed Op2 Reduction OR for B = 111,output: 1
# Passed Adding for A = 6, B = -3 excpected output: 3
# Passed Subtracting for A = 6, B = -3 excpected output: 9
# Passed Op1 Inverting for A = 110,output: 11001
# Passed Op2 Reduction OR for B = 1101,output: 1
# Reset Passed
# Error Count = 0
#
# Correct Count = 42
#
```

Waveform:

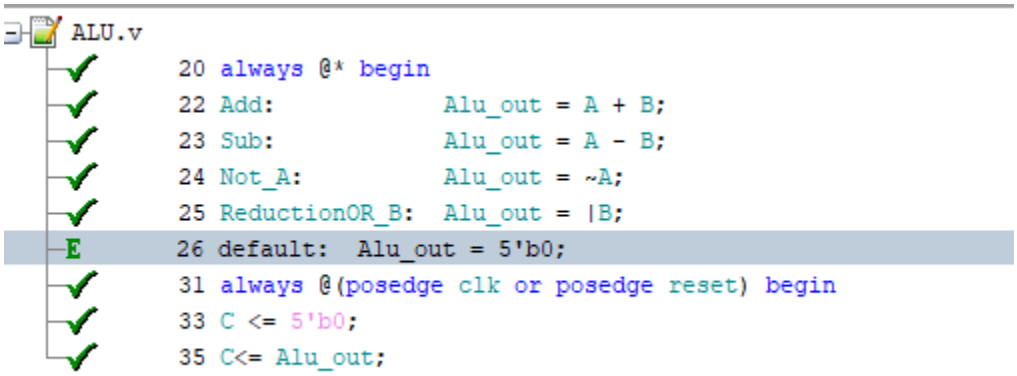


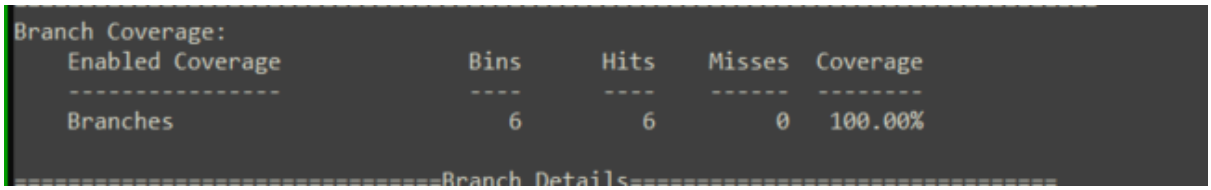
❖ Coverage Report:

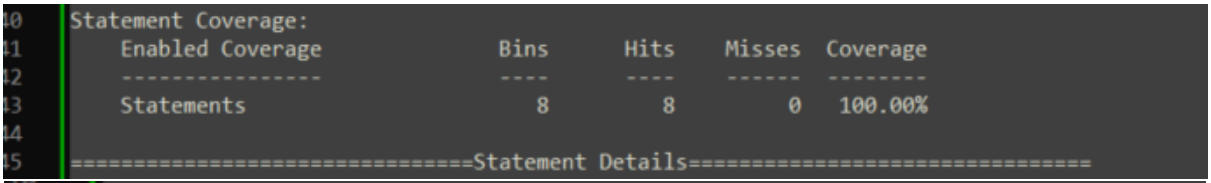
```
Coverage Report by instance with details

=================================================================================
=== Instance: /\ALU_tb#DUT
=== Design Unit: work.ALU_4_bit
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         7         6         1    85.71%

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

Branch Coverage for instance /\ALU_tb#DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ALU.v
-----------------------------------CASE Branch-----------------------------------
    21                                        39     Count coming in to CASE
    22              1                         10             Add:            Alu_out = A + B;

    23              1                          9             Sub:            Alu_out = A - B;

    24              1                         10             Not_A:          Alu_out = ~A;

    25              1                         10             ReductionOR_B:  Alu_out = |B;

    26              1                    ***0***             default:  Alu_out = 5'b0;

Branch totals: 4 hits of 5 branches = 80.00%

                                    If Branch
```

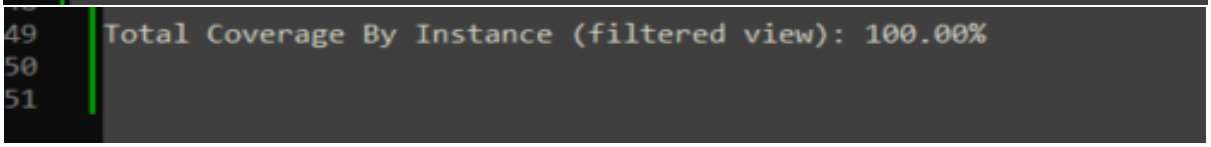
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         44        44         0   100.00%

================================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)
```

- Missing Bin in Branch and statement coverages is the **default** of the Opcode Case Statement.
- This statement can't be reached because the Case statement handles all the possible Opcode value, so this miss will be excluded from both reports.









Q3) DSP

❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
DSP1	when reset is de-asserted, Output (P) should be **Low for three clock cycles** (Internal registers should also be reset).	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
DSP2	checking the normal operation out of reset.	Randomized during the simulation	-	Comparing P to the output of a reference golden model's P_expected.

❖ TestBench:

```
module DSP_tb ();
    logic [17:0] A,B,D;
    logic [47:0] C;
    logic rst_n,clk;
    logic [47:0] P;

    integer Error_Count, Correct_Count;

    DSP DUT (.*);

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

initial begin
    Error_Count = 0; Correct_Count = 0;

//DSP_1
     A = 1;B = 1; D = 1; C = 1;
    assert_reset;

//DSP_2
    repeat (25) begin
        A = $random;
        B = $random;
        D = $random;
        C = $random;
        check_result(A,B,D,C);
    end

    assert_reset;  // to complete the toggle coverage for rst_n signal
```

```
        $display("Error Count = %0d\n",Error_Count);
        $display("Correct Count = %0d\n",Correct_Count);
        $stop;
    end


    // Golden model
    task DSP_Golden_Model(
            input [17:0] A,B,D,
            input [47:0] C,
            output [47:0] P_expected
    );
        logic [17:0] adder_result;
        adder_result = B+D;
        P_expected = (adder_result * A)+C;

    endtask
```

```
    // Tasks
    task assert_reset ;

        rst_n = 0;
        @(negedge clk);   // check the reset of the Output register
        if (P != 'b0)
         begin
          $display("Reset failed");
          Error_Count++;
        end
        else begin
            C = 0;
            rst_n = 1; // to allow inputs to propegate
             @(negedge clk);
             @(negedge clk);   // check the rest of internal registers

             if (P !== 'b0)
             begin
                $display("Reset failed");
                Error_Count++;
            end
            else begin
                 $display("Reset Passed");
                 Correct_Count++;
            end
        end
    endtask

```

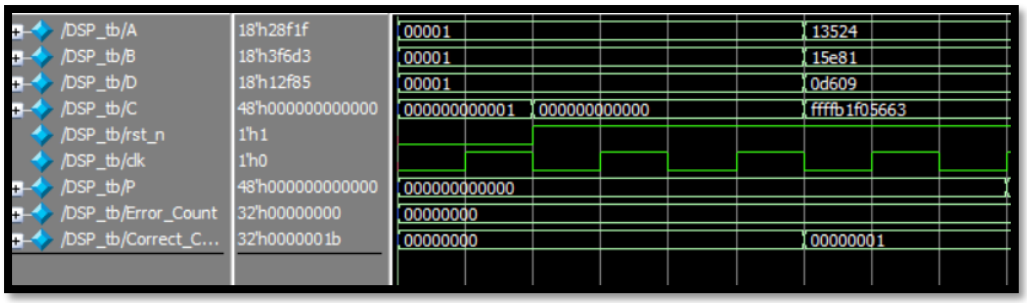
```
task check_result (
        input [17:0] A,B,D,
        input [47:0] C
);
    logic [47:0] P_expected;

    DSP_Golden_Model(A,B,D,C,P_expected);

    @(negedge clk);
    @(negedge clk);
    @(negedge clk);
    @(negedge clk);

    if(P == P_expected) begin
        $display("Passed for A = %0h, B = %0h, D = %0h, C = %0h -> Expected Result = %0h",A,B,D,C,P_expected);
        Correct_Count++;
    end
    else begin
        $display("Time: %0t, Failed for A = %0h, B = %0h, D = %0h, C = %0h -> Expected Result = %0h, Result = %0h",
        $time, A,B,D,C,P_expected,P);

        Error_Count++;
        $stop;
    end
endtask

endmodule
```

❖ **Results:**

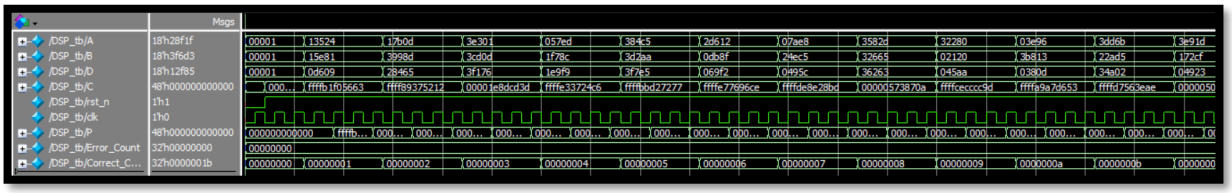
- Transcript:

```
# vsim -voptargs="+acc" DSP_tb -coverage
# Start time: 22:23:19 on Mar 06,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.DSP_tb(fast)
# Loading work.DSP(fast)
# Reset Passed
# Passed for A = 13524, B = 15e81, D = d609, C = ffffb1f05663 -> Expected Result = 25baa4bcb
# Passed for A = 17b0d, B = 3998d, D = 28465, C = ffff89375212 -> Expected Result = 2aba81d5c
# Passed for A = 3e301, B = 3cd0d, D = 3f176, C = 1e8dcd3d -> Expected Result = eac08b4c0
# Passed for A = 57ed, B = 1f78c, D = 1e9f9, C = ffffe33724c6 -> Expected Result = 138731fe7
# Passed for A = 384c5, B = 3d2aa, D = 3f7e5, C = ffffbbd27277 -> Expected Result = d12dc0e82
# Passed for A = 2d612, B = db8f, D = 69f2, C = ffffe77696ce -> Expected Result = 382a94fe0
# Passed for A = 7ae8, B = 24ec5, D = 495c, C = ffffde8e28bd -> Expected Result = 11d67c0a5
# Passed for A = 3582d, B = 32665, D = 36263, C = 573870a -> Expected Result = 87f425232
# Passed for A = 32280, B = 2120, D = 45aa, C = ffffcecccc9d -> Expected Result = 11105059d
# Passed for A = 3e96, B = 3b813, D = 380d, C = ffffa9a7d653 -> Expected Result = a01e4913
# Passed for A = 3dd6b, B = 22ad5, D = 34a02, C = ffffd7563eae -> Expected Result = 57854af8b
# Passed for A = 3e91d, B = 172cf, D = 4923, C = 509650a -> Expected Result = 6cd20f174
# Passed for A = 30aca, B = 14c3c, D = bdf2, C = 452e618a -> Expected Result = 679ba35d6
# Passed for A = b341, B = 334d8, D = f378, C = ffffc48a1289 -> Expected Result = ffffe0c43ed9
# Passed for A = 10deb, B = 265b6, D = 3f9c6, C = 571513ae -> Expected Result = 2d7980682
# Passed for A = 102bc, B = 3dd2a, D = 39a0b, C = ffffb897be71 -> Expected Result = 33946b35d
# Passed for A = 24185, B = 2554f, D = 603b, C = 1d06333a -> Expected Result = 6399a8dec
# Passed for A = 3327e, B = 24b15, D = 19bf1, C = 6c9c4bd9 -> Expected Result = ce6b92ccd
# Passed for A = 30762, B = 1fb4c, D = 1559f, C = 47b9a18f -> Expected Result = a52f60885
# Passed for A = 1a9f8, B = 160b7, D = 569f, C = ffffae7d945c -> Expected Result = 28984f5ac
# Passed for A = 3c05b, B = 23789, D = 23249, C = ffffe8233ed0 -> Expected Result = 1751c5c76
# Passed for A = 2c0d7, B = 3fc51, D = 12f96, C = 61d7f0c -> Expected Result = 33fd49e0d
# Passed for A = cec2, B = 3edc8, D = 25a77, C = 1ef2ed3d -> Expected Result = 1f6d05efb
# Passed for A = db12, B = 1007e, D = 1816d, C = 1cd9e739 -> Expected Result = 2422b12bf
# Passed for A = 28f1f, B = 3f6d3, D = 12f85, C = ffffbc148878 -> Expected Result = 2ad535520
# Reset Passed
# Passed for A = 3ffff, B = 1, D = 0, C = 0 -> Expected Result = 3ffff
# Error Count = 0
#
# Correct Count = 28
#
# ** Note: $stop    : DSP_tb.sv(44)
#    Time: 220 ns  Iteration: 1  Instance: /DSP_tb
```

- DSP_1:



- DSP_2:



❖ Bugs found:

Label	Bug Description	Wrong code	Correction

DSP_1	Internal register(C_reg) wasn't affected by rst_n.	`// reset`
		`A_reg_stg1 <= 0;`
		`A_reg_stg2 <= 0;`
		`B_reg <= 0;`
		`D_reg <= 0;`
		`C_reg <= 0;  //Bug: This line was missing`
		`adder_out_stg1 <= 0;`
		`mult_out <= 0;`
		`P <= 0;`
			Added C_reg to the reset condition in the RTL.

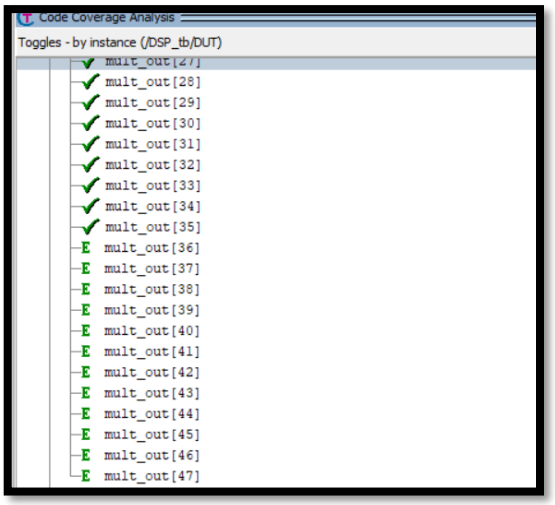
DSP_2	the internal signal (adder_out_stg2) is redundant and causes an extra register after the first stage adder	`//adder_out_stg2 <= adder_out_stg1;` `//Bug: causing a redundant register after the first adder`	Signal is totally removed, and multiplier_out signal will directly take the value of adder_out_stg1.

❖ Coverage Report:

```
================================================================================
=== Instance: /\DSP_tb#DUT
=== Design Unit: work.DSP
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         2         2         0   100.00%

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

Branch Coverage for instance /\DSP_tb#DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File DSP.v
------------------------------------IF Branch------------------------------------
    12                                       112     Count coming in to IF
    12             1                           4        if (!rst_n) begin

    24             1                         108        else begin

Branch totals: 2 hits of 2 branches = 100.00%


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      17        17         0   100.00%
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        676       652        24    96.44%

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

Toggle Coverage for instance /\DSP_tb#DUT  --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                        A[0-17]           1           1      100.00
                                 A_reg_stg1[17-0]         1           1      100.00
                                 A_reg_stg2[17-0]         1           1      100.00
                                        B[0-17]           1           1      100.00
                                     B_reg[17-0]          1           1      100.00
                                        C[0-47]           1           1      100.00
                                     C_reg[47-0]          1           1      100.00
                                        D[0-17]           1           1      100.00
                                     D_reg[17-0]          1           1      100.00
                                        P[47-0]           1           1      100.00
                            adder_out_stg1[17-0]          1           1      100.00
                                         clk              1           1      100.00
                                mult_out[47-36]           0           0        0.00
                                 mult_out[35-0]           1           1      100.00
                                         rst_n            1           1      100.00

Total Node Count      =        338
Toggled Node Count    =        326
Untoggled Node Count =         12

Toggle Coverage       =      96.44% (652 of 676 bins)


Total Coverage By Instance (filtered view): 98.81%
```

- Un-Toggled upper bits of mult_out (mult_out[47:36]) will never be toggled, since the multiplier inputs (A and B/D) are **18-bit signals**, so their multiplication will need at most **36 bits** (mult_out[35:0]).



```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        652       652         0   100.00%

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

Toggle Coverage for instance /\DSP_tb#DUT  --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                     A[0-17]           1           1      100.00
                                A_reg_stg1[17-0]       1           1      100.00
                                A_reg_stg2[17-0]       1           1      100.00
                                     B[0-17]           1           1      100.00
                                B_reg[17-0]            1           1      100.00
                                     C[0-47]           1           1      100.00
                                C_reg[47-0]            1           1      100.00
                                     D[0-17]           1           1      100.00
                                D_reg[17-0]            1           1      100.00
                                     P[47-0]           1           1      100.00
                           adder_out_stg1[17-0]        1           1      100.00
                                       clk             1           1      100.00
                             mult_out[35-0]            1           1      100.00
                                     rst_n             1           1      100.00

Total Node Count     =        326
Toggled Node Count   =        326
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (652 of 652 bins)


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