q1

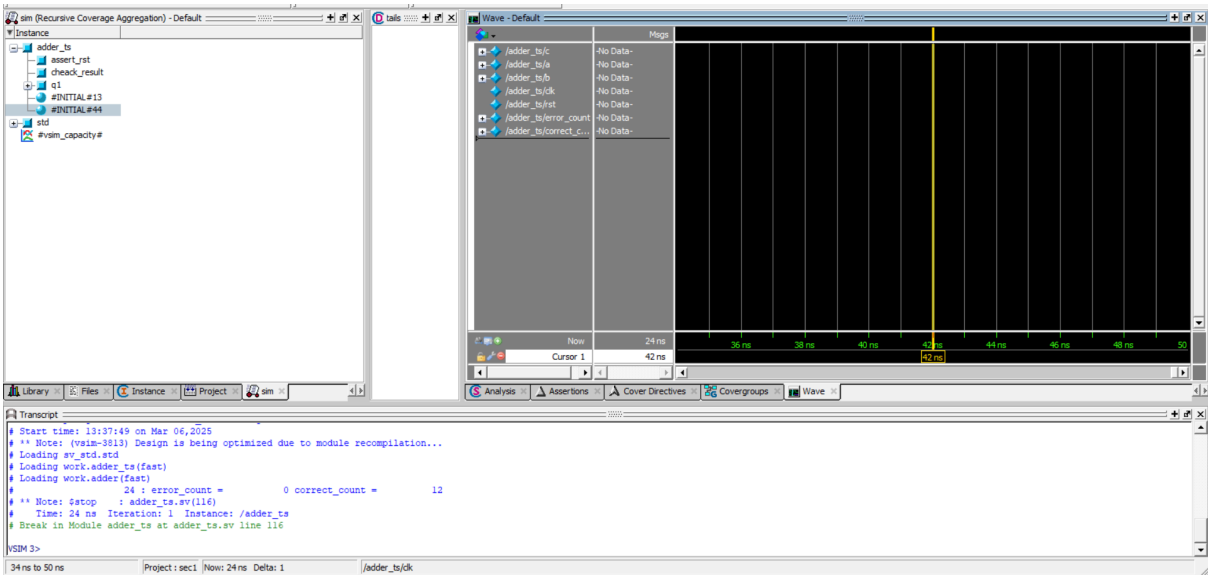
adder_ts.sv

```
module adder_ts ();
wire signed [4:0] c;      // output
reg signed [3:0] a,b;     // input
reg clk,rst;              // input

localparam MAXPOS = 7, ZERO = 0, MAXNEG = -8;
int error_count = 0;
int correct_count = 0;

adder q1 (c,a,b,clk,rst);
// adder q1 (.*); when the same name

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

    task  assert_rst ();
        rst = 1;
        @(negedge clk);
        if (c !== 5'b0) begin
            $display ("not correct");
            error_count++;
            $stop;
        end
        else
            correct_count++;
        rst = 0;
    endtask

        task  cheack_result ( input signed [4:0] c_expected);
          @(negedge clk);
          if (c !== c_expected) begin
             $display("not correct for a=%b b=%b", a, b);
             error_count++;
             $stop;
          end
          else
             correct_count++;
    endtask

initial begin
    a = 0;
    b = 0;

    // adder_1

     assert_rst;

    // adder_2

    assert_rst;

    // adder_3

    a = MAXPOS;
    b = MAXNEG;
```

```
44  initial begin
59      b = MAXNEG;
60      cheack_result (-1);
61
62      // adder_4
63      a = MAXPOS;
64      b = ZERO;
65      cheack_result (7);
66
67      // adder_5
68      a = MAXPOS;
69      b = MAXPOS;
70      cheack_result (14);
71
72      // adder_6
73
74      a = ZERO;
75      b = MAXNEG;
76      cheack_result (-8);
77
78      // adder_7
79
80      a = ZERO;
81      b = ZERO;
82      cheack_result (0);
83
84      // adder_8
85
86      a = ZERO;
87      b = MAXPOS;
88      cheack_result (7);
89
90      // adder_9
91
92      a = MAXNEG;
93      b = MAXNEG;
94      cheack_result (-16);
95
96      // adder_10
97
98      a = MAXNEG;
99      b = ZERO;
100     cheack_result (-8);
101
102     // adder_11
103
104     a = MAXNEG;
105     b = MAXPOS;
106     cheack_result (-1);
107
108     // adder_12
109
110     a = MAXPOS;
111     b = MAXNEG;
112     cheack_result (-1);
113
114
115     $display (" %t : error_count = %d correct_count = %d",$time ,error_count,correct_count);
```

adder.do

```
1   vlib work
2   vlog adder.v adder_ts.sv +cover -covercells
3   vsim -voptargs=+acc work.adder_ts -cover
4   add wave *
5   coverage save adder_ts.ucdb -onexit
6   run -all
```




```
File    Edit    View
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       3         3         0   100.00%

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

Statement Coverage for instance /adder_ts/q1 --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File adder.v
    1                                                module adder ( output reg signed [4:0] c,
    2                                                              input signed [3:0] a,b,
    3                                                              input clk,rst);
    4               1                           14         always @ (posedge clk , posedge rst ) begin
    5                                                             if (rst)
    6               1                            4                c <= 5'b00000;
    7                                                             else
    8               1                           10                c <= a + b;

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

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

Toggle Coverage for instance /adder_ts/q1 --

                                       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
                                       rst           1           1      100.00

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

Toggle Coverage      =     100.00% (30 of 30 bins)
```

q2

	A	B	C	D
1	label	description	stimulus generation	functionality cheack
2	ALU_1	When the reset is asserted, the output C value must be low	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
3	ALU_2	Verifying opcode to be 2'b00 ,output c be A + B	Directed during the simulation	A checker in the testbench to make sure the output is correct
4	ALU_3	When the reset is asserted, the output C value must be low	Directed during the simulation	A checker in the testbench to make sure the output is correct
5	ALU_4	Verifying opcode to be 2'b00 ,output c be A + B	Directed during the simulation	A checker in the testbench to make sure the output is correct
6	ALU_5	Verifying opcode to be 2'b01 ,output c be A - B	Directed during the simulation	A checker in the testbench to make sure the output is correct
7	ALU_6	Verifying opcode to be 2'b01 ,output c be A - B	Directed during the simulation	A checker in the testbench to make sure the output is correct
8	ALU_7	Verifying opcode to be 2'b10 ,output c be ~A	Directed during the simulation	A checker in the testbench to make sure the output is correct
9	ALU_8	Verifying opcode to be 2'b10 ,output c be ~A	Directed during the simulation	A checker in the testbench to make sure the output is correct
10	ALU_9	Verifying opcode to be 2'b11 ,output c be |B	Directed during the simulation	A checker in the testbench to make sure the output is correct
11	ALU_10	Verifying opcode to be 2'b11 ,output c be |B	Directed during the simulation	A checker in the testbench to make sure the output is correct
12	ALU_11	Verifying opcode to be 2'b00 ,output c be A + B	Directed during the simulation	A checker in the testbench to make sure the output is correct
13	ALU_12	Verifying opcode to be 2'b01 ,output c be A - B	Directed during the simulation	A checker in the testbench to make sure the output is correct
14	ALU_13	Verifying opcode to be 2'b10 ,output c be ~A	Directed during the simulation	A checker in the testbench to make sure the output is correct
15				

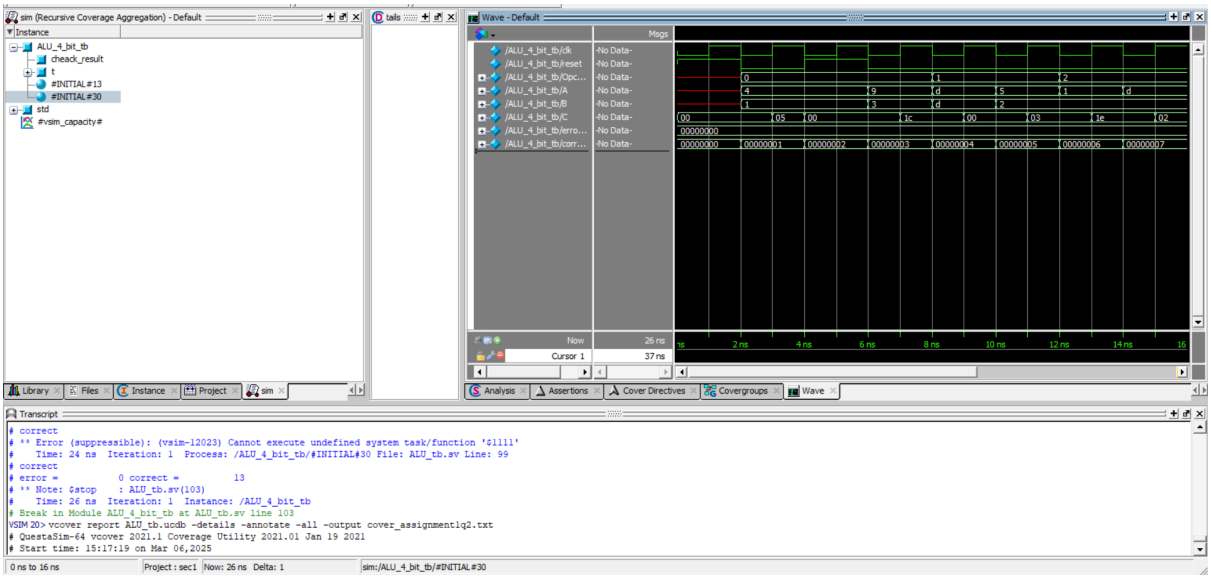
ALU_tb.sv

```
module ALU_4_bit_tb ;
    reg  clk;
    reg  reset;
    reg  [1:0] Opcode;  // The opcode
    reg  signed [3:0] A;    // Input data A in 2's complement
    reg  signed [3:0] B;    // Input data B in 2's complement
    wire signed [4:0] C; // ALU output in 2's complement
   ALU_4_bit t (.*);

   int error_count = 0;
   int correct_count =0;

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

   task cheack_result ( input signed [4:0] out);
    @(negedge clk);
    if (C != out) begin
        $display ("error");
        error_count++;
    end
    else begin
        $display ("correct");
        correct_count++;
    end
    endtask

    initial begin
        //label1
        reset = 1;
        cheack_result (0);
        reset = 0;

        //label2
        Opcode = 2'b00;
        A = $random;
        B = $random;
        cheack_result (A+B);

        //label3
        reset = 1;
        cheack_result (0);
        reset = 0;

        //label4
        Opcode = 2'b00;
        A = $random;
        B = $random;
        cheack_result (A+B);

        //label5
        Opcode = 2'b01;
        A = $random;
        B = $random;
        cheack_result (A-B);
```

```
58
59          //label6
60          Opcode = 2'b01;
61          A = $random;
62          B = $random;
63          cheack_result (A-B);
64
65          //label7
66          Opcode = 2'b10;
67          A = $random;
68          cheack_result (~A);
69
70          //label8
71          Opcode = 2'b10;
72          A = $random;
73          cheack_result (~A);
74
75          //label9
76          Opcode = 2'b11;
77          B = $random;
78          cheack_result (/B);
79
80          //label10
81          Opcode = 2'b11;
82          B = $random;
83          cheack_result (/B);
84
85          //label11
86          Opcode = 2'b00;
87          A = 4'b1111;
88          B = $random;
89          cheack_result (A+B);
90
91          //label12
92          Opcode = 2'b01;
93          A = 4'b0000;
94          B = $random;
95          cheack_result (A-B);
96
97          //label13
98          Opcode = 2'b10;
99          A = $1111;
100         cheack_result (~A);
101
102         $display("error = %d correct = %d", error_count, correct_count);
103         $stop;
104     end
```

ALU.do

```
1   vlib work
2   vlog ALU.v ALU_tb.sv +cover -covercells
3   vsim -voptargs=+acc work.ALU_4_bit_tb -cover
4   add wave *
5   coverage save ALU_tb.ucdb -onexit
6   run -all
```



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

Toggle Coverage for instance /ALU_4_bit_tb/t --

                                          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)

================================================================================
=== Instance: /ALU_4_bit_tb
=== Design Unit: work.ALU_4_bit_tb
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         2         1         1    50.00%

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

Branch Coverage for instance /ALU_4_bit_tb
NOTE: The modification timestamp for source file 'ALU_tb.sv' has been altered since compilation.

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File ALU_tb.sv
------------------------------------IF Branch------------------------------------
    20                                        13     Count coming in to IF
    20              1                     ***0***        if (C != out) begin
    24              1                        13         else begin
Branch totals: 1 hit of 2 branches = 50.00%
```

q3

	A	B	C	D
1	label	description	stimulus generation	functionality cheack
2	ALU_1	When the reset is asserted, the output C value must be low	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
3	ALU_2	Verifying D = 4'b1000 , output Y be 0 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
4	ALU_3	When the reset is asserted, the output C value must be low	Directed during the simulation	A checker in the testbench to make sure the output is correct
5	ALU_4	Verifying D = 4'b1100 , output Y be 1 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
6	ALU_5	Verifying D = 4'b0100 , output Y be 1 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
7	ALU_6	Verifying D = 4'b1010 , output Y be 2 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
8	ALU_7	Verifying D = 4'b1110 , output Y be 2 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
9	ALU_8	Verifying D = 4'b1111 , output Y be 3 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
10	ALU_9	Verifying D = 4'b1000 , output Y be 0 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
11	ALU_10	Verifying D = 4'b0111 , output Y be 0 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
12	ALU_11	Verifying D = 4'b0000 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct
13	ALU_12	Verifying D = 4'b0011 , output Y be 3 , output valid be |D	Directed during the simulation	A checker in the testbench to make sure the output is correct

priority_enc_tb.sv

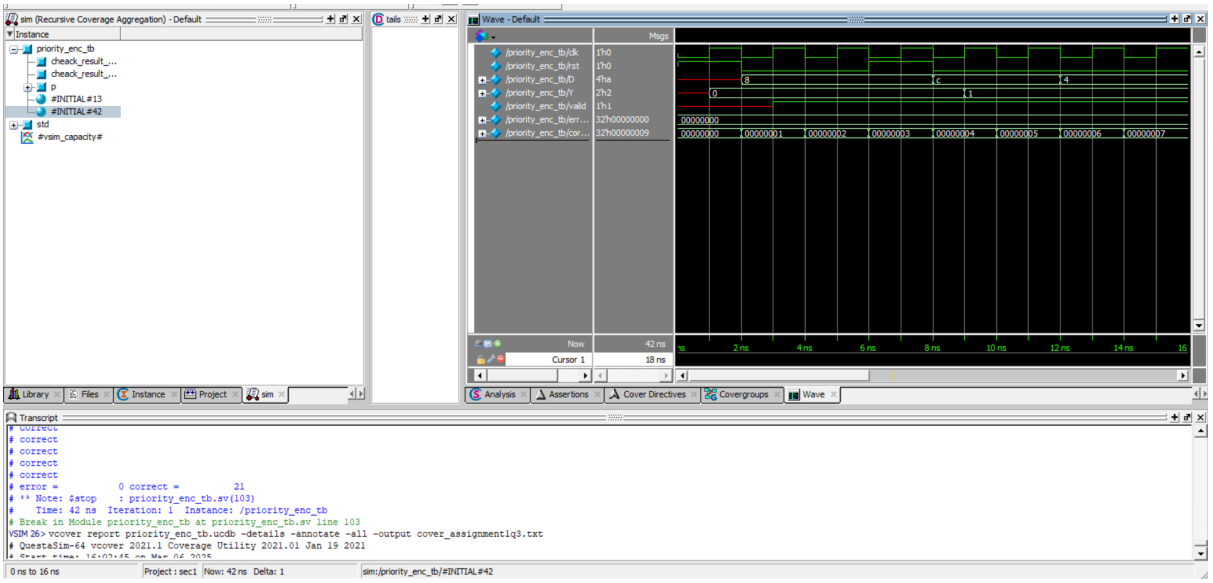
```
module priority_enc_tb ;
reg  clk;
reg  rst;
reg  [3:0] D;
wire  [1:0] Y;
wire valid;

priority_enc p (.*);

    int error_count = 0;
    int correct_count =0;

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

    task cheack_result_y ( input [1:0] out);
     @(negedge clk);
     if (out != Y) begin
        $display ("error");
        error_count++;
     end
     else begin
         $display ("correct");
         correct_count++;
     end
     endtask

    task cheack_result_valid ( input valid_expected);
     @(negedge clk);
     if (valid_expected != valid) begin
        $display ("error");
        error_count++;
     end
     else begin
         $display ("correct");
         correct_count++;
     end
     endtask

    initial begin
     // label1
     rst = 1;
     cheack_result_y (0);
     rst = 0;

     // label 2
     D = 4'b1000;
     cheack_result_y (0);
     cheack_result_valid (|D);

     // label3
     rst = 1;
     cheack_result_y (0);
     rst = 0;

```

```
    // label 5
    D = 4'b0100;
    cheack_result_y (1);
    cheack_result_valid (!D);

    // label 6
    D = 4'b1010;
    cheack_result_y (2);
    cheack_result_valid (!D);

    // label 7
    D = 4'b1110;
    cheack_result_y (2);
    cheack_result_valid (!D);

    // label 8
    D = 4'b1111;
    cheack_result_y (3);
    cheack_result_valid (!D);

    // label 9
    D = 4'b1000;
    cheack_result_y (0);
    cheack_result_valid (!D);

    // label 10
    D = 4'b0111;
    cheack_result_y (3);
    cheack_result_valid (!D);

    // label 11
    D = 4'b0000;
    cheack_result_valid (!D);

    // label 6
    D = 4'b0011;
    cheack_result_y (3);
    cheack_result_valid (!D);

        $display("error = %d correct = %d", error_count, correct_count);
        $stop;
    end
```

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

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

priority_enc.do

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




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


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       7         7         0   100.00%

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

Statement Coverage for instance /priority_enc_tb/p --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  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,             /////// reg
    6                                                output reg valid
    7                                                );
    8
    9               1                           19    always @(posedge clk) begin
    10                                                 if (rst)
    11              1                            2       Y <= 2'b0;
    12                                                 else
    13                                                  casex (D)
    14              1                            4              4'b1000: Y <= 0;
    15              1                            3              4'bX100: Y <= 1;
    16              1                            3              4'bXX10: Y <= 2;
    17              1                            6              4'bXXX1: Y <= 3;
    18                                                  endcase
    19              1                           19        valid <= (~|D)? 1'b0: 1'b1;

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

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

Toggle Coverage for instance /priority_enc_tb/p
```

q4

	A	B	C	D
1	label	description	stimulus generation	functionality cheack
2	ALU_1	When the reset is asserted, the output C value must be low	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
3	ALU_2	Verifying A = random ,B = random , C = random , D = random and cheack output with golden model for 1000 round	Directed during the simulation	A checker in the testbench to make sure the output is correct
4	ALU_3	When the reset is asserted, the output C value must be low	Directed during the simulation	A checker in the testbench to make sure the output is correct