Adder project

1-design

```
module adder (
    input  clk,
    input  reset,
    input  signed [3:0] A,      // Input data A in 2's complement
    input  signed [3:0] B,      // Input data B in 2's complement
    output reg signed [4:0] C // Adder output in 2's complement
                  );

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

endmodule
```

2-verification plan

Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
ADDER_1	When the reset is asserted, the output adder value should be low the we assert reset to low	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
ADDER_2	When A is asserted to Max postive number and B asserted to Max negative number the output C should be equal -1	Directed at the simulation at time 10	-	A checker in the testbench to make sure the output is correct
ADDER_3	When A is asserted to Max postive number and B asserted to Max postive number the output C should be equal 14	Directed at the simulation at time 20	-	A checker in the testbench to make sure the output is correct
ADDER_4	When A is asserted to Max negative number and B asserted to Max negative number the output C should be equal -16	Directed at the simulation at time 30	-	A checker in the testbench to make sure the output is correct
ADDER_5	When A is asserted to Max negative number and B asserted to Max postive number the output C should be equal -1	Directed at the simulation at time 40	-	A checker in the testbench to make sure the output is correct
ADDER_6	When A is asserted to Zero number and B asserted to Max negative number the output C should be equal -8	Directed at the simulation at time 50	-	A checker in the testbench to make sure the output is correct
ADDER_7	When A is asserted to zero number and B asserted to Max positive number the output C should be equal 7	Directed at the simulation at time 60	-	A checker in the testbench to make sure the output is correct
ADDER_8	When A is asserted to Max positive number and B asserted to Zero number the output C should be equal 7	Directed at the simulation at time 70	-	A checker in the testbench to make sure the output is correct
ADDER_9	When A is asserted to Max negative number and B asserted to zero number the output C should be equal -8	Directed at the simulation at time 80	-	A checker in the testbench to make sure the output is correct
ADDER_10	When A is asserted to zero number and B asserted to zero number the output C should be equal 0	Directed at the simulation at time 90	-	A checker in the testbench to make sure the output is correct
ADDER_11	When the reset is asserted, the output adder value should be low	Directed at the simulation at time 100	-	A checker in the testbench to make sure the output is correct
ADDER_12	we end simulation here by $stop	Directed at the simulation at time 110	-	A checker in the testbench to make sure the output is correct

3-testbench

```
30
31  A=MAXNEG;
32  B=MAXPOS;
33  checker_res(-1);
34
35  A=ZERO;
36  B=MAXNEG;
37  checker_res(-8);
38
39  A=ZERO;
40  B=MAXPOS;
41  checker_res(7);
42
43  A=MAXPOS;
44  B=ZERO;
45  checker_res(7);
46
47  A=MAXNEG;
48  B=ZERO;
49  checker_res(-8);
50
51  A=ZERO;
52  B=ZERO;
53  checker_res(0);
54
55  $display("error_count=%0d  ------- correct_count=%0d",error,correct);
56  $stop;
57
58  end
59
```

tb.sv

```
1   module tb();
2
3   localparam MAXPOS=7,MAXNEG=-8,ZERO=0;//when we used too much we should make as param
4   int correct,error;
5   bit  clk,reset;
6   logic  signed [3:0] A;  // Input data A in 2's complement
7   logic  signed [3:0] B;  // Input data B in 2's complement
8   logic signed [4:0] C;
9
10  adder tb1(.*);
11  initial
12  begin clk=0;
13  forever #5 clk=!clk;
14  end
15
16  initial begin
17  call_reset;
18
19  A=MAXPOS;
20  B=MAXNEG;
21  checker_res(-1);
22
23  A=MAXPOS;
24  B=MAXPOS;
25  checker_res(14);
26
27  A=MAXNEG;
28  B=MAXNEG;
29  checker_res(-16);
30
31  A=MAXNEG;
32  B=MAXPOS;
```

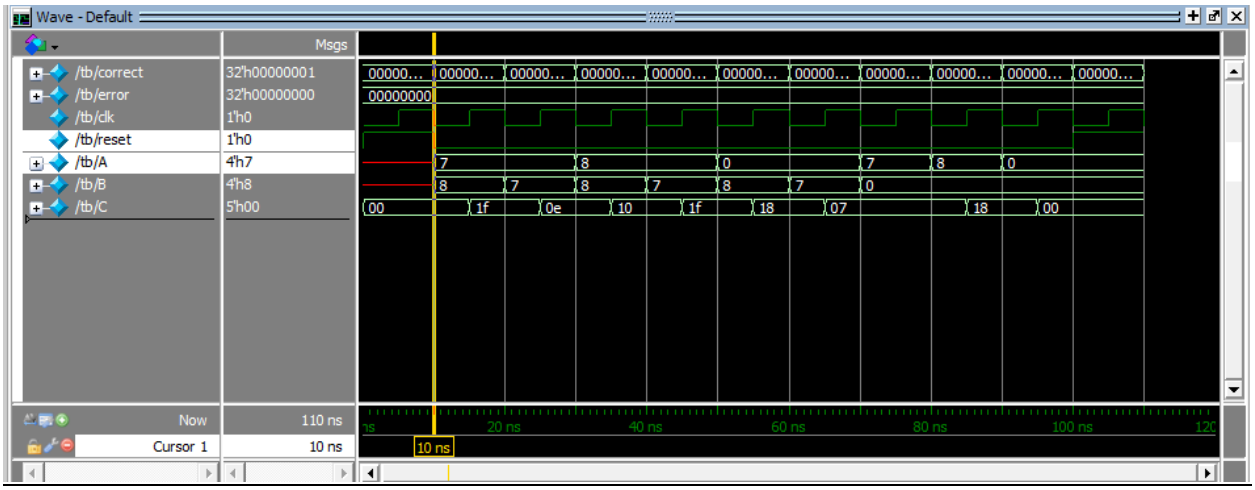
```
59
60  task checker_res(input logic signed[4:0] check_result);
61  @(negedge clk);
62  if(check_result!=C)begin
63  $display("there is somthing wrong @%t",$time);
64  error++;
65  end
66  else
67  correct++;
68
69  endtask
70
71  task call_reset;
72  reset=1;
73  checker_res(0);
74  reset=0;
75  endtask
76
77  endmodule
```

4-do file

File Edit Format View Help

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


//vcover report adder_tb.ucdb -details -all -output coverage_report.txt
```

5-simulation to check functionality



Coverage report

```
================================================================================
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
```

```
================================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%


Condition Coverage:
    Enabled Coverage           Active   Covered    Misses % Covered
    ----------------           ------      ----    ------ ---------
    FEC Condition Terms             0         0         0     100.0
Expression Coverage:
    Enabled Coverage           Active   Covered    Misses % Covered
    ----------------           ------      ----    ------ ---------
    FEC Expression Terms            0         0         0     100.0
FSM Coverage:
    Enabled Coverage           Active      Hits    Misses % Covered
    ----------------           ------      ----    ------ ---------
    FSMs                                                     100.0
        States                      0         0         0     100.0
        Transitions                 0         0         0     100.0
Toggle Coverage:
    Enabled Coverage           Active      Hits    Misses % Covered
    ----------------           ------      ----    ------ ---------
    Toggle Bins                   30        29         1      96.6

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

```
================================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

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

Toggle Coverage      =      100.0% (30 of 30 bins)
```

priority Encoder project

1-Design

E: > study > kareem wassem > ASSIGMENTS > 1 > priority_encoder > 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   always @(posedge clk) begin
10    if (rst) begin
11      Y <= 2'b0;
12      valid<=1'b0;
13      end
14    else
15      casex (D)
16          4'b1000: Y <= 0;//2'b00  1
17          4'bX100: Y <= 1;//2'b01  1
18          4'bXX10: Y <= 2;//2'b10  1
19          4'bXXX1: Y <= 3;//2'b11  1
20      endcase
21      valid <= (~|D)? 1'b0: 1'b1;
22  end
23  endmodule
```

2-Verification plan

A	B	C	D	E
Label	**Description**	**Stimulus Generation**	**Functional Coverage (Later)**	**Functionality Check**
PRIORTY_E_1	When the reset is asserted, the output valid value should be low and Y dont care	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
PRIORTY_E_2	we assert D to number from 0 to 15 with FOR LOOP but Y still dont care and Valid still 0 because reset is HIGH	Directed at the simulation	-	A checker in the testbench to make sure the output is correct
PRIORTY_E_3	we assert the reset to 0 to see the stimulus on Output Y and valid	Directed at the simulation	-	A checker in the testbench to make sure the output is correct
PRIORTY_E_4	we assert D to number from 0 to 15 with FOR LOOP and check output Y and valid according to if D[0]=1->Y=2'b11 ,valid=1'b1 else D[1]=1 ->Y=2'b10 ,valid=1'b1 else D[2]=1 ->Y=2'b01 ,valid=1'b1 else D[3]=1 ->Y=2'b00 ,valid=1'b1 else if D=0 ->Y=2'bxx ,valid=1'b0	Directed at the simulation	-	A checker in the testbench to make sure the output is correct

3-tb

```
initial begin
for(j=0;j<2;j++)
begin
reset_call(!j);
    for(i=0;i<16;i++)
    begin
    if(rst)
      check_correctness(0);
    else begin
      D=i;
      if(D[0]==1'b1)
        check_correctness(7);
      else if(D[1]==1'b1)
        check_correctness(5);
      else if(D[2]==1'b1)
        check_correctness(3);
      else if(D[3]==1'b1)
        check_correctness(1);
      else
        check_correctness(0);
    end
    end
end
D=0;
check_correctness(0);

reset_call(1);
check_correctness(0);

$display("error_count=%0d  ------- correct_count=%0d",wrong,correct);
$stop;
end
```

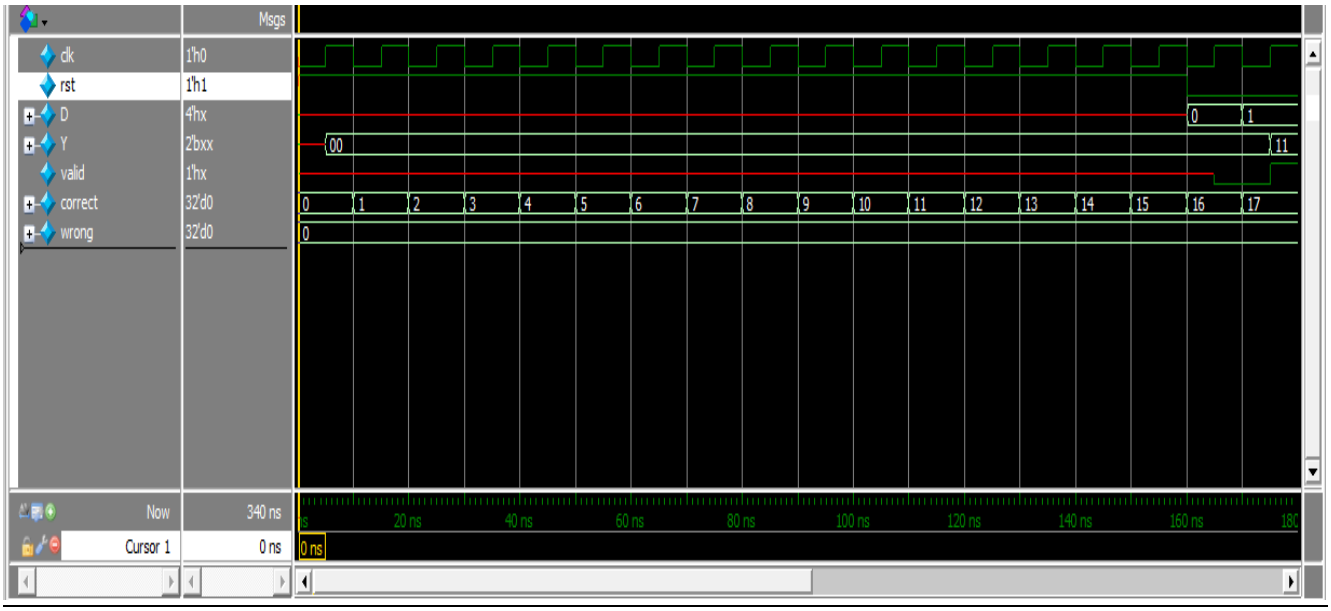
```
1  module priority_enc_tb();
2
3  logic clk;
4  logic rst;
5  logic [3:0] D;
6  logic [1:0] Y;
7  logic valid;
8
9
10 int correct=0 ,wrong=0,i=0,j=0;
11 priority_enc p1(.*);
12
13 initial begin
14 clk=0;
15 forever #5 clk=!clk;
16 end
```

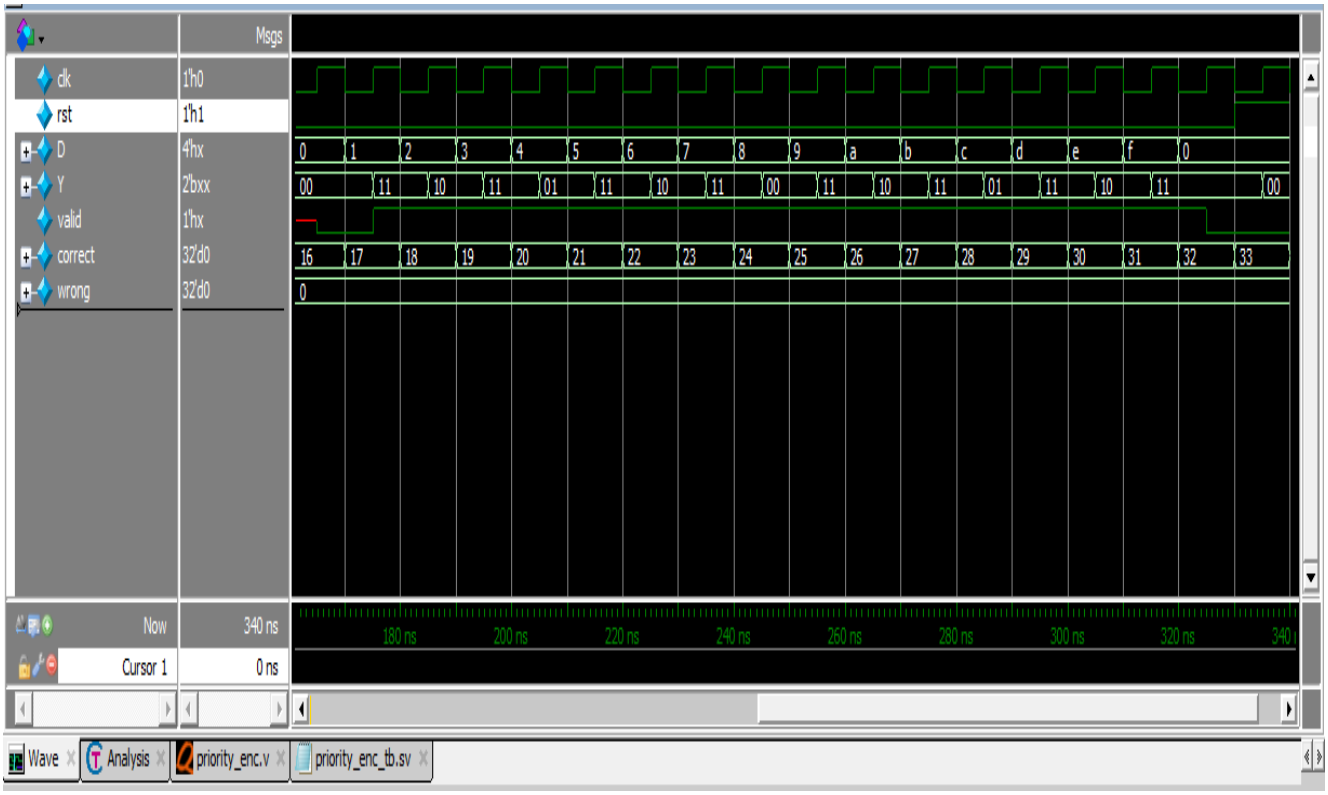
```
task check_correctness(input logic [2:0]value);
@(negedge clk);
if(rst && (value[2:1] !=Y))begin
    $display("there is somthing wrong @%t the number %0d",$time,{Y,valid});
    wrong++;
    end
else if(rst==0 && D >0 && value!={Y,valid})begin
    $display("there is somthing wrong @%t the number %0d",$time,{Y,valid});
    wrong++;
    end
else if(rst==0 && D==0 && value[0]!= valid)begin
    $display("there is somthing wrong @%t the number %0d",$time,{Y,valid});
    wrong++;
    end
else
    correct++;


endtask

task reset_call(input bit turn_on);
if(turn_on)
    rst=1;
else
    rst=0;
endtask

endmodule
```

4-do file

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


//vcover report priroty_encoder_tb.ucdb -details -all -output coverage_report.txt
```

5-simulation





5.2coverage report :statment

```
Coverage Report by file with details

================================================================================
=== File: priority_enc.v
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            7         7         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                   20   always @(posedge clk) begin
    10                                         if (rst)
    11              1                    3        Y <= 2'b0;
    12                                        else
    13                                         casex (D)
    14              1                    1               4'b1000: Y <= 0;//2'b00  1
    15              1                    2               4'bX100: Y <= 1;//2'b01  1
    16              1                    4               4'bXX10: Y <= 2;//2'b10  1
    17              1                    8               4'bXXX1: Y <= 3;//2'b11  1
    18                                         endcase
    19              1                   20        valid <= (~|D)? 1'b0: 1'b1;
    20                                       end
    21                                       endmodule
```

Branch

```
================================Branch Details================================

Branch Coverage for file priority_enc.v --

------------------------------------IF Branch------------------------------------
    10                                        20     Count coming in to IF
    10              1                          3       if (rst)
    12              1                         17       else
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------CASE Branch----------------------------------
    13                                        17     Count coming in to CASE
    14              1                          1               4'b1000: Y <= 0;//2'b00  1
    15              1                          2               4'bX100: Y <= 1;//2'b01  1
    16              1                          4               4'bXX10: Y <= 2;//2'b10  1
    17              1                          8               4'bXXX1: Y <= 3;//2'b11  1
                                               2     All False Count
Branch totals: 5 hits of 5 branches = 100.0%


Condition Coverage:
    Enabled Coverage             Active    Covered    Misses % Covered
    ----------------             ------      ----    ------ ---------
    FEC Condition Terms               0         0         0     100.0
Expression Coverage:
    Enabled Coverage             Active    Covered    Misses % Covered
    ----------------             ------      ----    ------ ---------
    FEC Expression Terms              0         0         0     100.0
FSM Coverage:
    Enabled Coverage             Active       Hits    Misses % Covered
    ----------------             ------       ----    ------ ---------
    FSMs                                                        100.0
        States                        0          0         0     100.0
        Transitions                   0          0         0     100.0
Toggle Coverage:
    Enabled Coverage             Active       Hits    Misses % Covered
    ----------------             ------       ----    ------ ---------
    Toggle Bins                      18         18         0     100.0

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

Toggle

```
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)


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

ALU PROJECT

DESIGN

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

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

        );

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

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

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

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

endmodule
```

TB

```
module tb();

logic clk;
logic reset;
logic  [1:0] Opcode;    // The opcode
logic signed [3:0] A;   // Input data A in 2's complement
logic signed [3:0] B;   // Input data B in 2's complement
logic signed [4:0] C;    // ALU output in 2's complement

int error=0,correct=0;
localparam MAXPOS=7,MAXNEG=-8,ZERO=0;
localparam Add = 2'b00,Sub = 2'b01,Not_A= 2'b10,ReductionOR_B = 2'b11;

ALU_4_bit T1(.*);
initial begin
clk=0;
forever #5 clk=!clk;
end

initial begin
Opcode=ZERO;
A=ZERO;
B=ZERO;
call_reset;

Opcode=Add;//
A=MAXPOS;
B=MAXNEG;
checker_res(-1);

Opcode=Sub;
checker_res(15);

Opcode=Add;
A=MAXPOS;
B=MAXPOS;
checker_res(14);
```

```
38
39  Opcode=Sub;
40  checker_res(0);
41
42  Opcode=Add;
43  A=MAXNEG;
44  B=MAXNEG;
45  checker_res(-16);
46
47  Opcode=Sub;
48  checker_res(0);
49
50  Opcode=Add;
51  A=MAXNEG;
52  B=MAXPOS;
53  checker_res(-1);
54
55  Opcode=Sub;
56  checker_res(-15);
57
58  Opcode=Add;
59  A=ZERO;
60  B=MAXNEG;
61  checker_res(-8);
62
63  Opcode=Sub;
64  checker_res(8);
65
66  Opcode=Add;
67  A=ZERO;
68  B=MAXPOS;
69  checker_res(7);
70
71  Opcode=Sub;
72  checker_res(-7);
73
```

```
73
74  Opcode=Add;
75  A=MAXPOS;
76  B=ZERO;
77  checker_res(7);
78
79  Opcode=Sub;
80  checker_res(7);
81
82  Opcode=Add;
83  A=MAXNEG;
84  B=ZERO;
85  checker_res(-8);
86
87  Opcode=Sub;
88  checker_res(-8);
89
90  Opcode=Add;
91  A=ZERO;
92  B=ZERO;
93  checker_res(0);
94
95  Opcode=Sub;
96  checker_res(0);
97
98  Opcode=Not_A;
99  A=ZERO;
100 checker_res(-1);
101
102 A=-1;
103 checker_res(ZERO);
104
105 A=MAXNEG;
106 checker_res(MAXPOS);
107
```

```
A=MAXPOS;
checker_res(MAXNEG);

Opcode=ReductionOR_B;
B=ZERO;
checker_res(ZERO);

B=MAXPOS;
checker_res(1);

B=MAXNEG;
checker_res(1);

B=-1;
checker_res(1);

B=1;
checker_res(1);

reset=1;
Opcode=Add;
checker_res(0);

$display("error_count=%0d  ------- correct_count=%0d",error,correct);
$stop;
end

task checker_res(input logic signed[4:0] check_result);
@(negedge clk);
if(check_result!=C)begin
$display("there is somthing wrong @%t",$time);
error++;
end
else
correct++;

endtask
```

```
task call_reset;
reset=1;
checker_res(0);
reset=0;
endtask




endmodule
```

DO_FILE

```
vlib work
vlog ALU.v tb.sv  +cover -covercells
vsim -voptargs=+acc work.tb -cover
add wave *
coverage save ALU_tb.ucdb -onexit -du work.ALU_4_bit
run -all

coverage exclude -src ALU.v -line 26 -code s
coverage exclude -src ALU.v -line 26 -code b
```

COUNTER

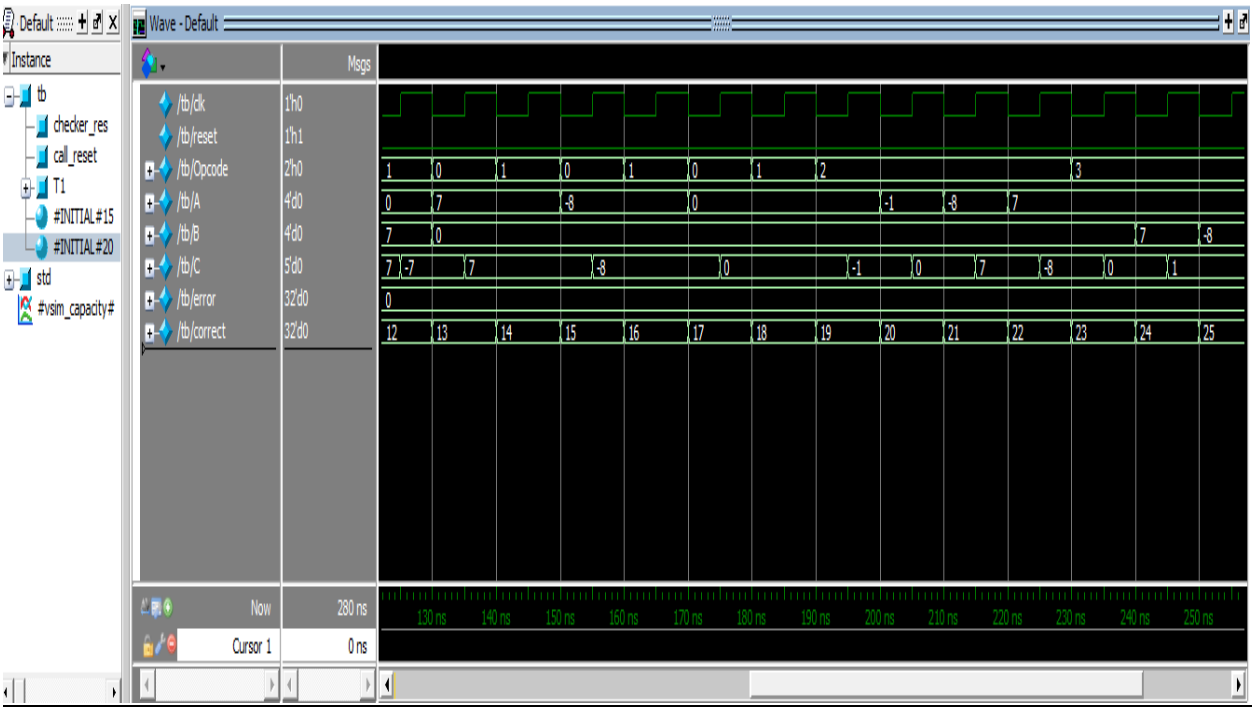
```
# Loading work.ALU_4_bit(fast)
# error_count=0  ------- correct_count=29
# ** Note: $stop    : tb.sv(132)
#    Time: 290 ns  Iteration: 1  Instance: /tb
```

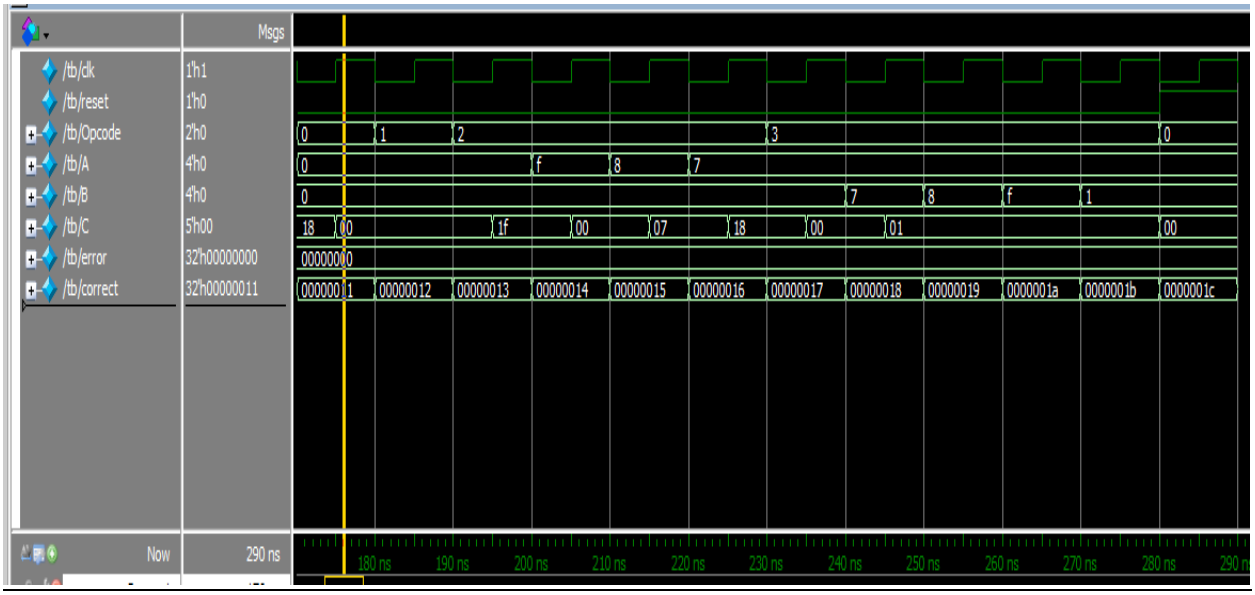
VERIFICATION PLAN

1	Label	Description	Stimulus Generation	Functionality Check
2	reset	When the reset is asserted, the output adder value should be low the we assert reset to low	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
3	ADD_1	When A is asserted to Max postive number and B asserted to Max negative number and OPcode is add the output C should be equal -1	Directed at the simulation at time 20	A checker in the testbench to make sure the output is correct
4	SUB_1	When A is asserted to Max postive number and B asserted to Max negative number and OPcode is SUB the output C should be equal 15	Directed at the simulation at time 30	A checker in the testbench to make sure the output is correct
5	ADD_2	When A is asserted to Max postive number and B asserted to Max posetive number and OPcode is ADD the output C should be equal 14	Directed at the simulation at time 30	A checker in the testbench to make sure the output is correct
6	SUB_2	When A is asserted to Max postive number and B asserted to Max posetive number and OPcode is SUB the output C should be equal 0	Directed at the simulation at time 40	A checker in the testbench to make sure the output is correct
7	ADD_3	When A is asserted to Max negative number and B asserted to Max negative number OP=ADD the output C should be equal -16	Directed at the simulation at time 50	A checker in the testbench to make sure the output is correct
8	SUB_3	When A is asserted to Max negative number and B asserted to Max negative number and OPcode =SUB the output C should be equal 0	Directed at the simulation at time 60	A checker in the testbench to make sure the output is correct
9	ADD_4	When A is asserted to Max negative number and B asserted to Max postive number OP=ADD the output C should be equal -1	Directed at the simulation at time 70	A checker in the testbench to make sure the output is correct
10	SUB_4	When A is asserted to Max negative number and B asserted to Max postive number OP=SUB the output C should be equal -15	Directed at the simulation at time 80	A checker in the testbench to make sure the output is correct
11	ADD_5	When A is asserted to Zero number and B asserted to Max negative number OP = ADD the output C should be equal -8	Directed at the simulation at time 90	A checker in the testbench to make sure the output is correct

12	SUB_5	When A is asserted to Zero number and B asserted to Max negative number OP = SUB the output C should be equal 8	Directed at the simulation at time 100	A checker in the testbench to make sure the output is correct
13	ADD_6	When A is asserted to zero number and B asserted to Max positive number OP=ADD the output C should be equal 7	Directed at the simulation at time 110	A checker in the testbench to make sure the output is correct
14	SUB_6	When A is asserted to zero number and B asserted to Max positive number OP=SUB the output C should be equal -7	Directed at the simulation at time 120	A checker in the testbench to make sure the output is correct
15	ADD_7	When A is asserted to Max positive number and B asserted to Zero number OP=ADD the output C should be equal 7	Directed at the simulation at time 130	A checker in the testbench to make sure the output is correct
16	SUB_7	When A is asserted to Max positive number and B asserted to Zero number OP=SUB the output C should be equal 7	Directed at the simulation at time 140	A checker in the testbench to make sure the output is correct
17	ADD_8	When A is asserted to Max negative number and B asserted to zero number OP = ADD the output C should be equal -8	Directed at the simulation at time 150	A checker in the testbench to make sure the output is correct
18	SUB_8	When A is asserted to Max negative number and B asserted to zero number OP = SUB the output C should be equal -8	Directed at the simulation at time 160	A checker in the testbench to make sure the output is correct
19	ADD_9	When A is asserted to zero number and B asserted to zero number OP=ADD the output C should be equal 0	Directed at the simulation at time 170	A checker in the testbench to make sure the output is correct
20	SUB_9	When A is asserted to zero number and B asserted to zero number OP=SUB the output C should be equal 0	Directed at the simulation at time 180	A checker in the testbench to make sure the output is correct
21	Not_A_1	When A is asserted to zero OP=NOT_A the output C should be equal -1	Directed at the simulation at time 190	A checker in the testbench to make sure the output is correct
22	Not_A_2	When A is asserted to -1 OP=NOT_A the output C should be equal 0	Directed at the simulation at time 200	A checker in the testbench to make sure the output is correct
23	Not_A_3	When A is asserted to MAXNEG OP=NOT_A the output C should be equal MAXPOS	Directed at the simulation at time 210	A checker in the testbench to make sure the output is correct
24	Not_A_4	When A is asserted to MAXPOS OP=NOT_A the output C should be equal MAXNEG	Directed at the simulation at time 220	A checker in the testbench to make sure the output is correct
25	Reduction_Or_1	When B is asserted to Zero OP=REDUCTION_OR the output C should be equal ZERO	Directed at the simulation at time 230	A checker in the testbench to make sure the output is correct
26	Reduction_Or_2	When B is asserted to MAXPOS OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 240	A checker in the testbench to make sure the output is correct
27	Reduction_Or_3	When B is asserted to MAXNEG OP=REDUCTION_OR the output C should be equal ONE	Directed at the simulation at time 250	A checker in the testbench to make sure the output is correct
28	Reduction_Or_4	When B is asserted to -1 OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 260	A checker in the testbench to make sure the output is correct
29	Reduction_Or_5	When B is asserted to 1 OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 270	A checker in the testbench to make sure the output is correct
30	RESET	When the reset is asserted, the output adder value should be low	Directed at the simulation at time 280	A checker in the testbench to make sure the output is correct
31	STOP	we end simulation here by $stop	Directed at the simulation at time 290	A checker in the testbench to make sure the output is correct

SIMULATION


Default
Instance
tb
checker_res
call_reset
T1
#INITIAL#15
#INITIAL#20
std
#vsim_capacity#
Wave - Default
Msgs
/tb/clk
/tb/reset
/tb/Opcode
/tb/A
/tb/B
/tb/C
/tb/error
/tb/correct
1'h0
1'h1
2'h0
4'd0
4'd0
5'd0
32'd0
32'd0
Now
280 ns
Cursor 1
0 ns
130 ns
140 ns
150 ns
160 ns
170 ns
180 ns
190 ns
200 ns
210 ns
220 ns
230 ns
240 ns
250 ns


Msgs
/tb/clk
/tb/reset
/tb/Opcode
/tb/A
/tb/B
/tb/C
/tb/error
/tb/correct
1'h1
1'h0
2'h0
4'h0
4'h0
5'h00
32'h00000000
32'h00000011
Now
290 ns
180 ns
190 ns
200 ns
210 ns
220 ns
230 ns
240 ns
250 ns
260 ns
270 ns
280 ns

COVERAGE REPORT

```
Coverage Report by file with details

================================================================================
=== File: ALU.v
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         0     100.0

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

Statement Coverage for file ALU.v --

    1                                           module ALU_4_bit(
    2                                               input  clk,
    3                                               input  reset,
    4                                               input  [1:0] Opcode,   // The opcode
    5                                               input  signed [3:0] A, // Input data A in 2's complement
    6                                               input  signed [3:0] B, // Input data B in 2's complement
    7
    8                                               output reg signed [4:0] C // ALU output in 2's complement
    9
    10                                                      );
    11
    12                                              reg signed [4:0]            Alu_out; // ALU output in 2's complement
    13
    14                                              localparam                  Add            = 2'b00; // A + B
    15                                              localparam                  Sub            = 2'b01; // A - B
    16                                              localparam                  Not_A               = 2'b10; // ~A
    17                                              localparam                  ReductionOR_B  = 2'b11; // |B
    18
    19                                              // Do the operation
    20              1                       29      always @* begin
    21                                                 case (Opcode)
    22              1                       11              Add:            Alu_out = A + B;
```

```
    11
    12                                              reg signed [4:0]            Alu_out; // ALU output in 2's complement
    13
    14                                              localparam                  Add            = 2'b00; // A + B
    15                                              localparam                  Sub            = 2'b01; // A - B
    16                                              localparam                  Not_A               = 2'b10; // ~A
    17                                              localparam                  ReductionOR_B  = 2'b11; // |B
    18
    19                                              // Do the operation
    20              1                       29      always @* begin
    21                                                 case (Opcode)
    22              1                       11              Add:            Alu_out = A + B;
    23              1                        9              Sub:            Alu_out = A - B;
    24              1                        4              Not_A:          Alu_out = ~A;
    25              1                        5              ReductionOR_B:  Alu_out = |B;
    26              1                        E            default:  Alu_out = 5'b0;
    27                                                 endcase
    28                                              end // always @ *
    29
    30                                              // Register output C
    31              1                       29      always @(posedge clk or posedge reset) begin
    32                                                 if (reset)
    33              1                        4           C <= 5'b0;
    34                                                 else
    35              1                       25           C<= Alu_out;
    36                                              end
    37
    38                                          endmodule
```

```
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                                      29     Count coming in to CASE
    22              1                       11              Add:            Alu_out = A + B;
    23              1                        9              Sub:            Alu_out = A - B;
    24              1                        4              Not_A:          Alu_out = ~A;
    25              1                        5              ReductionOR_B:  Alu_out = |B;
    26              1                        E            default:  Alu_out = 5'b0;
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    32                                      29     Count coming in to IF
    32              1                        4          if (reset)
    34              1                       25          else
Branch totals: 2 hits of 2 branches = 100.0%


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

```
    ----------------            ------      ----    ------ ---------
    FEC Expression Terms             0         0         0     100.0
FSM Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FSMs                                                       100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                      44        44         0     100.0
```

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

Toggle Coverage for File ALU.v --

       Line                                 Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          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]          1           1      100.00

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

Toggle Coverage      =      100.0% (44 of 44 bins)


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

DSP project

Design

Here we change mult_out to 37 bit because A_reg*add_out2=36bit and 1bit for carry

And change adder_out_stg reg to 18 bit to take carry from addition.

```
module DSP(A, B, C, D, clk, rst_n, P);
parameter OPERATION = "ADD";
input  [17:0] A, B, D;
input  [47:0] C;
input clk, rst_n;
output reg  [47:0] P;

reg  [17:0] A_reg_stg1, A_reg_stg2, B_reg, D_reg ;
reg  [18:0] adder_out_stg1, adder_out_stg2;
reg  [36:0] mult_out;
reg  [47:0] C_reg;
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;
        adder_out_stg1 <= 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
            adder_out_stg1 <= D_reg + B_reg;
            P <= mult_out + C_reg;
        end
        else if (OPERATION == "SUBTRACT") begin
            adder_out_stg1 <= D_reg - B_reg;
            P <= mult_out - C_reg;
        end
        mult_out <= A_reg_stg2 * adder_out_stg2;
    end
end

endmodule
```

TB

```
module tb();

parameter OPERATION = "ADD";
logic  [17:0] A, B, D;
logic  [47:0] C;
logic clk, rst_n;
logic  [47:0] P;


int error=0,correct=0;

initial begin
    clk=0;
    forever #5 clk=!clk;
end

DSP D1(.*);

initial begin
A=0;
B=0;
C=0;
D=0;
call_rst_n;

for(int i=0;i<30;i++)begin
    A=$random;
    B=$random;
    C=$random;
    D=$random;

    checker_res(((D+B)*A)+C);
end

call_rst_n;
$display("error_count=%0d  ------- correct_count=%0d",error,correct);
$stop;

end


task checker_res(input logic signed[47:0] check_result);
repeat(5) @(negedge clk);
if(check_result!=P)begin
$display("there is somthing wrong @%t",$time);
error++;
end
else
correct++;

endtask

task call_rst_n;
rst_n=0;
checker_res(0);
rst_n=1;
endtask

endmodule
```

VERIFICATION PLAN

1	Label	Description	Stimulus Generation	Functionality Check
2	DSP_1	When the reset is asserted,all the output port adn internal reg should be equal zero	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
3	DSP_2	here we randomize on i/p A,B,C,D and check output P by Task checker for 30 times	Directed in TB at time 50 ns unti 1550 ns	we check in testbench with task checker by send expected result and compare

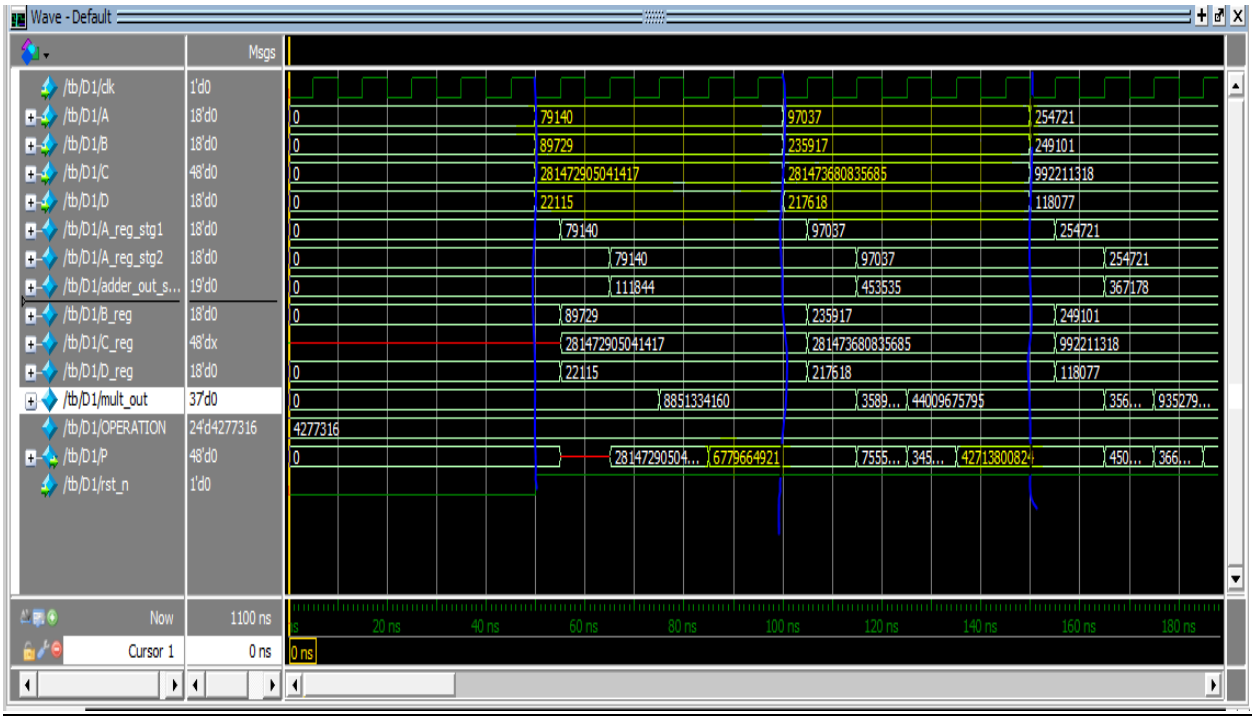
DO FILE

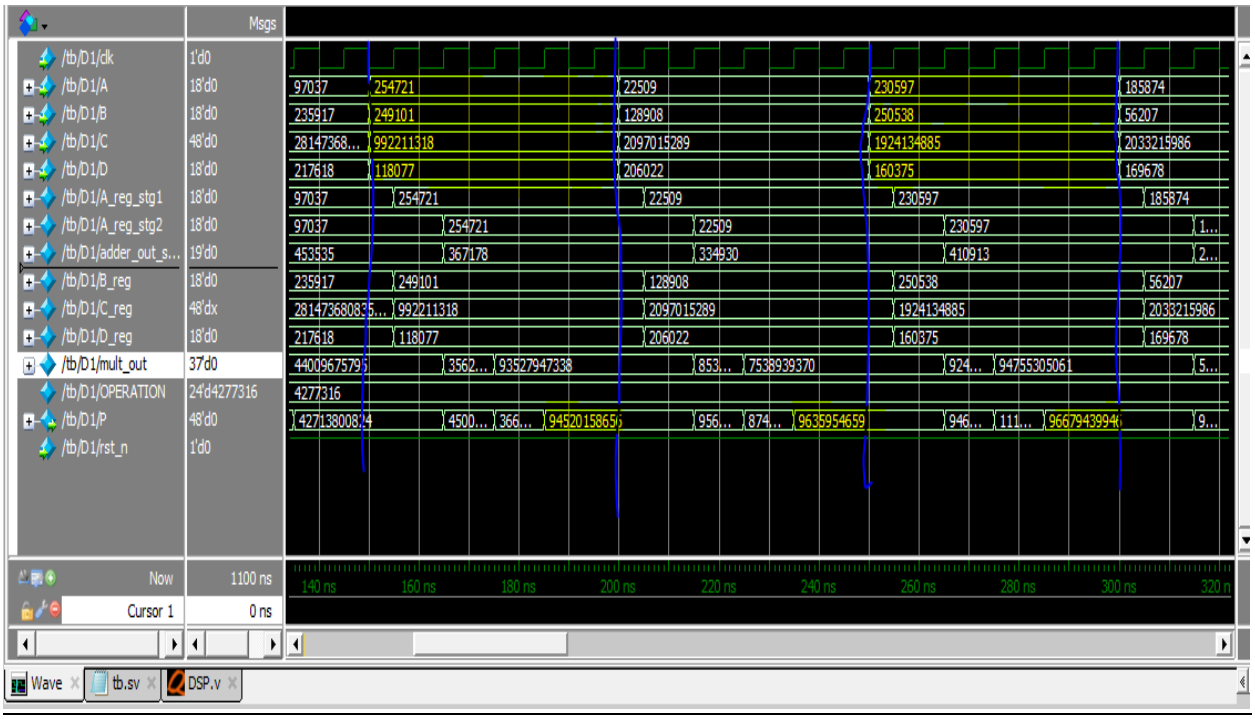
```
vlib work
vlog DSP.v tb.sv  +cover -covercells
vsim -voptargs=+acc work.tb -cover
add wave *
coverage save DSP_tb.ucdb -onexit -du work.DSP
run -all
```

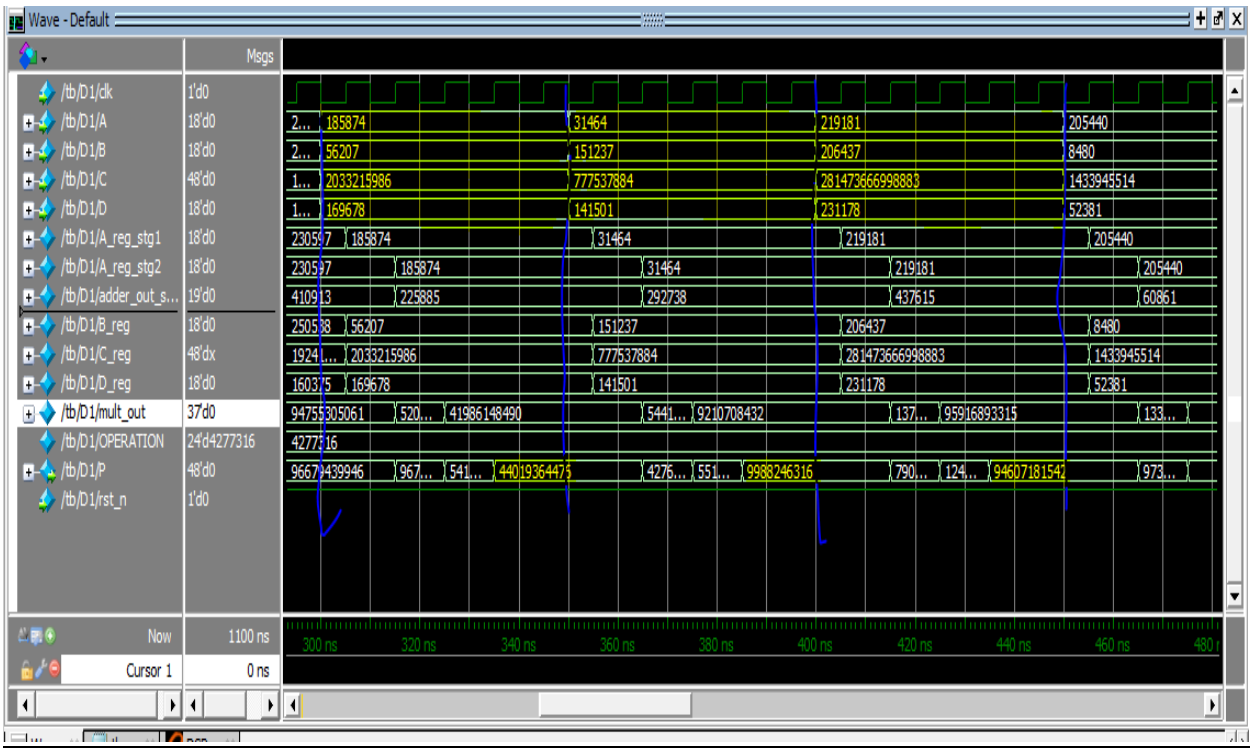
COUNTER

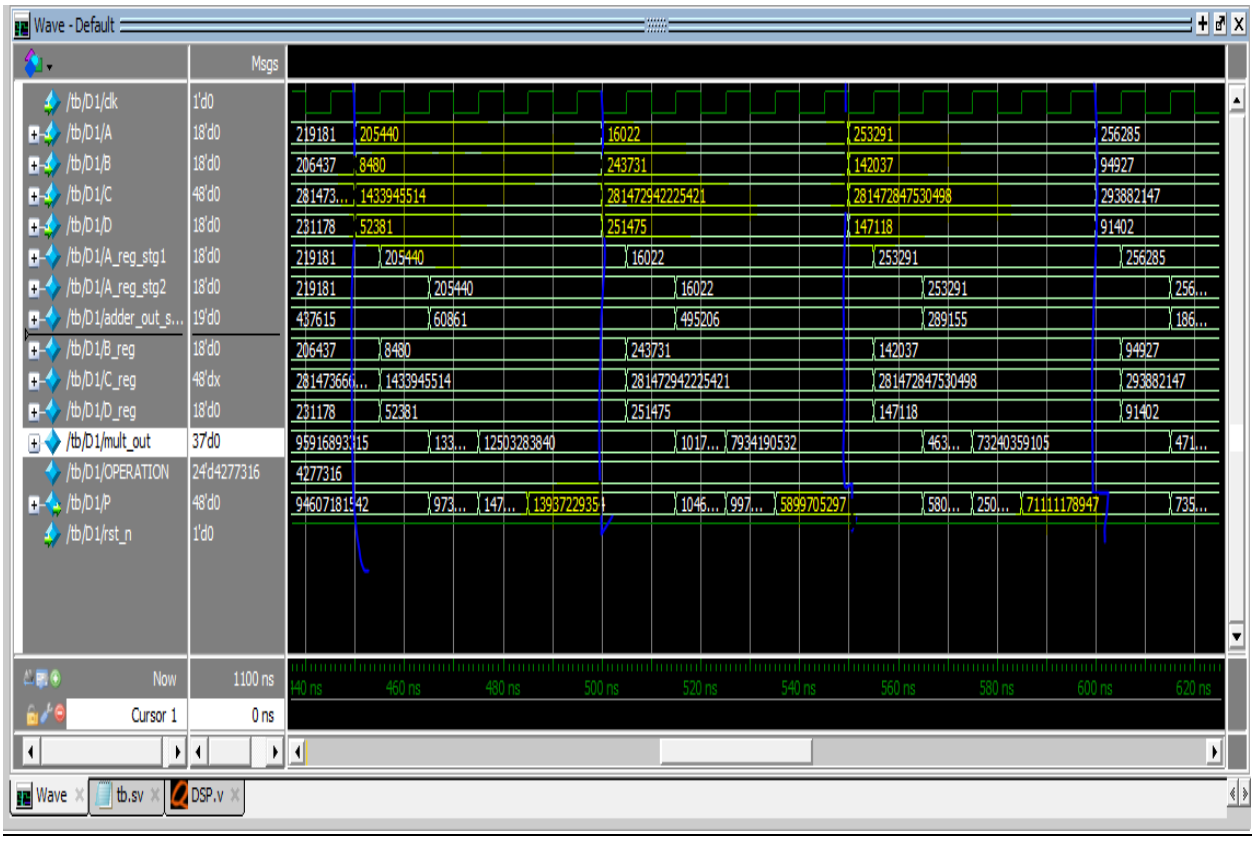
```
# Loading work.tb(fast)
# error_count=0  ------- correct_count=22
# ** Note: $stop    : tb.sv(37)
#    Time: 1100 ns  Iteration: 1  Instance: /tb
```

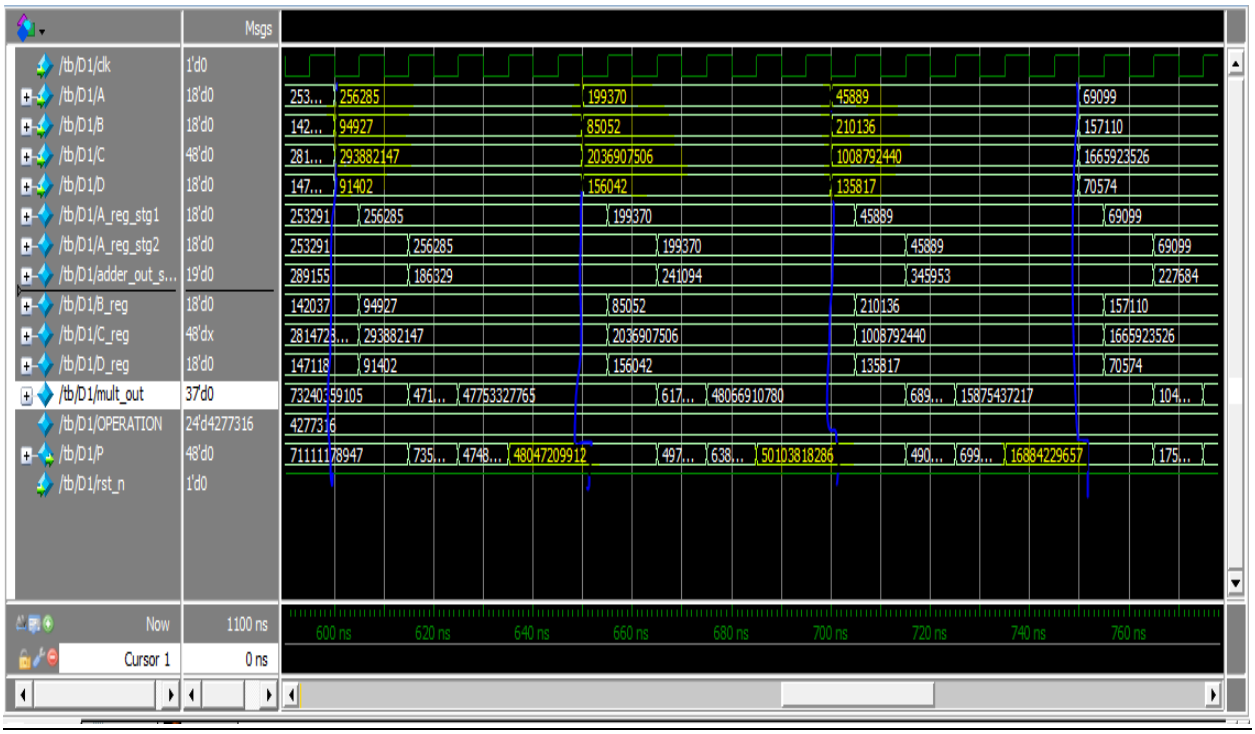
SIMULATION

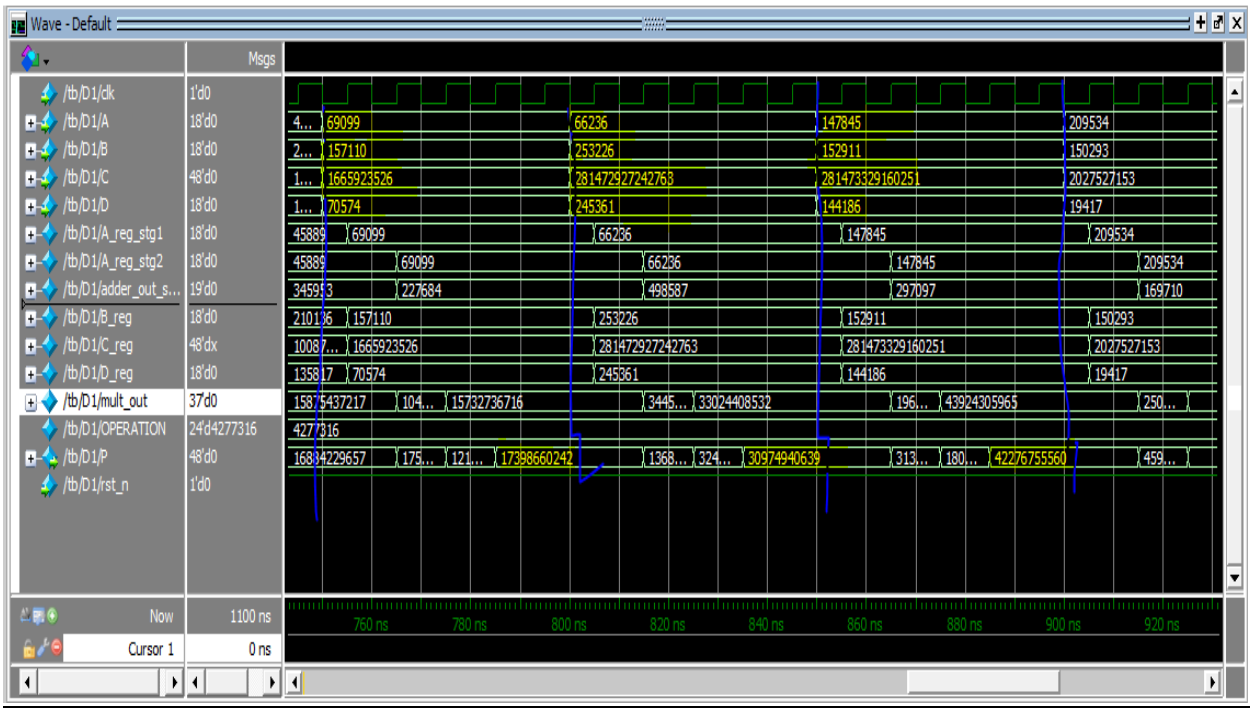

Wave - Default
Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns

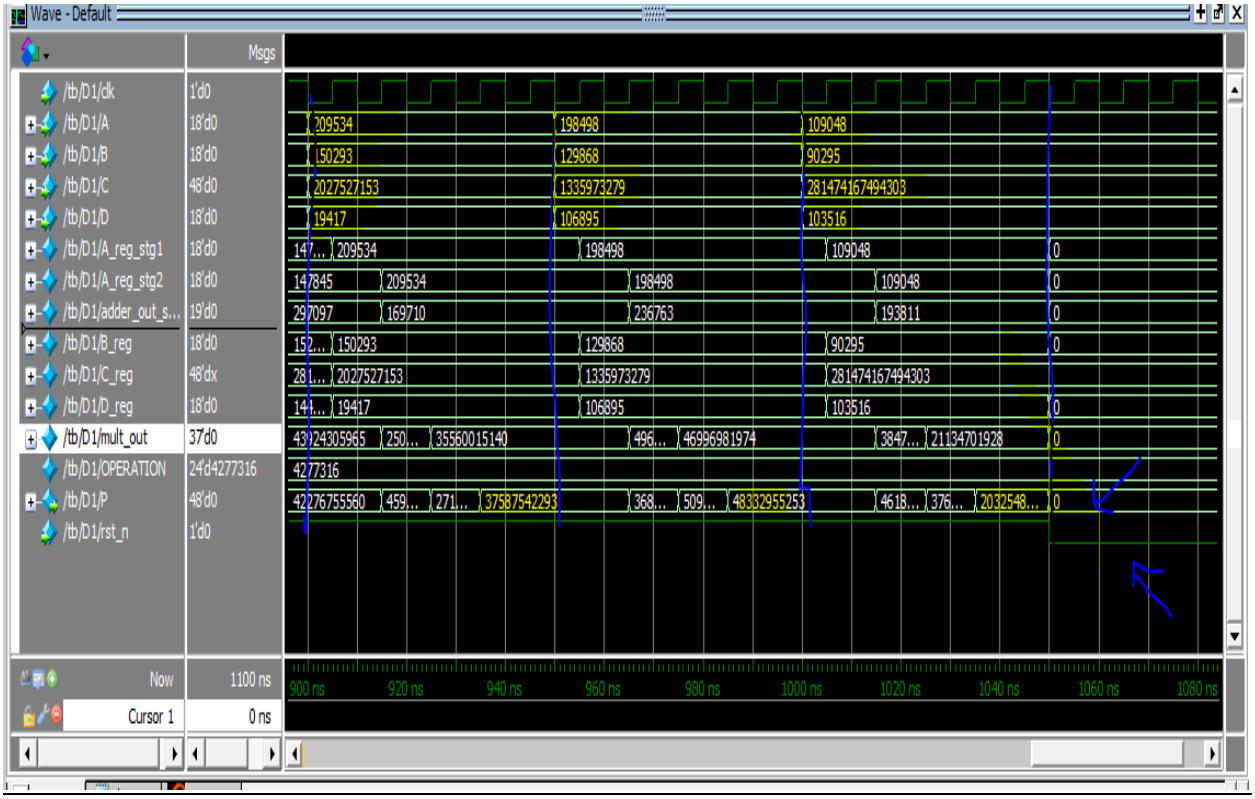

Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
140 ns
160 ns
180 ns
200 ns
220 ns
240 ns
260 ns
280 ns
300 ns
Wave
tb.sv
DSP.v


Wave - Default
Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
300 ns
320 ns
340 ns
360 ns
380 ns
400 ns
420 ns
440 ns
460 ns


Wave - Default
Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
460 ns
480 ns
500 ns
520 ns
540 ns
560 ns
580 ns
600 ns
620 ns
Wave
tb.sv
DSP.v


Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
600 ns
620 ns
640 ns
660 ns
680 ns
700 ns
720 ns
740 ns
760 ns


Wave - Default
Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
Now
1100 ns
Cursor 1
0 ns
760 ns
780 ns
800 ns
820 ns
840 ns
860 ns
880 ns
900 ns
920 ns


Wave - Default
Msgs
/tb/D1/clk
/tb/D1/A
/tb/D1/B
/tb/D1/C
/tb/D1/D
/tb/D1/A_reg_stg1
/tb/D1/A_reg_stg2
/tb/D1/adder_out_s...
/tb/D1/B_reg
/tb/D1/C_reg
/tb/D1/D_reg
/tb/D1/mult_out
/tb/D1/OPERATION
/tb/D1/P
/tb/D1/rst_n
1'd0
18'd0
18'd0
48'd0
18'd0
18'd0
18'd0
19'd0
18'd0
48'dx
18'd0
37'd0
24'd4277316
48'd0
1'd0
209534
198498
109048
150293
129868
90295
2027527153
1335973279
281474167494303
19417
106895
103516
147845
297097
169710
236763
193811
43924305965
35560015140
46996981974
21134701928
4277316
42276755560
37587542293
48332955253
2032548...
Now
1100 ns
Cursor 1
0 ns
900 ns
920 ns
940 ns
960 ns
980 ns
1000 ns
1020 ns
1040 ns
1060 ns
1080 ns

COVERAGE REPORT :STATEMENT

```
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           17        17         0     100.0

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

Statement Coverage for file DSP.v --
NOTE: The modification timestamp for source file 'DSP.v' has been altered since compilation.

    1                                           module DSP(A, B, C, D, clk, rst_n, P);
    2                                           parameter OPERATION = "ADD";
    3                                           input  [17:0] A, B, D;
    4                                           input  [47:0] C;
    5                                           input clk, rst_n;
    6                                           output reg  [47:0] P;
    7
    8                                           reg  [17:0] A_reg_stg1, A_reg_stg2, B_reg, D_reg ;
    9                                           reg  [18:0] adder_out_stg1, adder_out_stg2;
    10                                          reg  [36:0] mult_out;
    11              1                   154     reg  [47:0] C_reg;
    12                                          always @(posedge clk or negedge rst_n) begin
    13                                             if (!rst_n) begin
    14              1                     4                 // reset
    15              1                     4                  A_reg_stg1 <= 0;
    16              1                     4                  A_reg_stg2 <= 0;
    17              1                     4                  B_reg <= 0;
    18              1                     4                  D_reg <= 0;
    19              1                     4                  adder_out_stg1 <= 0;
    20              1                     4                  mult_out <= 0;
    21                                                       P <= 0;
    22                                             end
    23              1                   150        else begin
    24              1                   150                 A_reg_stg1 <= A;
    25              1                   150                 A_reg_stg2 <= A_reg_stg1;
    26              1                   150                 B_reg <= B;
    27              1                   150                 C_reg <= C;
    28              1                   150                 D_reg <= D;
    29                                                      adder_out_stg2 <= adder_out_stg1;
    30
    31
    32              1                   150                 if (OPERATION == "ADD") begin
    33              1                   150                         adder_out_stg1 <= D_reg + B_reg;
    34                                                              P <= mult_out + C_reg;
```

```
    34                                                              P <= mult_out + C_reg;
    35                                                     end
    36                                                     else if (OPERATION == "SUBTRACT") begin
    37                                                             adder_out_stg1 <= D_reg - B_reg;
    38                                                             P <= mult_out - C_reg;
    39              1                   150                end
    40                                                     mult_out <= A_reg_stg2 * adder_out_stg2;
    41                                                end
    42                                             end
    43
    44                                             endmodule

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

BRANCH

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

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

Branch Coverage for file DSP.v --
NOTE: The modification timestamp for source file 'DSP.v' has been altered since compilation.

------------------------------------IF Branch------------------------------------
    12                                      154     Count coming in to IF
    12              1                         4     always @(posedge clk or negedge rst_n) begin
    22              1                       150        end
Branch totals: 2 hits of 2 branches = 100.0%


Condition Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Condition Terms              0         0         0     100.0
Expression Coverage:
    Enabled Coverage            Active   Covered    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FEC Expression Terms             0         0         0     100.0
FSM Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    FSMs                                                        100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                    688       688         0     100.0
```

TOGGLE

```
        Transitions                 0         0         0     100.0
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                    688       688         0     100.0
```

D_FF PROJECT

Design

Here I change on design on line 14 by assert output to same output which mean no change

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

VERIFICATION PLAN

Label	Description	Stimulus Generation	Functionality Check
dFF_1	When the reset is asserted the output port q =0	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
dFF_2	Here we check first on USE_EN=1 and check on all possible 4 i/p for d and en 1-d=0 ,en=0 2-d=1 ,en=1 3-d=1 ,en=0 4-d=0 ,en=1	Directed in TB_1 at time 10 ns until 50 ns	we check in testbench with task checker by send expected result and compare with output q
dFF_2_1	When the reset is asserted the output port q =0	Directed at the start of the simulation	A checker in the testbench to make sure the output is correct
dFF2_2	Here we check on USE_EN=0 and check on all possible 4 i/p for d and en 1-d=0 ,en=0 2-d=1 ,en=1 3-d=1 ,en=0 4-d=0 ,en=1	Directed in TB_2 at time 10 ns until 50 ns	we check in testbench with task checker by send expected result and compare with output q

TB_1

```
module tb_1();

parameter USE_EN = 1;
logic clk=0, rst, d, en;
logic q;

int correct=0,error=0;
always #5 clk=~clk;

dff #(.USE_EN(USE_EN)) DUT(.*);

initial begin
d=0;
en=0;
call_reset;
d=0;
en=0;
check_resul(q);
d=1;
en=1;
check_resul(1);
d=1;
en=0;
check_resul(q);
d=0;
en=1;
check_resul(0);
call_reset;
$display("error_count=%0d  ------- correct_count=%0d",error,correct);
$stop;
end
```

```
task check_resul(input logic check_result);
@(negedge clk);
if(check_result!=q)begin
$display("there is somthing wrong @%t  check_result =%0d   q=%0d  ",$time,check_result,q);
error++;
end
else
correct++;

endtask

task call_reset;
rst=1;
check_resul(0);
rst=0;
endtask


endmodule

```

TB_2

```
1   module tb_2();
2
3   parameter USE_EN = 0;
4   logic clk=0, rst, d, en;
5   logic q;
6
7   int correct=0,error=0;
8   always #5 clk=~clk;
9
10  dff #(.USE_EN(USE_EN)) DUT(.*);
11
12  initial begin
13
14  d=0;
15  en=0;
16  call_reset;
17  d=0;
18  en=0;
19  check_resul(q);
20  d=1;
21  en=1;
22  check_resul(q);
23  d=1;
24  en=0;
25  check_resul(q);
26  d=0;
27  en=1;
28  check_resul(q);
29  call_reset;
30  $display("error_count=%0d  ------- correct_count=%0d",error,correct);
31  $stop;
32  end
```

```
34  task check_resul(input logic check_result);
35  @(negedge clk);
36  if(check_result!=q)begin
37  $display("there is somthing wrong @%t  check_result =%0d   q=%0d  ",$time,check_result,q);
38  error++;
39  end
40  else
41  correct++;
42
43  endtask
44
45  task call_reset;
46  rst=1;
47  check_resul(0);
48  rst=0;
49  endtask
50
51
52  endmodule
53
```

COUNTER FOR FIRST_TB

```
# error_count=0   ------- correct_count=6
# ** Note: $stop    : tb.sv(37)
#    Time: 60 ns  Iteration: 1  Instance: /tb_1
# Break in Module tb_1 at tb.sv line 37
```

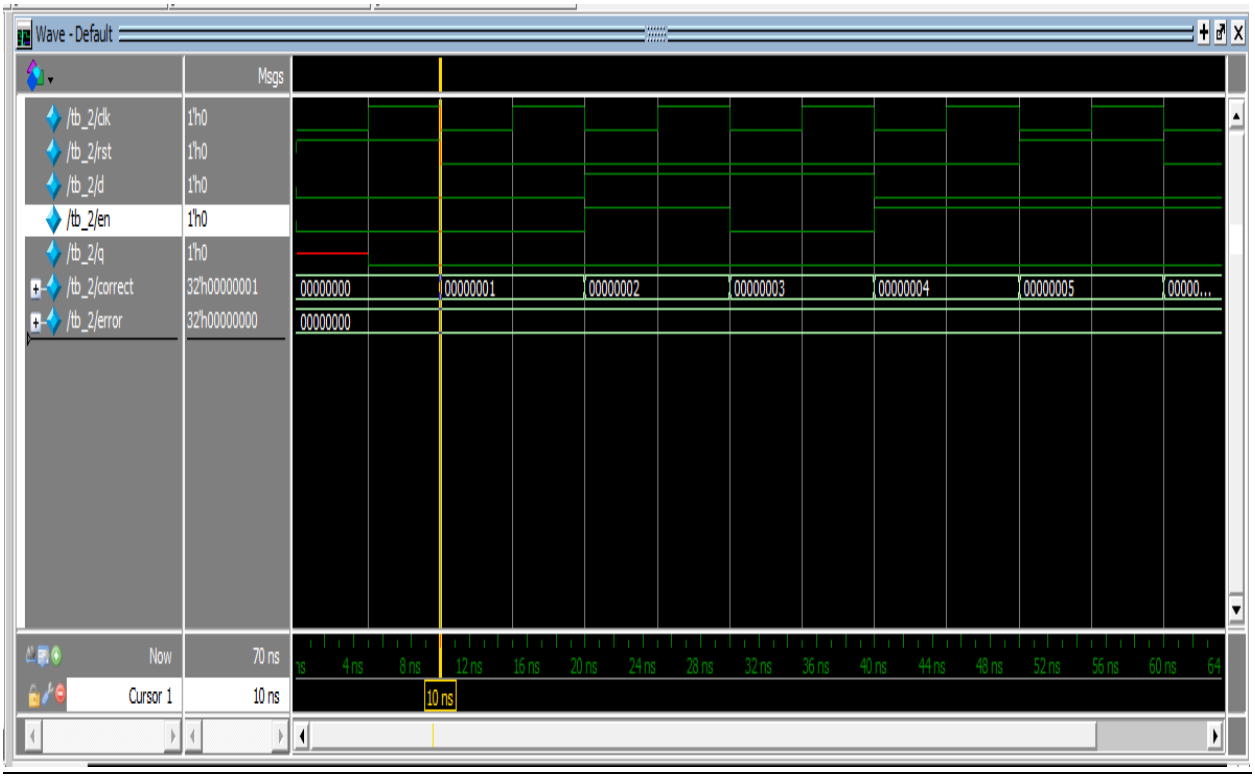
COUNTER FOR SECOND_TB

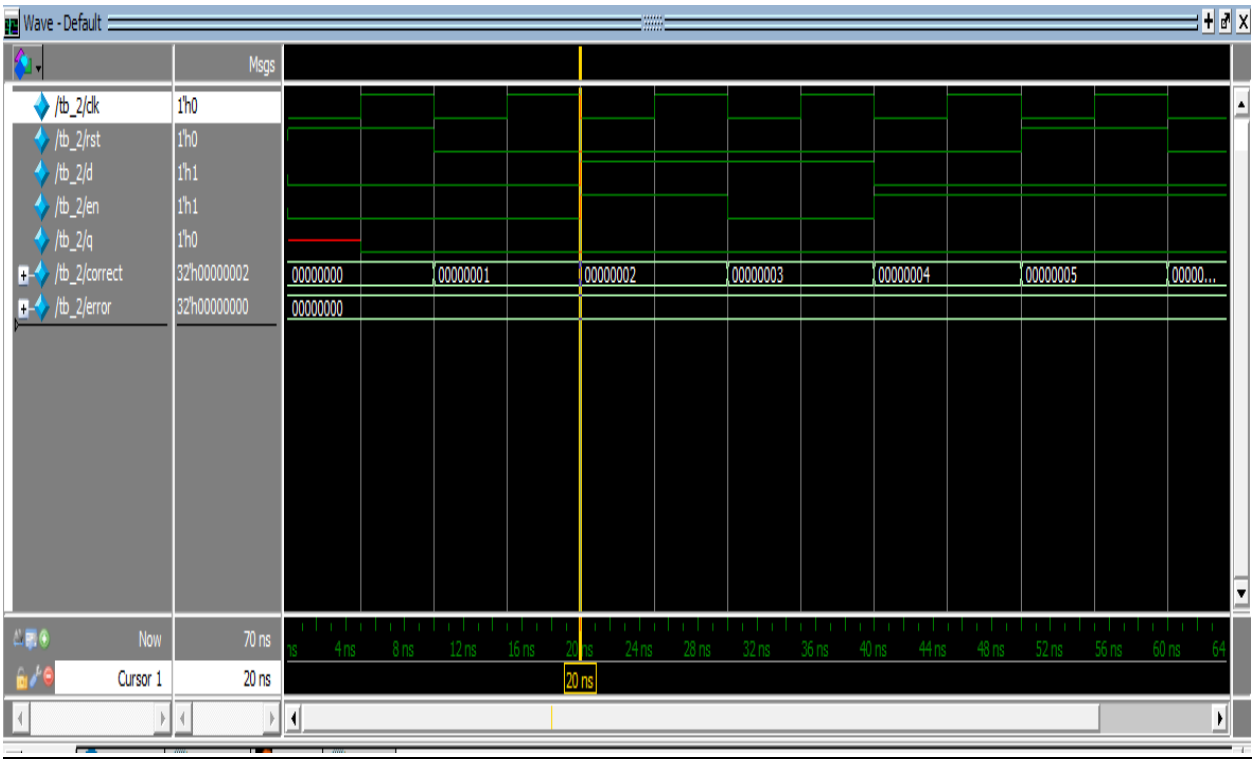
```
# error_count=0   ------- correct_count=6
# ** Note: $stop    : tb_2.sv(37)
#    Time: 60 ns  Iteration: 1  Instance: /tb_2
# Break in Module tb_2 at tb_2.sv line 37
```

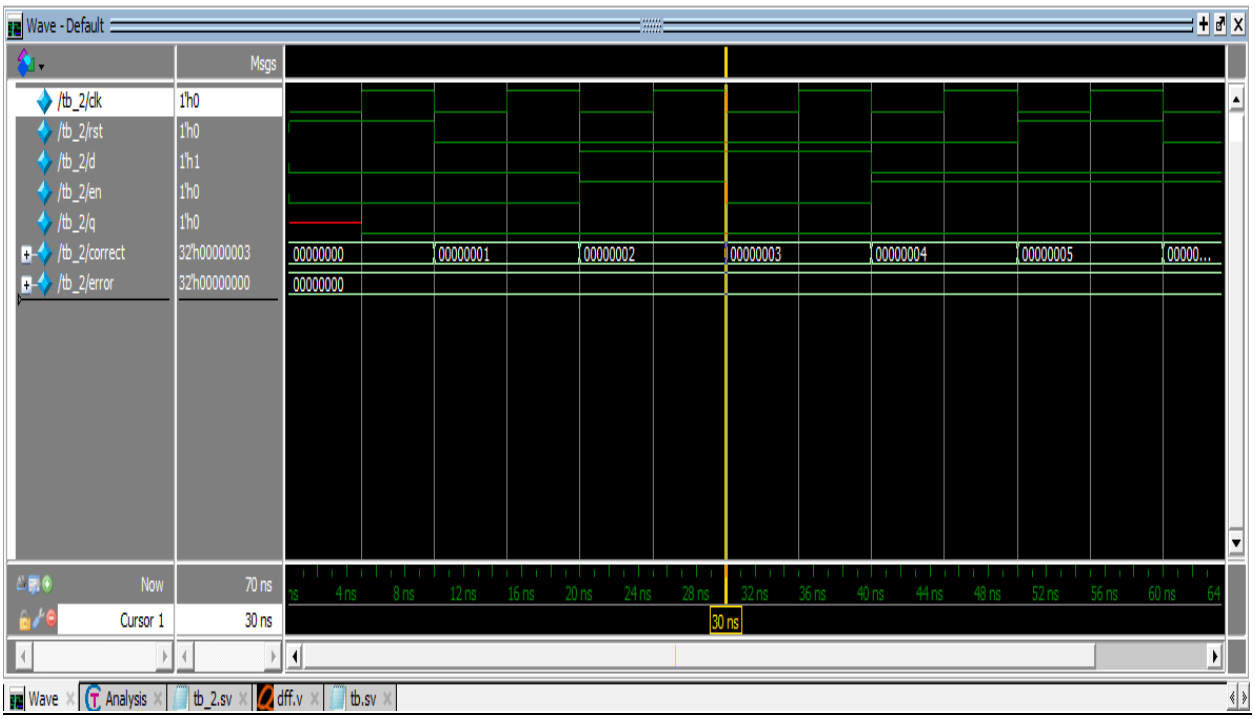
DO_FILE

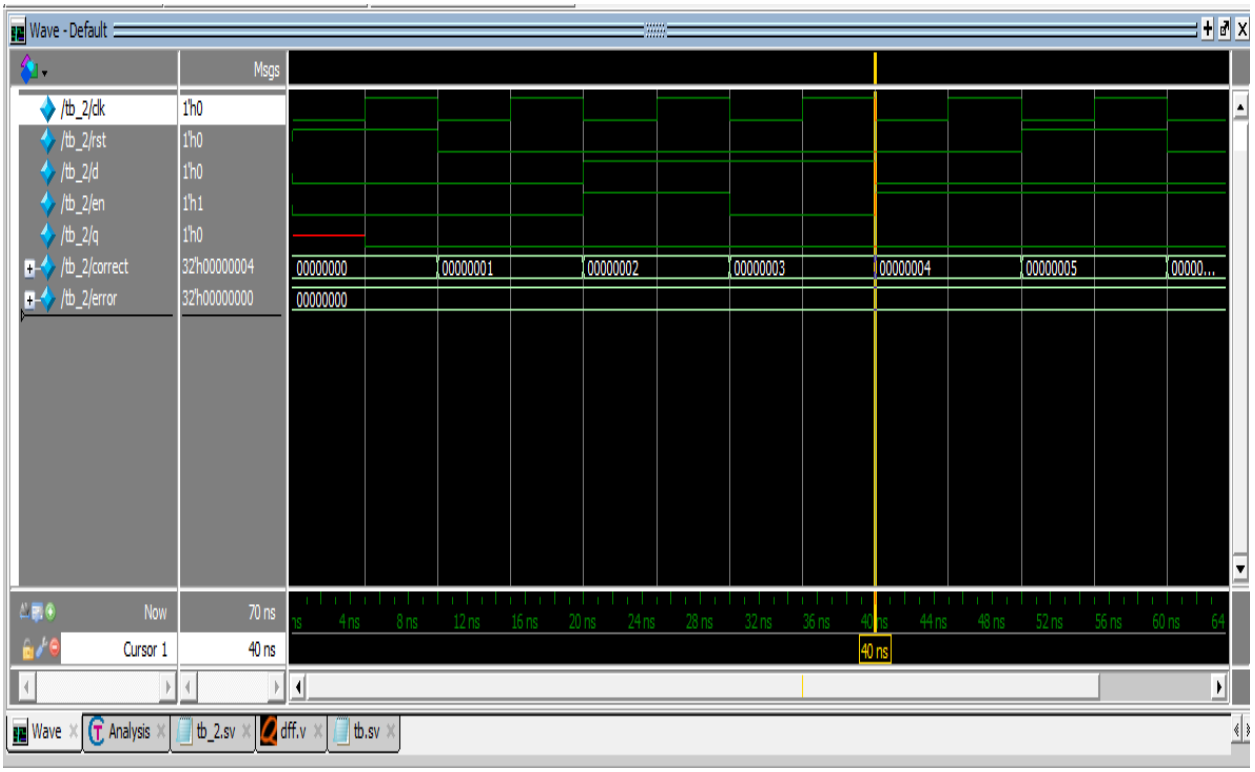
```
vlib work
vlog dff.v tb.sv +cover -covercells
vsim -voptargs=+acc work.tb_1 -cover
add wave *
coverage save dff_t1.ucdb -onexit -du work.dff
run -all
quit -sim

vlog dff.v tb_2.sv  +cover -covercells
vsim -voptargs=+acc work.tb_2 -cover
add wave *
coverage save dff_t2.ucdb -onexit -du work.dff
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_report.txt
```

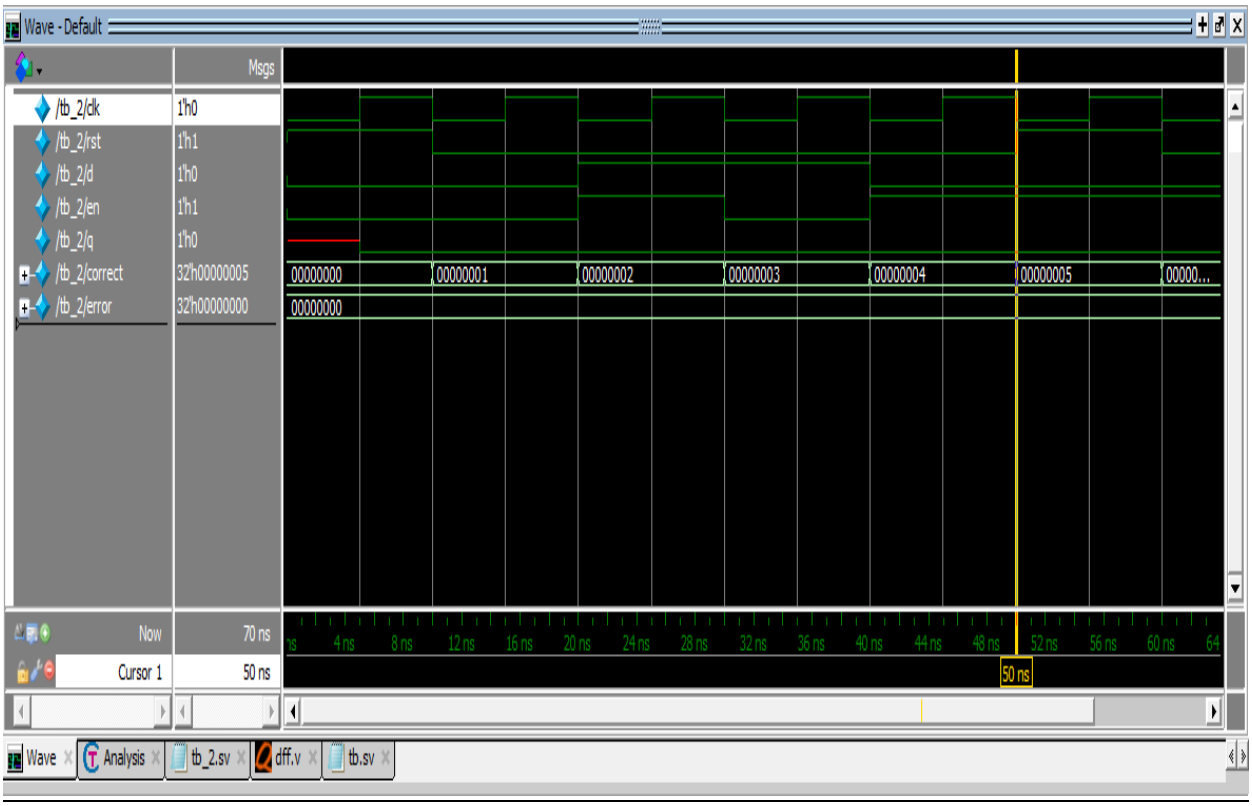
SIMULATION FOR tb1

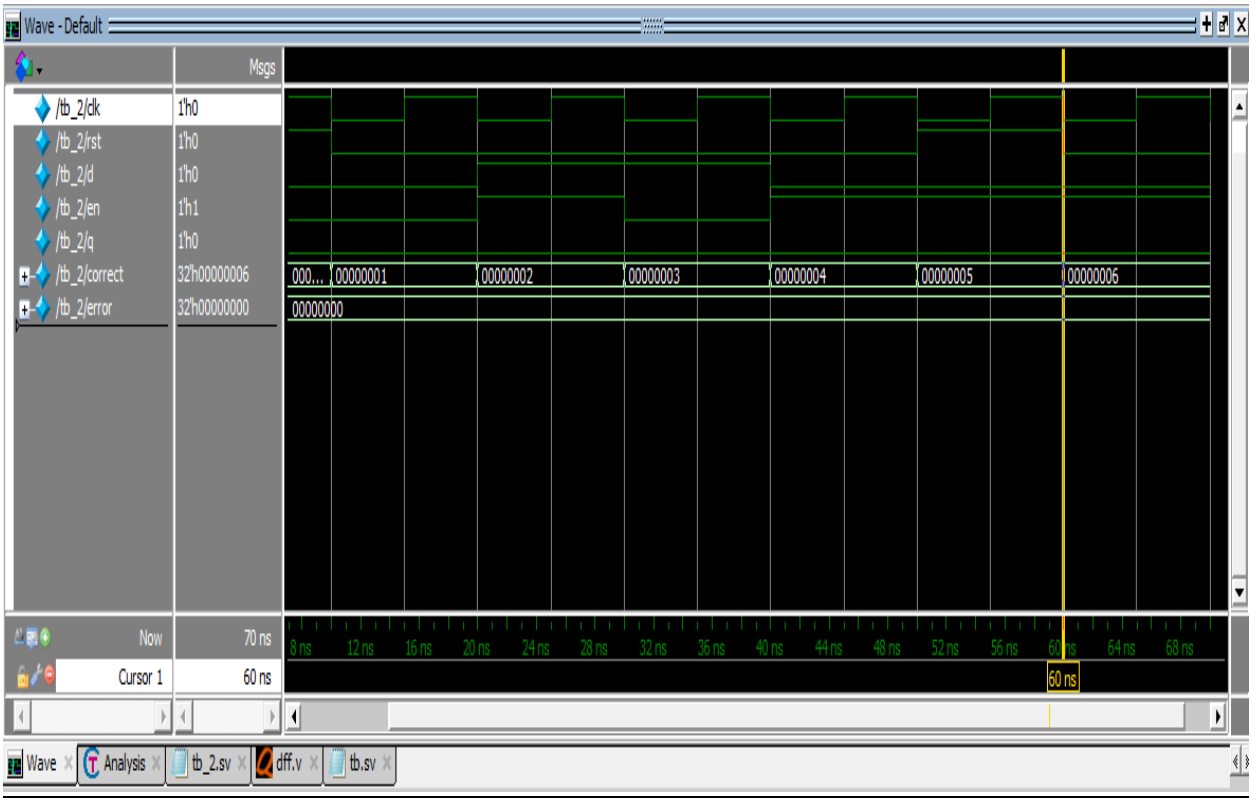




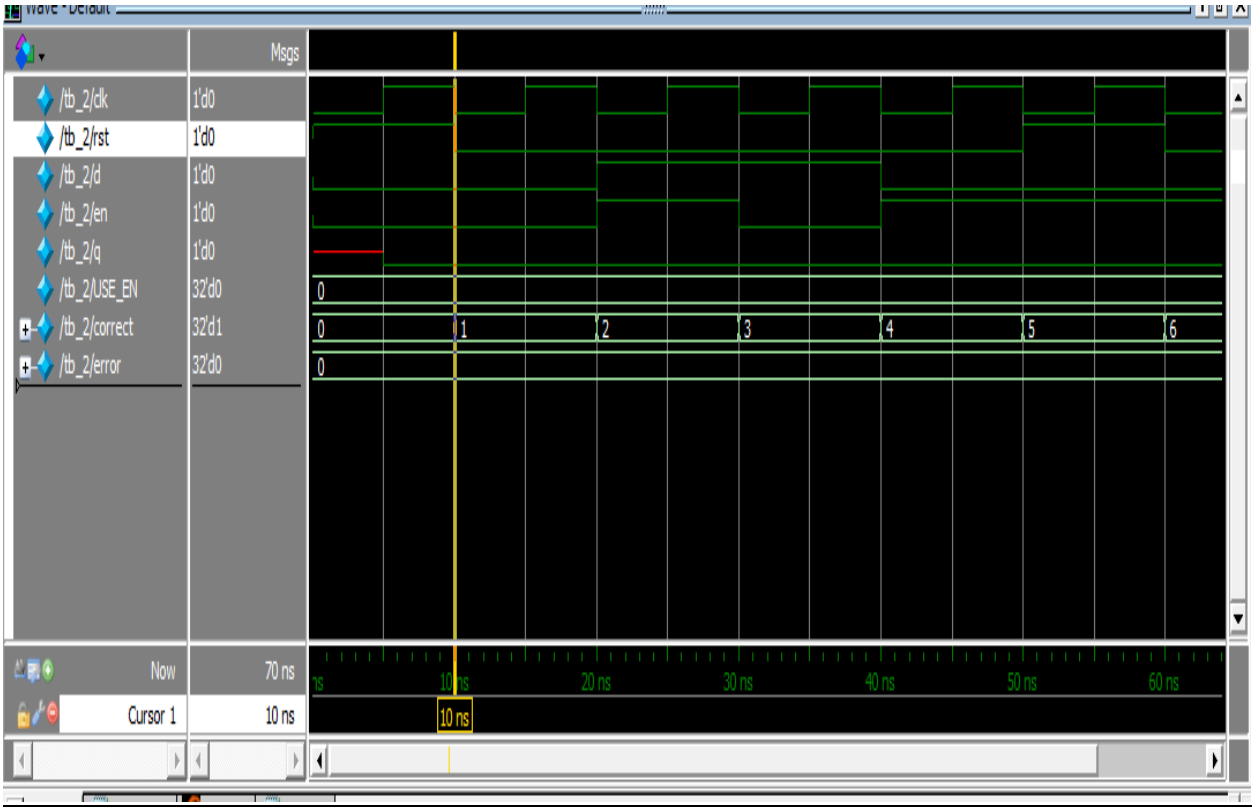

Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/correct
/tb_2/error
1'h0
1'h0
1'h1
1'h0
1'h0
32'h00000003
32'h00000000
00000000
00000001
00000002
00000003
00000004
00000005
00000...
00000000
Now
70 ns
Cursor 1
30 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
60 ns
64
30 ns
Wave
Analysis
tb_2.sv
dff.v
tb.sv

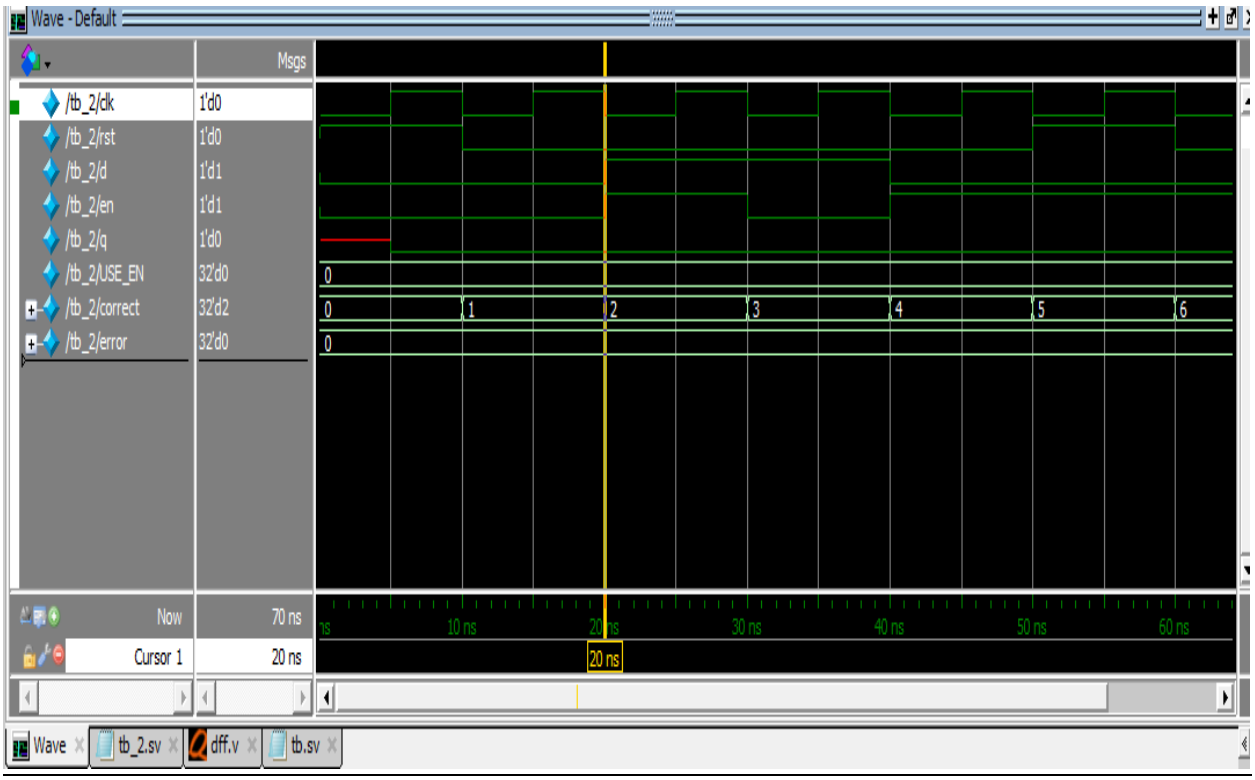

Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/correct
/tb_2/error
1'h0
1'h0
1'h0
1'h1
1'h0
32'h00000004
32'h00000000
00000000
00000001
00000002
00000003
00000004
00000005
00000...
00000000
Now
70 ns
Cursor 1
40 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
60 ns
64
40 ns
Wave
Analysis
tb_2.sv
dff.v
tb.sv

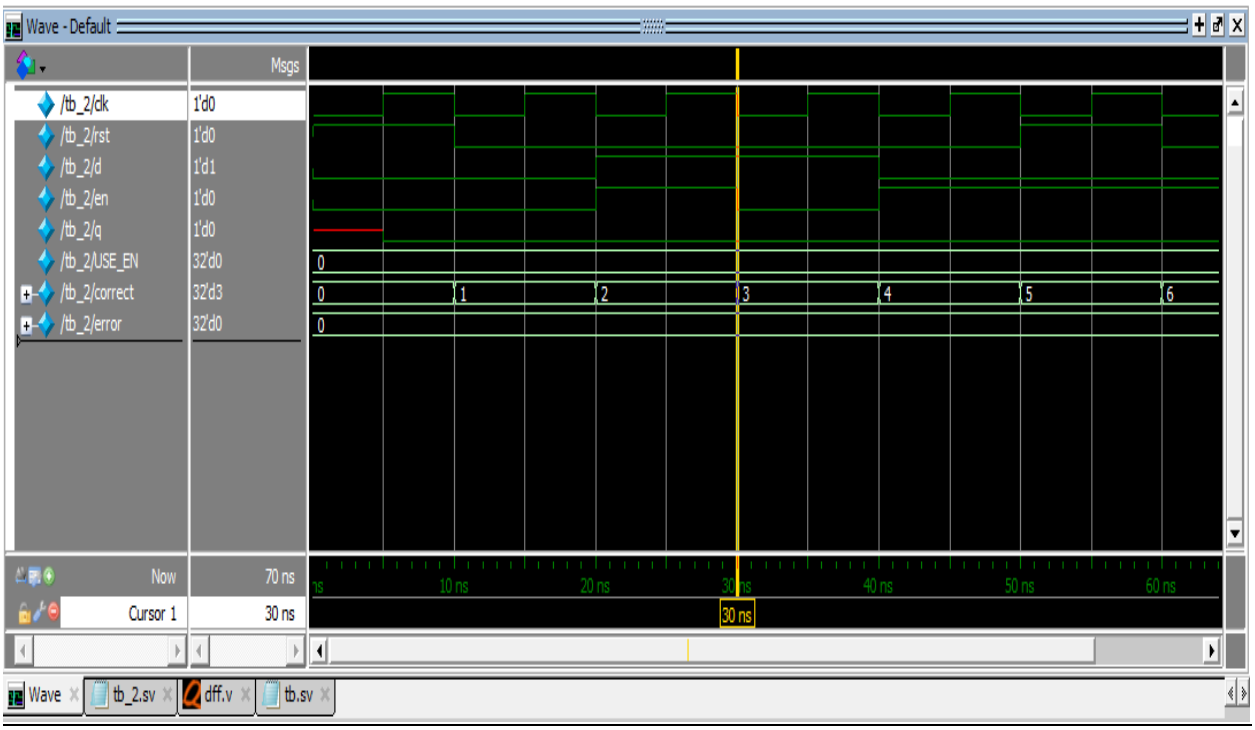

Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/correct
/tb_2/error
1'h0
1'h1
1'h0
1'h1
1'h0
32'h00000005
32'h00000000
00000000
00000001
00000002
00000003
00000004
00000005
00000...
00000000
Now
70 ns
Cursor 1
50 ns
4 ns
8 ns
12 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
60 ns
64
50 ns
Wave
Analysis
tb_2.sv
dff.v
tb.sv

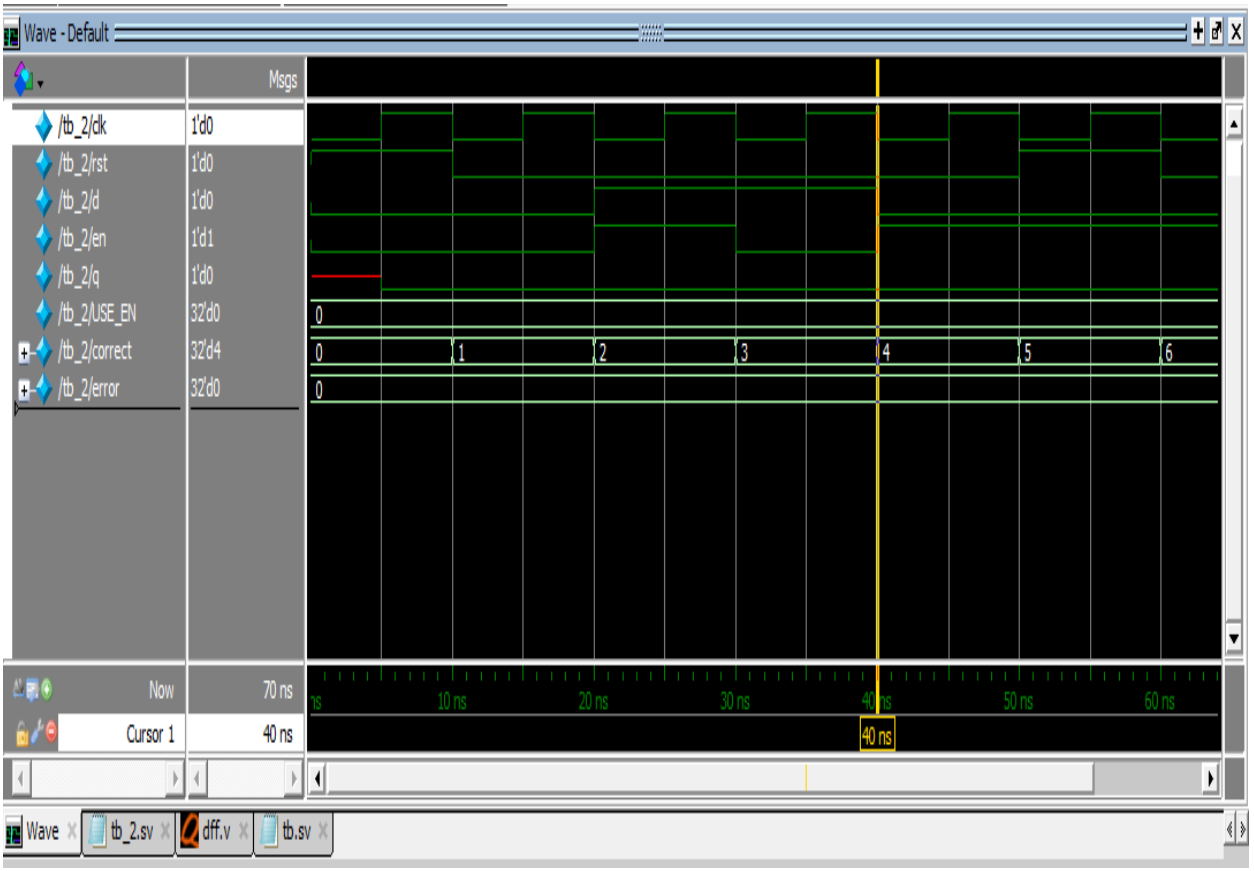

Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/correct
/tb_2/error
1'h0
1'h0
1'h0
1'h1
1'h0
32'h00000006
32'h00000000
000...
00000001
00000002
00000003
00000004
00000005
00000006
00000000
Now
70 ns
Cursor 1
60 ns
8 ns
12 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
64 ns
68 ns
60 ns
Wave
Analysis
tb_2.sv
dff.v
tb.sv

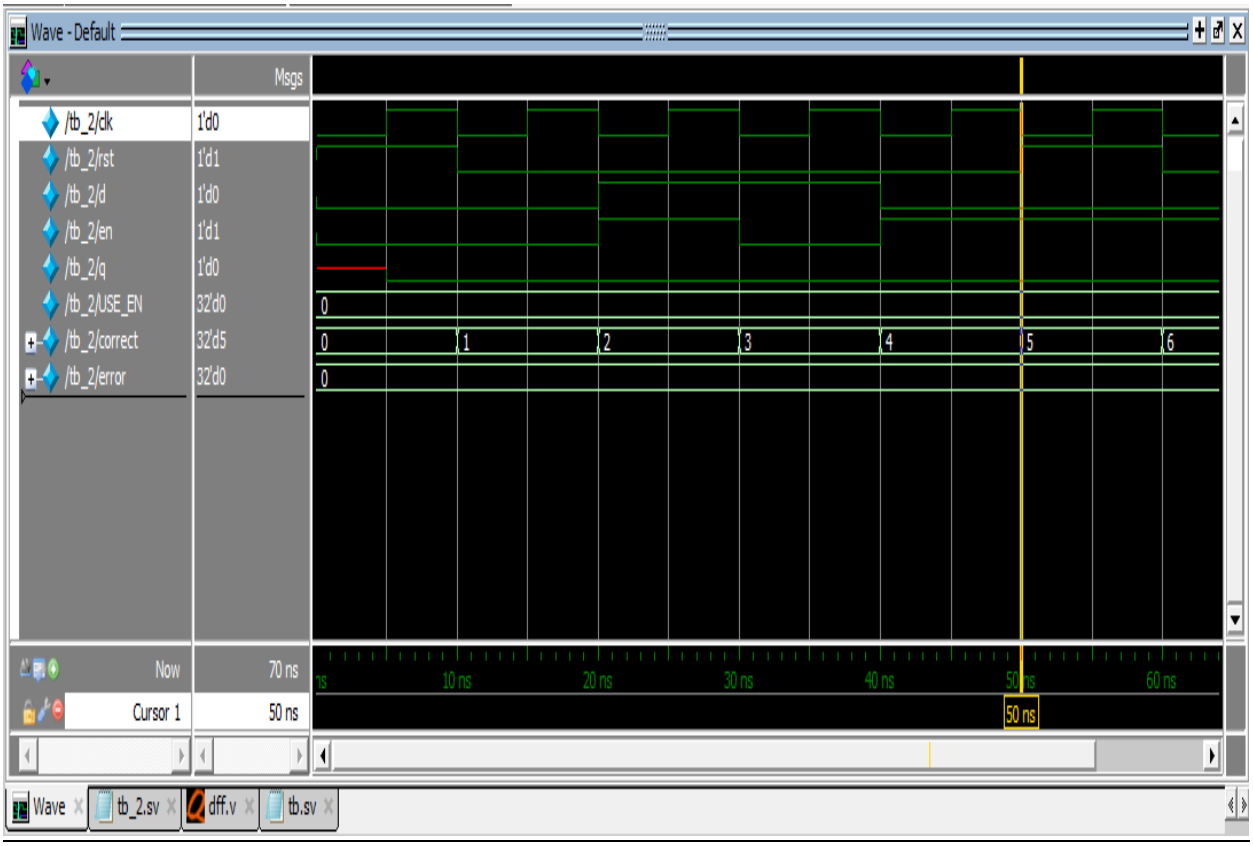
SIMULATION FOR tb2

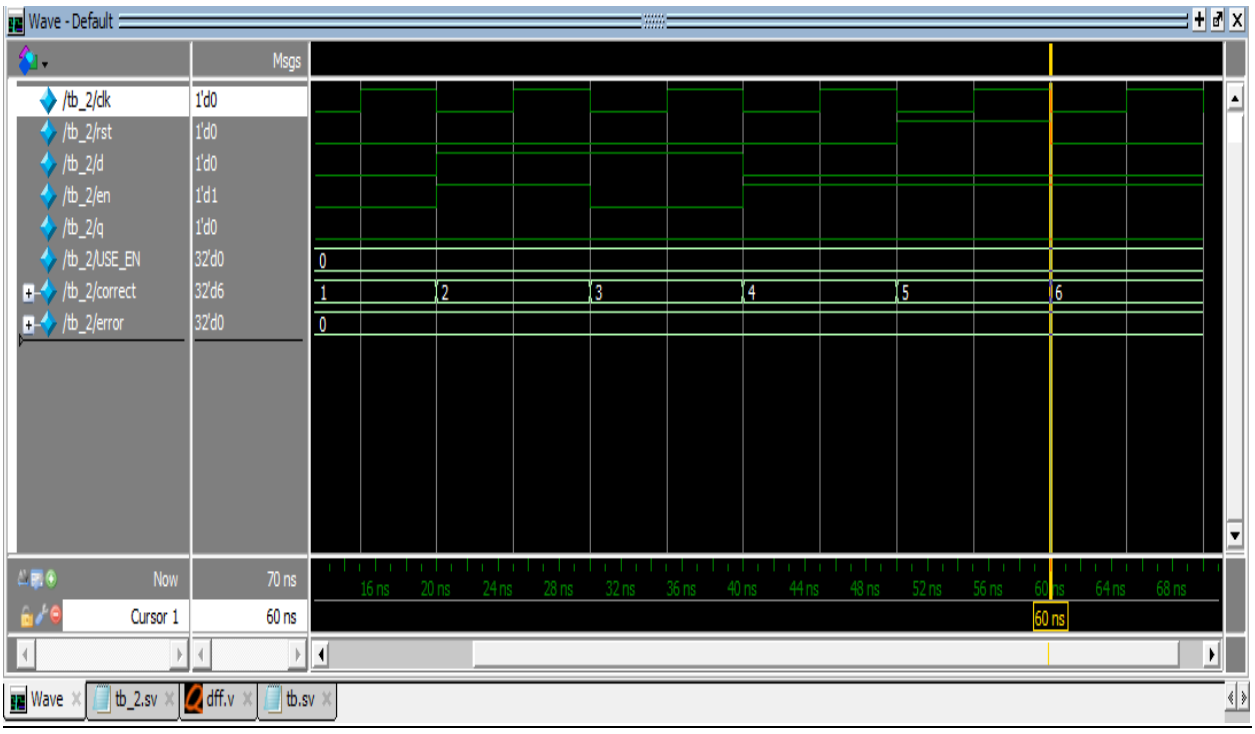


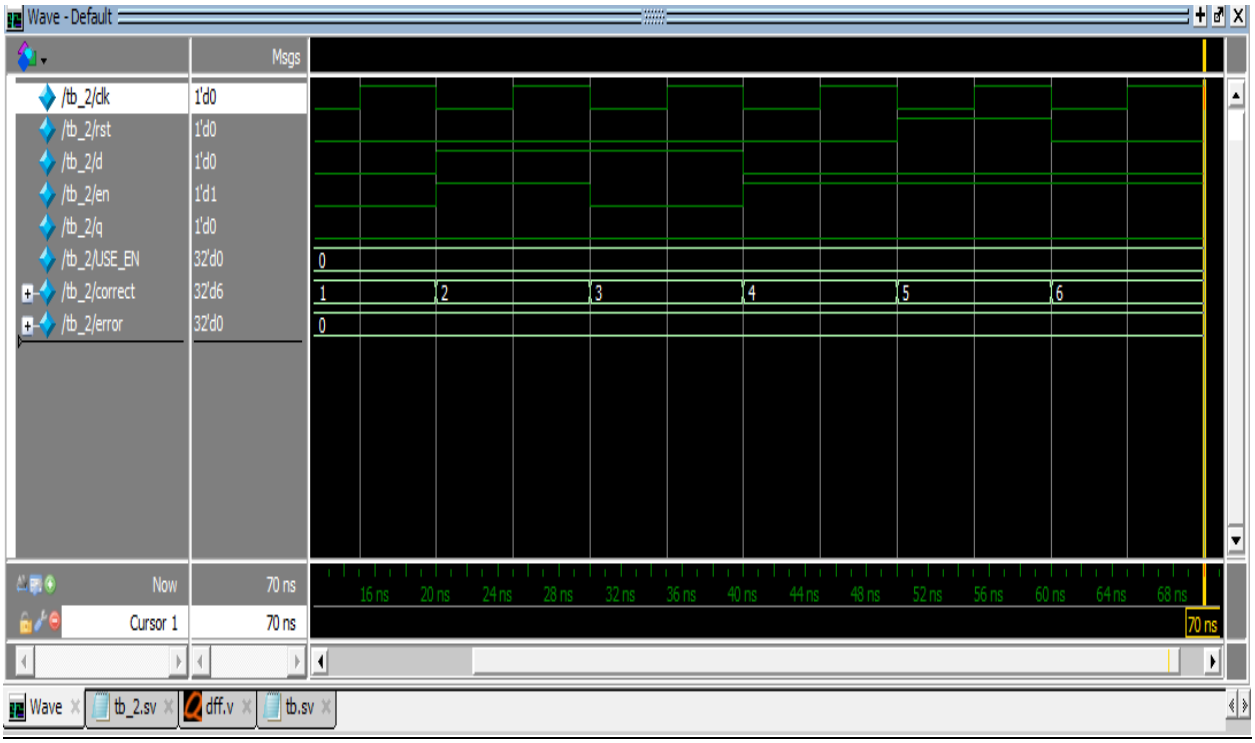



Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/USE_EN
/tb_2/correct
/tb_2/error
1'd0
1'd0
1'd1
1'd0
1'd0
32'd0
32'd3
32'd0
Now
70 ns
Cursor 1
30 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
Wave
tb_2.sv
dff.v
tb.sv


Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/USE_EN
/tb_2/correct
/tb_2/error
1'd0
1'd0
1'd0
1'd1
1'd0
32'd0
32'd4
32'd0
Now
70 ns
Cursor 1
40 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
Wave
tb_2.sv
dff.v
tb.sv


Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/USE_EN
/tb_2/correct
/tb_2/error
1'd0
1'd1
1'd0
1'd1
1'd0
32'd0
32'd5
32'd0
0
1
2
3
4
5
6
Now
70 ns
Cursor 1
50 ns
10 ns
20 ns
30 ns
40 ns
60 ns
Wave
tb_2.sv
dff.v
tb.sv


Wave - Default
Msgs
/tb_2/clk
/tb_2/rst
/tb_2/d
/tb_2/en
/tb_2/q
/tb_2/USE_EN
/tb_2/correct
/tb_2/error
1'd0
1'd0
1'd0
1'd1
1'd0
32'd0
32'd6
32'd0
0
1
2
3
4
5
6
Now
70 ns
Cursor 1
60 ns
16 ns
20 ns
24 ns
28 ns
32 ns
36 ns
40 ns
44 ns
48 ns
52 ns
56 ns
64 ns
68 ns
Wave
tb_2.sv
dff.v
tb.sv



COVERAGE REPORT

```
================================================================================
=== File: dff.v
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            4         4         0     100.0

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

Statement Coverage for file dff.v --

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

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

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

Branch Coverage for file dff.v --

------------------------------------IF Branch------------------------------------
    7                                           9     Count coming in to IF
    7               1                           4        if (rst)
    11              1                           2             if (en)
    13              1                           3             else
Branch totals: 3 hits of 3 branches = 100.0%


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

```
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"
--------------------------------------------------------------------------------------------
          3                                       rst           2           2      100.00
          3                                        en           2           2      100.00
          3                                         d           2           2      100.00
          3                                       clk           2           2      100.00
          4                                         q           1           1      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%
```