ALSU

Design

```
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input  [2:0] opcode;
input signed [2:0] A, B;
output reg [15:0] leds;
output reg signed[5:0] out;

reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg signed [1:0] cin_reg;
reg [2:0] opcode_reg;
reg signed [2:0] A_reg, B_reg; //change to signed
reg signed [5:0] out_next;
wire invalid_red_op, invalid_opcode, invalid;

//Invalid handling
assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
assign invalid = invalid_red_op | invalid_opcode;

//Registering input signals
always @(posedge clk or posedge rst) begin
  if(rst) begin
    cin_reg <= 0;
    red_op_B_reg <= 0;
    red_op_A_reg <= 0;
    bypass_B_reg <= 0;
    bypass_A_reg <= 0;
    direction_reg <= 0;
    serial_in_reg <= 0;
    opcode_reg <= 0;
```

```
    opcode_reg <= 0;
    A_reg <= 0;
    B_reg <= 0;

  end else begin
    cin_reg <= cin;
    red_op_B_reg <= red_op_B;
    red_op_A_reg <= red_op_A;
    bypass_B_reg <= bypass_B;
    bypass_A_reg <= bypass_A;
    direction_reg <= direction;
    serial_in_reg <= serial_in;
    opcode_reg <= opcode;
    A_reg <= A;
    B_reg <= B;

  end
end

//leds output blinking
always @(posedge clk or posedge rst) begin
  if(rst) begin
    leds <= 0;
  end else begin
    if (invalid)
      leds <= ~leds;
    else
      leds <= 0;
  end
end
```

```
63  //ALSU output processing
64  always @(posedge clk or posedge rst) begin
65
66    if(rst) begin
67      out <= 0;
68    end
69    else begin
70      if (bypass_A_reg && bypass_B_reg)
71        out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
72      else if (bypass_A_reg)
73        out <= A_reg;
74      else if (bypass_B_reg)
75        out <= B_reg;
76      else if (invalid) // cahnge the priority of invalid bits after bypass_reg
77          out <= 0;
78      else begin
79          case (opcode_reg)
80            3'h0: begin //change Opcode to OR not AND
81              if (red_op_A_reg && red_op_B_reg)
82                out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
83              else if (red_op_A_reg)
84                out <= |A_reg;
85              else if (red_op_B_reg)
86                out <= |B_reg;
87              else
88                out <= A_reg | B_reg;
89            end
90            3'h1: begin // change opcode to XOR not OR
91              if (red_op_A_reg && red_op_B_reg)
92                out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
93              else if (red_op_A_reg)
94                out <= ^A_reg;
```

```
94                out <= ^A_reg;
95              else if (red_op_B_reg)
96                out <= ^B_reg;
97              else
98                out <= A_reg ^ B_reg;
99            end
100           3'h2: begin //here we add condition to check full adder if ON or OFF
101             if(FULL_ADDER == "ON")
102               out <= A_reg + B_reg+cin_reg;
103             else if(FULL_ADDER == "OFF")
104               out <= A_reg + B_reg;
105
106           end
107           3'h3: out <= A_reg * B_reg;
108           3'h4: begin
109
110             if (direction_reg)
111               out <= {out_next[4:0], serial_in_reg};
112             else
113               out <= {serial_in_reg, out_next[5:1]};
114           end
115           3'h5: begin
116             if (direction_reg)
117               out <= {out_next[4:0], out_next[5]};
118             else
119               out <= {out_next[0], out_next[5:1]};
```

```
119               out <= {out_next[0], out_next[5:1]};
120           end
121         default: out<=out;
122         endcase
123     end
124   end
125   out_next<=out;
126 end
127
128 endmodule
```

Tb

```
import pack_alsu::*;

module ALSU_tb ();
parameter INPUT_PRIORITY = "B";
parameter FULL_ADDER = "ON";

bit  clk=0, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
Opcode_e opcode;
bit signed [2:0] A, B;
bit  [15:0] leds;
bit signed [5:0] out;
bit  [15:0] leds_exp;
bit signed [5:0] out_exp;

//register internal
reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg signed [1:0] cin_reg;
reg [2:0] opcode_reg;
reg signed [2:0] A_reg, B_reg; //change to signed



int error =0,correct=0;
bit invalid_t,x1,x2;
bit signed [5:0] last_out=0;

ALSU #(.INPUT_PRIORITY(INPUT_PRIORITY),.FULL_ADDER(FULL_ADDER)) tb (.*);

```

```
always #10 clk=!clk;
transaction tr=new();


initial begin
    assert_rst();
    check_rst();

    tr.constraint_mode(0);
    tr.x.constraint_mode(1);
    repeat(300) begin
        assert(tr.randomize());
        init(tr);
        if(rst)begin
            check_rst();
            reset_internal();
    end
        else begin
            check_result();
            sampling(tr);
        end

    end

    tr.constraint_mode(0);
    tr.y.constraint_mode(1);
    rst=0;bypass_A=0;bypass_B=0;red_op_A=0;red_op_B=0;
    tr.rst.rand_mode(0);tr.bypass_A.rand_mode(0);tr.bypass_B.rand_mode(0);tr.red_op_A.rand_mode(0);
    tr.red_op_B.rand_mode(0);
    init(tr);

```

```
    for(int i=0;i<1000;i++)begin
        assert(tr.randomize());
        cin=tr.cin;
        direction=direction_reg;
        serial_in=serial_in_reg;
        A=tr.A;
        B=tr.B;
        if('{OR,XOR,ADD,MULT,SHIFT,ROTATE} ==tr.arr)$display("@%0t the wanted sequence is %p",$time,tr.arr);
            foreach(tr.arr[j])begin
                tr.opcode=tr.arr[j];
                opcode=tr.arr[j];
                tr.out=out;
                tr.leds=leds;
                check_result();
                sampling(tr);
            end
    end
    $display("number of correct =%0d ,error=%0d",correct,error);
$stop;

end
```

```
task check_result();

    golden_model();
    @(negedge clk);
    if(out_exp!= out && leds_exp != leds)begin
        $display("@%t there is error out=%0b ,leds=%0b " ,$time ,tr.out,tr.leds);error++;
    end
    else
        correct++;


endtask

task golden_model();
    if(is_invalid())
        leds_exp= ~leds_exp;
    else
        leds_exp=0;
    if(bypass_A_reg && bypass_B_reg)begin
        if (INPUT_PRIORITY== "A")
            out_exp = A_reg;
        else if (INPUT_PRIORITY== "B")
            out_exp = B_reg;
        end
        // check on pypass operations
        else if(bypass_A) //check on bypass
                out_exp = A_reg;
        else if(bypass_B)
                out_exp = B_reg;
        //check on invalid_t condition output
        else if(is_invalid()) begin
```

```
                out_exp=0;
        end
        else begin
            //here we check on OP code
            case (opcode_reg)
            OR:begin// check on priority first
                if(red_op_A_reg && red_op_B_reg && INPUT_PRIORITY== "A")
                    out_exp = A_reg;
                else if(red_op_A_reg && red_op_B_reg&& INPUT_PRIORITY== "B")
                    out_exp = B_reg;
                else if(red_op_A_reg)
                    out_exp = (|A_reg);
                else if(red_op_B_reg)
                    out_exp = (|B_reg);
                else
                    out_exp = (A_reg|B_reg);
            end
            XOR:begin
                if(red_op_A_reg && red_op_B_reg && INPUT_PRIORITY== "A")
                    out_exp = A_reg;
                else if(red_op_A_reg && red_op_B_reg&& INPUT_PRIORITY== "B")
                    out_exp = B_reg;
                else if(red_op_A_reg)
                    out_exp = (^A_reg);
                else if(red_op_B_reg)
                    out_exp = (^A_reg);
                else
                    out_exp = (A_reg^B_reg);
            end
            ADD:begin
                if(FULL_ADDER == "ON")
```

```
                    out_exp= A_reg+B_reg+cin_reg;
                else if(FULL_ADDER == "OFF")
                    out_exp= A_reg+B_reg;
            end
            MULT:begin
                out_exp= A_reg*B_reg;
            end
            SHIFT:begin
                if(direction_reg)
                    out_exp = {out_exp[4:0],serial_in_reg};
                else if(!direction_reg)
                    out_exp = {serial_in_reg,out_exp[5:1]};
            end
            ROTATE:begin
                if(direction_reg)
                    out_exp = {out_exp[4:0],out_exp[5]};
                else if(!direction_reg)
                    out_exp = {out_exp[0],out_exp[5:1]};
            end
            endcase

        end
        update_internals();
    endtask

    task update_internals();
        red_op_A_reg=red_op_A; red_op_B_reg=red_op_B; bypass_A_reg=bypass_A;
        bypass_B_reg=bypass_B; direction_reg=direction; serial_in_reg=serial_in;
        cin_reg=cin;
        opcode_reg=opcode;
        A_reg=A;B_reg=B;
    endtask
```

```
178     task assert_rst();
179         rst=1;
180         @(negedge clk);
181         check_rst();
182         rst=0;
183     endtask
184 
185     task check_rst();
186         @(negedge clk);
187         if(out!=0 || leds!=0)begin
188             error++;$display("@$0t there is error on reset",$time);
189         end
190         else correct++;
191 
192         reset_internal();
193     endtask
194 
195     task reset_internal();
196         red_op_A_reg=0; red_op_B_reg=0; bypass_A_reg=0; bypass_B_reg=0; direction_reg=0; serial_in_reg=0;
197         cin_reg=0;
198         opcode_reg=0;
199         A_reg=0; B_reg=0; //change to signed
200     endtask
201 
```

```
202     function bit is_invalid();
203         if(opcode_reg==INVALID6 || opcode_reg==INVALID7)
204             return 1;
205         else if( (opcode_reg>3'b001) && (red_op_A_reg|red_op_B_reg))
206             return 1;
207         else
208             return 0;
209     endfunction
210 
211     function void init(transaction in);
212         opcode=tr.opcode;
213         A=tr.A;
214         B=tr.B;
215         rst=tr.rst;
216         cin=tr.cin;
217         red_op_A=tr.red_op_A;
218         red_op_B=tr.red_op_B;
219         bypass_A=tr.bypass_A;
220         bypass_B=tr.bypass_B;
221         direction=tr.direction;
222         serial_in=tr.serial_in ;
223         tr.out=out;
224         tr.leds=leds;
225     endfunction
```

```
227     function void sampling(transaction tr);
228         if(rst ||bypass_A ||bypass_B)begin
229             tr.cvr_gp.stop();
230         end
231         else begin
232             tr.cvr_gp.start();
233             tr.cvr_gp.sample();
234         end
235     endfunction
236 endmodule
```

Package

```
package pack_alsu;
typedef enum { OR=0,XOR,ADD,MULT,SHIFT,ROTATE,INVALID6,INVALID7 } Opcode_e;
typedef enum {MAXPOS=3,MAXNEG=-4,ZERO=0}corner_state_e;

class transaction;
    rand bit  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    rand Opcode_e opcode;
    rand bit signed [2:0] A, B;
    bit [2:0] ones_number={3'b001,3'b010,3'b100};
    rand bit [2:0] found,notfound;
    rand corner_state_e a_state;
    rand bit [2:0] rem_numbers;
    bit signed [5:0] out;
    bit [15:0] leds;

    rand Opcode_e arr[6];

    constraint x {

      rem_numbers!= MAXPOS||MAXNEG||ZERO;

       rst dist {1:=5 , 0:=95};

       found inside {ones_number};
       !(notfound inside {ones_number});

       if (opcode ==ADD || opcode== MULT){
          A dist {a_state:=80,rem_numbers:=20};
          B dist {a_state:=80,rem_numbers:=20};
       }
       if (opcode ==OR || opcode== XOR ){
          if(red_op_A){
```

```
          if(red_op_A){
              A dist {found:=80,notfound:=20};
              B==3'b000;
          }
          else if (red_op_B){
              B dist {found:=80,notfound:=20};
              A==3'b000;
          }
       }


       opcode dist {[OR:ROTATE]:=80,[INVALID6:INVALID7]};

       bypass_A dist {0:=90,1:=10};
       bypass_B dist {0:=90,1:=10};
    }
    constraint y{
        unique{arr};
        foreach(arr[i])
            arr[i] inside {[OR:ROTATE]};
    }


    covergroup cvr_gp;
        special:coverpoint opcode{
            option.weight=0;
            bins operataions[]={[OR:ROTATE]};
        }
        CB1:coverpoint A{
            bins A_data_0={0};
```

```
        CB1:coverpoint A{
            bins A_data_0={0};
            bins A_data_max={MAXPOS};
            bins A_data_min={MAXNEG};
            bins A_data_walkingones[] ={3'b001,3'b010,3'b100} iff (red_op_A);
            bins A_data_default=default;
        }
        CB2:coverpoint B{
            bins B_data_0={0};
            bins B_data_max={MAXPOS};
            bins B_data_min={MAXNEG};
            bins B_data_walkingones[] ={3'b001,3'b010,3'b100} iff (red_op_B);
            bins B_data_default=default;
        }
        A_M:coverpoint opcode{
            bins Bins_arith[] ={ADD,MULT};
        }
        CB3:coverpoint opcode{
            bins Bins_shift[]={SHIFT,ROTATE};
            bins Bins_arith[] ={ADD,MULT};
            bins Bins_bitwise[] ={OR,XOR};
            illegal_bins Bins_invalid ={INVALID6,INVALID7};
            bins Bins_trans=(OR=>XOR=>ADD=>MULT=>SHIFT=>ROTATE);
        }
        //1
        corner_case:cross A_M,CB2,CB1{
            ignore_bins walkingA=binsof(CB1.A_data_walkingones) ;
            ignore_bins walkingB=binsof(CB2.B_data_walkingones) ;
        }
        //2
```

```
        Addation:cross cin,special{
            option.cross_auto_bin_max=0;
            bins Add_cin0=binsof(special) intersect {ADD} && binsof(cin) intersect {0};
            bins Add_cin1=binsof(special) intersect {ADD}&& binsof(cin) intersect {1};
        }
        //3
        shift:cross special,serial_in{
            option.cross_auto_bin_max=0;
            bins shift_Si0=binsof(special) intersect {SHIFT} && binsof(serial_in) intersect {0};
            bins shift_Si1=binsof(special) intersect {SHIFT} && binsof(serial_in) intersect {1};
        }
        //4
        shift_rotate:cross CB3,direction{
            option.cross_auto_bin_max=0;
            bins shu_rot_d0=binsof(CB3.Bins_shift) && binsof(direction) intersect {0} ;
            bins shu_rot_d1=binsof(CB3.Bins_shift) && binsof(direction) intersect {1} ;
        }
        //5
        walkingones:cross CB3,CB2,CB1{
            option.cross_auto_bin_max=0;
            bins arithA=binsof(CB3.Bins_bitwise) && binsof(CB2.B_data_0) &&binsof(CB1.A_data_walkingones) ;
            bins arithB=binsof(CB3.Bins_bitwise) && binsof(CB1.A_data_0) &&binsof(CB2.B_data_walkingones) ;
        }
        invalidation:cross red_op_A,red_op_B,special{
        option.cross_auto_bin_max=0;
            bins ROpA_notXoR=binsof(special) intersect{[ADD:ROTATE]} && binsof(red_op_A)  intersect{1};
            bins ROpB_notXoR=binsof(special) intersect{[ADD:ROTATE]} && binsof(red_op_B)  intersect{1};
        }

    endgroup
```

```
    function new();
        cvr_gp=new();
    endfunction

endclass
endpackage
```

Do file

```
vlib work
vlog ALSU.v ALSU_tb.sv pack_alsu.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save ALSU_tb.ucdb -onexit
run -all
coverage exclude -src ALSU.v -line 121 -code b
coverage exclude -src ALSU.v -line 121 -code s
coverage exclude -du ALSU -togglenode {cin_reg[1]}
quit -sim

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

Verification plan

1	Label	Description	Stimulus Generation	Functionality Check	Functionalit check
2	ALSU_1	we assert reset on start so OUT should be low and led should be low	Directed at the start of the simulation then it randomized under cosntraint to be of high 95 % from time	A checker in the testbench to make sure the output is correct by test function	-
3	ALSU_2	when the byPass_A is asserted OUT take value or reg A igonre Opcode and byPass_B is asserted OUT take value or reg B if Both high so out take input with HIGH priority	Randomized in class under constraint that make bypassA and bypass B is Low 90 % of time	A checker in the testbench to make sure the output is functionally correct by test function	-
4	ALSU_3	when Opcode= 6 or 7 its invalid and if red_op1 or red_op2 is high and OPcode is not or ,xor so its invalid case then Output is low	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	we cover this in coverage group with cross coverage when opcode is not or ,xor and reduction is active
5	ALSU_4	when opcode =OR so output is =|A when red_opA is high same output =|B if this red_op_b high if both high so output check priority and if both low output =A|B	Randomized in class under consraints that Opcode is valid most of simulation and also if OR operation so redA and RedB is high together most of simulation and also at least input if RedA is high only A should have at least 1 bit =1	A checker in the testbench to make sure the output is functionally correct by test function	we cover all Opcodes in from OR to ROTATE and put each one in Bin and and we cover transation from OR to ROTATE and we also we cover all corner casses of A ND B like MAXPOS and MAXNEG and ZERO with cross coveerage and when RED_OP is high for values 1 2 4 and reamaining values we
6	ALSU_5	when opcode =XOR so output is =^A when red_opA is high same output =^B if this red_op_b high if both high so output check priority and if both low output =A^B	Randomized in class under consraints if xOR operation so redA and RedB is high together most of simulation and also at least input if RedA is high only A should have at least 1 bit =1 and same for B	A checker in the testbench to make sure the output is functionally correct by test function	
7	ALSU_6	when opcode =ADD and full adder on so output =A+B+cin if full adder off out=A+B	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	cover with cross coverage when op code is add and cin should be 1 or 0
8	ALSU_7	when opcode =mult so out=A*B	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	cover with cross coverage when op code is mult and inputs take corner cases
9	ALSU_8	when opcode =SHIFT and depend on Direction output will be left or right and serial in	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	
10	ALSU_9	when opcode =ROTATE and depend on Direction output will be rotate left or right	Randomized in class	A checker in the testbench to make sure the output is functionally correct by test function	

Code Coverage report

```
=== File: ALSU.v
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           49        49         0     100.0

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

Statement Coverage for file ALSU.v --

    1                                           module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
    2                                           parameter INPUT_PRIORITY = "A";
    3                                           parameter FULL_ADDER = "ON";
    4                                           input  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    5                                           input  [2:0] opcode;
    6                                           input signed [2:0] A, B;
    7                                           output reg [15:0] leds;
    8                                           output reg signed[5:0] out;
    9
    10                                          reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
    11                                          reg signed [1:0] cin_reg;
    12                                          reg [2:0] opcode_reg;
    13                                          reg signed [2:0] A_reg, B_reg; //change to signed
    14                                          reg signed [5:0] out_next;
    15                                          wire invalid_red_op, invalid_opcode, invalid;
    16
    17                                          //Invalid handling
    18              1                   6106    assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
    19              1                   6073    assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    20              1                   3294    assign invalid = invalid_red_op | invalid_opcode;
    21
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        31        31         0     100.0

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

Branch Coverage for file ALSU.v --

----------------------------------IF Branch-----------------------------------
    24                                      6325    Count coming in to IF
    24              1                         46      if(rst) begin
    36              1                       6279      end else begin
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------IF Branch-----------------------------------
    53                                      6326    Count coming in to IF
    53              1                         47      if(rst) begin
    55              1                       6279      end else begin
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------IF Branch-----------------------------------
    56                                      6279    Count coming in to IF
    56              1                       4138          if (invalid)
    58              1                       2141          else
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------IF Branch-----------------------------------
    66                                      6325    Count coming in to IF
    66              1                         47      if(rst) begin
    69              1                       6278      else begin
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------IF Branch-----------------------------------
    70                                      6278    Count coming in to IF
    70              1                          2        if (bypass_A_reg && bypass_B_reg)
    72              1                         19        else if (bypass_A_reg)
```

```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ----------
    Toggle Bins                    130       130         0     100.0

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

Toggle Coverage for File ALSU.v --
```

Line	Node	1H->0L	0L->1H	"Coverage"
4	serial_in	1	1	100.00
4	rst	1	1	100.00
4	red_op_B	1	1	100.00
4	red_op_A	1	1	100.00
4	direction	1	1	100.00
4	clk	1	1	100.00
4	cin	1	1	100.00
4	bypass_B	1	1	100.00
4	bypass_A	1	1	100.00
5	opcode[2]	1	1	100.00
5	opcode[1]	1	1	100.00
5	opcode[0]	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
6	A[2]	1	1	100.00
6	A[1]	1	1	100.00
6	A[0]	1	1	100.00
7	leds[9]	1	1	100.00

Function Coverage report

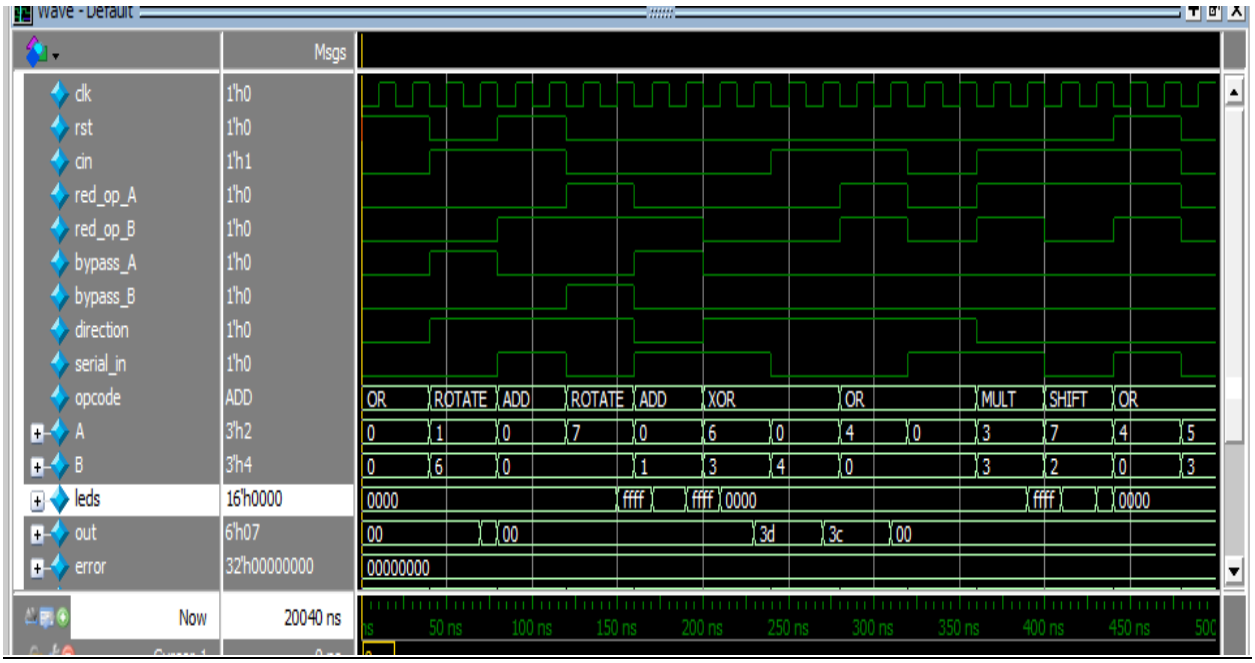
```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal    Status

----------------------------------------------------------------------------------------
 TYPE /pack_alsu/transaction/cvr_gp                 100.0%        100    Covered
    covered/total bins:                                 63         63
    missing/total bins:                                  0         63
    % Hit:                                          100.0%        100
    Coverpoint cvr_gp::special                        0.0%        100    ZERO
        covered/total bins:                              6          6
        missing/total bins:                              0          6
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::CB1                          100.0%        100    Covered
        covered/total bins:                              5          5
        missing/total bins:                              0          5
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::CB2                          100.0%        100    Covered
        covered/total bins:                              5          5
        missing/total bins:                              0          5
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::A_M                          100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::CB3                          100.0%        100    Covered
        covered/total bins:                              7          7
        missing/total bins:                              0          7
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::red_op_A                     100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::red_op_B                     100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::direction                    100.0%        100    Covered
```

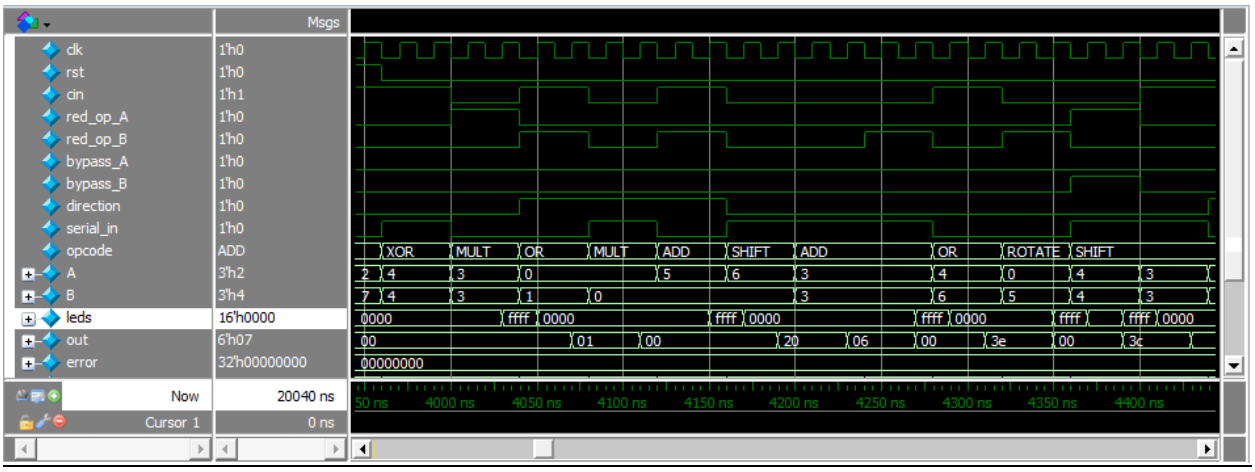
```
    Coverpoint cvr_gp::red_op_B                    100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::direction                   100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::serial_in                   100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Coverpoint cvr_gp::cin                         100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Cross cvr_gp::corner_case                      100.0%        100    Covered
        covered/total bins:                            18         18
        missing/total bins:                             0         18
        % Hit:                                      100.0%        100
    Cross cvr_gp::Addation                         100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Cross cvr_gp::shift                            100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Cross cvr_gp::shift_rotate                     100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Cross cvr_gp::walkingones                      100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
    Cross cvr_gp::invalidation                     100.0%        100    Covered
        covered/total bins:                             2          2
```

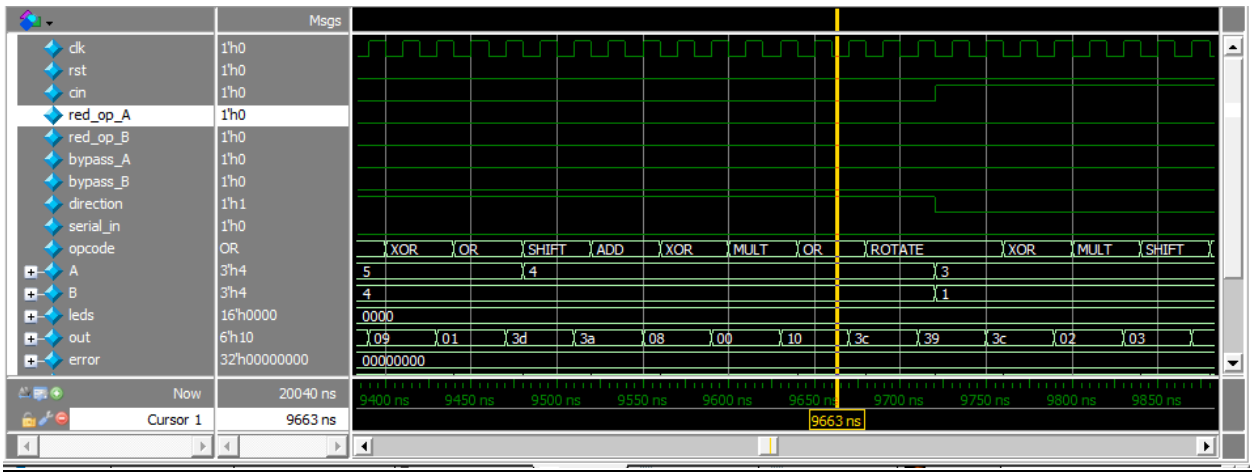
```
    Cross cvr_gp::invalidation                     100.0%        100    Covered
        covered/total bins:                             2          2
        missing/total bins:                             0          2
        % Hit:                                      100.0%        100
CLASS transaction
Covergroup instance \/pack_alsu::transaction::cvr_gp
                                                    100.0%        100    Covered
    covered/total bins:                                63         63
    missing/total bins:                                 0         63
    % Hit:                                          100.0%        100
    Coverpoint special [1]                         100.0%        100    Covered
        covered/total bins:                             6          6
        missing/total bins:                             0          6
        % Hit:                                      100.0%        100
        bin operataions[OR]                          1038          1    Covered
        bin operataions[XOR]                         1036          1    Covered
        bin operataions[ADD]                         1027          1    Covered
        bin operataions[MULT]                        1039          1    Covered
        bin operataions[SHIFT]                       1047          1    Covered
        bin operataions[ROTATE]                      1044          1    Covered
    Coverpoint CB1                                 100.0%        100    Covered
        covered/total bins:                             5          5
        missing/total bins:                             0          5
        % Hit:                                      100.0%        100
        bin A_data_0                                  797          1    Covered
        bin A_data_max                                825          1    Covered
        bin A_data_min                                709          1    Covered
        bin A_data_walkingones[1]                     830          1    Covered
        bin A_data_walkingones[2]                     759          1    Covered
        default bin A_data_default                   2282               Occurred
    Coverpoint CB2                                 100.0%        100    Covered
        covered/total bins:                             5          5
        missing/total bins:                             0          5
        % Hit:                                      100.0%        100
        bin B_data_0                                  837          1    Covered
        bin B_data_max                                863          1    Covered
        bin B_data_min                                831          1    Covered
        bin B_data_walkingones[1]                       9          1    Covered
```

```
    Cross shift_rotate                              100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                       100.0%        100
        bin shu_rot_d0                                1092          1    Covered
        bin shu_rot_d1                                 999          1    Covered
    Cross walkingones                               100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                       100.0%        100
        bin arithA                                     58          1    Covered
        bin arithB                                      2          1    Covered
    Cross invalidation                              100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                       100.0%        100
        bin ROpA_notXoR                              4082          1    Covered
        bin ROpB_notXoR                                87          1    Covered

 [1] - Does not contribute coverage as weight is 0

TOTAL COVERGROUP COVERAGE: 100.0%  COVERGROUP TYPES: 1

ASSERTION RESULTS:
---------------------------------------------------------
Name                File(Line)           Failure Pass
                                         Count   Count
---------------------------------------------------------
```

Simulation



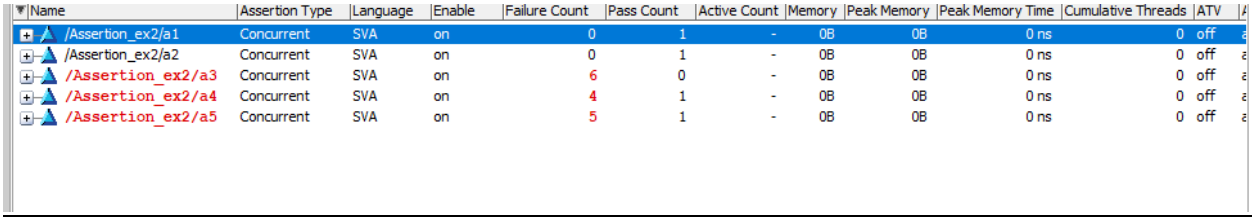

Msgs
clk
rst
cin
red_op_A
red_op_B
bypass_A
bypass_B
direction
serial_in
opcode
A
B
leds
out
error
1'h0
1'h0
1'h1
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
ADD
3'h2
3'h4
16'h0000
6'h07
32'h00000000
XOR MULT OR MULT ADD SHIFT ADD OR ROTATE SHIFT
2 4 3 0 5 6 3 4 0 4 3
7 4 3 1 0 3 6 5 4 3
0000 ffff 0000 ffff 0000 ffff 0000 ffff ffff 0000
00 01 00 20 06 00 3e 00 3c
00000000
Now
20040 ns
Cursor 1
0 ns
50 ns 4000 ns 4050 ns 4100 ns 4150 ns 4200 ns 4250 ns 4300 ns 4350 ns 4400 ns


Msgs
clk
rst
cin
red_op_A
red_op_B
bypass_A
bypass_B
direction
serial_in
opcode
A
B
leds
out
error
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h0
1'h1
1'h0
OR
3'h4
3'h4
16'h0000
6'h10
32'h00000000
XOR OR SHIFT ADD XOR MULT OR ROTATE XOR MULT SHIFT
5 4 3
4 1
0000
09 01 3d 3a 08 00 10 3c 39 3c 02 03
00000000
Now
20040 ns
Cursor 1
9663 ns
9400 ns 9450 ns 9500 ns 9550 ns 9600 ns 9650 ns 9700 ns 9750 ns 9800 ns 9850 ns
9663 ns

Ex2

```
module Assertion_ex2 ();
logic a,b,c,valid;
logic [3:0]d;
bit clk=0;
logic[7:0]Y;

always #100 clk=!clk;

property x1;
    @(posedge clk)(a |-> ##2 b);
endproperty

property x2;
@(posedge clk)(a && b |-> ##[1:3] c);
endproperty

sequence s11b;
##2 !b;
endsequence

property x3;
@(posedge clk) s11b;
endproperty

property x4;
@(posedge clk) $onehot(Y);
endproperty

property x5;
@(posedge clk) ($countones(d)==0 |-> ##1 !valid);

endproperty
```

```
endproperty
a1:assert property(x1) ;
a1_c:cover property(x1);

a2:assert property(x2) ;
a2_c:cover property(x2);

a3:assert property(x3) ;
a3_c:cover property(x3);

a4:assert property(x4) ;
a4_c:cover property(x4);

a5:assert property(x5) ;
a5_c:cover property(x5);

initial begin
    a=1;b=0;c=0;Y='d1;
    repeat(2)@(negedge clk);
    a=1;b=1;
    repeat(2)@(negedge clk);
    c=1;Y=0;d=4'b0000;valid=1;
    repeat(2)@(negedge clk);
    valid=0;
    repeat(2)@(negedge clk);
    $stop;
end
endmodule
```



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV
/Assertion_ex2/a1	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off
/Assertion_ex2/a2	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off
/Assertion_ex2/a3	Concurrent	SVA	on	6	0	-	0B	0B	0 ns	0	off
/Assertion_ex2/a4	Concurrent	SVA	on	4	1	-	0B	0B	0 ns	0	off
/Assertion_ex2/a5	Concurrent	SVA	on	5	1	-	0B	0B	0 ns	0	off

Priority encoder

design

```
module priority_enc (if_pe.DUT if_t);

always @(posedge if_t.clk) begin
  if (if_t.rst) begin
     if_t.Y <= 2'b0;
     if_t.valid<=1'b0;
     end
  else
  begin
    casex (if_t.D)
        4'b1000: if_t.Y <= 0;//2'b00  1
        4'bX100: if_t.Y <= 1;//2'b01  1
        4'bXX10: if_t.Y <= 2;//2'b10  1
        4'bXXX1: if_t.Y <= 3;//2'b11  1
    endcase
    if_t.valid <= (~|if_t.D)? 1'b0: 1'b1;
  end
end
endmodule
```

Tb

```
module priority_enc_tb(if_pe.TB if_t);


initial begin

if_t.D=1;
if_t.rst=1;
@(negedge if_t.clk);#2;
if_t.rst=0;

for(int i=0;i<16;i++)begin
    if_t.D=i;
    @(negedge if_t.clk);
end
if_t.D=0;
if_t.rst=1;

repeat(2) @(negedge if_t.clk);
$stop;
end


endmodule
```

VERIFICATION PLAN

Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
PRIORTY_	When the reset is asserted, the output valid value should be low and Y dont care	Directed at the start of the simulation	check with concurent assertion on output	A checker in the testbench to make sure the output is correct
PRIORTY_	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	check with concurent assertion on output	A checker in the testbench to make sure the output is correct
PRIORTY_	we assert the reset to 0 to see the stimulus on Output Y and valid	Directed at the simulation	check with concurent assertion on output	A checker in the testbench to make sure the output is correct
PRIORTY_	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	check with concurent assertion on output	A checker in the testbench to make sure the output is correct

interface

```
interface if_pe(clk);
input bit clk;
logic  rst;
logic  [3:0] D;
logic  [1:0] Y;
logic  valid;

modport DUT (input clk,rst,D,output Y,valid);
modport TB (output clk,rst,D,input Y,valid);
modport Assertions (input clk,rst,D,output Y,valid);
endinterface
```

assertions

```
module asser(if_pe.Assertions if_t);

property rst;
@(negedge if_t.clk)  if_t.rst==1'b1 |-> ( if_t.Y==2'b00 && if_t.valid ==1'b0);
endproperty

property valid_bit;
@(negedge if_t.clk) disable iff(if_t.rst) (($countones(if_t.D) > 0) |-> if_t.valid);
endproperty

property Bit_0;
@(negedge if_t.clk) disable iff(if_t.rst)
    if_t.D[0] |-> if_t.Y==2'b11;
endproperty

property Bit_1;
@(negedge if_t.clk) disable iff(if_t.rst)

    (if_t.D[1] && ! if_t.D[0]) |-> if_t.Y==2'b10 ;

endproperty
property Bit_2;
@(negedge if_t.clk) disable iff(if_t.rst)
    (if_t.D[2] && !if_t.D[1] &&  ! if_t.D[0]  ) |-> if_t.Y==2'b01 ;
endproperty
property Bit_3;
@(negedge if_t.clk) disable iff(if_t.rst)
    ( if_t.D[3] && !if_t.D[2] && !if_t.D[1] &&  ! if_t.D[0]  ) |-> if_t.Y==2'b00 ;
endproperty

rst_check:assert property (rst)else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,(if_t.Y |if_t.valid) ,if_t.D,if_t.Y);
valid_bit_check:assert property (valid_bit)else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,($countones(if_t.D) > 0),if_t.D,if_t.Y);

output_c0:assert property (Bit_0) else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,$countones(if_t.D),if_t.D,if_t.Y);
output_c1:assert property (Bit_1) else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,$countones(if_t.D),if_t.D,if_t.Y);
output_c2:assert property (Bit_2) else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,$countones(if_t.D),if_t.D,if_t.Y);
output_c3:assert property (Bit_3) else $error("reset=%0d,valid=%0d,D=%0d,y=%0d",if_t.rst,$countones(if_t.D),if_t.D,if_t.Y);

rst_check_cover:cover property (rst);
valid_bit_check_cover:cover property (valid_bit);

output_check_cover0:cover property (Bit_0);
output_check_cover1:cover property (Bit_1);
output_check_cover2:cover property (Bit_2);
output_check_cover3:cover property (Bit_3);
endmodule
```

Top

```
module top();
bit clk=0;

always #5 clk=!clk;

if_pe if_t(clk);
priority_enc dut(if_t);
priority_enc_tb tb(if_t);
asser ASV(if_t);


endmodule
```

Do file

```
vlib work
vlog *v  +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave *
coverage save priroty_encoder_tb.ucdb -onexit
run -all
quit -sim

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

Code coverage

```
=== File: priority_enc.sv
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            8         8         0     100.0

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

Statement Coverage for file priority_enc.sv --

    1                                           module priority_enc (if_pe.DUT if_t);
    2
    3               1                    19     always @(posedge if_t.clk) begin
    4                                             if (if_t.rst) begin
    5               1                    3           if_t.Y <= 2'b0;
    6               1                    3         if_t.valid<=1'b0;
    7                                              end
    8                                            else
    9                                            begin
    10                                            casex (if_t.D)
    11              1                    1                4'b1000: if_t.Y <= 0;//2'b00  1
    12              1                    2                4'bX100: if_t.Y <= 1;//2'b01  1
    13              1                    4                4'bXX10: if_t.Y <= 2;//2'b10  1
    14              1                    8                4'bXXX1: if_t.Y <= 3;//2'b11  1
    15                                            endcase
    16              1                    16       if_t.valid <= (~|if_t.D)? 1'b0: 1'b1;
    17                                           end
    18                                          end
    19                                          endmodule
```

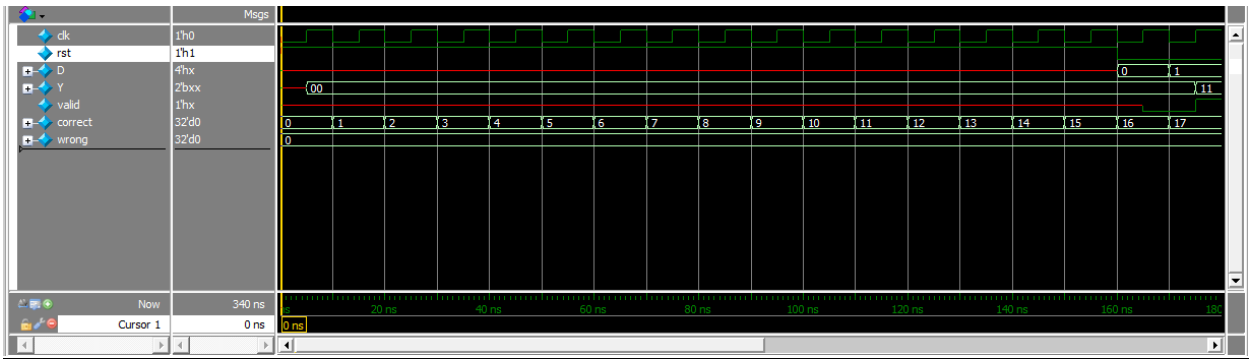
```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                         7         7         0     100.0

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

Branch Coverage for file priority_enc.sv --

------------------------------------IF Branch------------------------------------
    4                                       19     Count coming in to IF
    4               1                        3       if (if_t.rst) begin
    8               1                       16       else
Branch totals: 2 hits of 2 branches = 100.0%

----------------------------------CASE Branch----------------------------------
    10                                      16     Count coming in to CASE
    11              1                        1               4'b1000: if_t.Y <= 0;//2'b00  1
    12              1                        2               4'bX100: if_t.Y <= 1;//2'b01  1
    13              1                        4               4'bXX10: if_t.Y <= 2;//2'b10  1
    14              1                        8               4'bXXX1: if_t.Y <= 3;//2'b11  1
                                             1     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
```

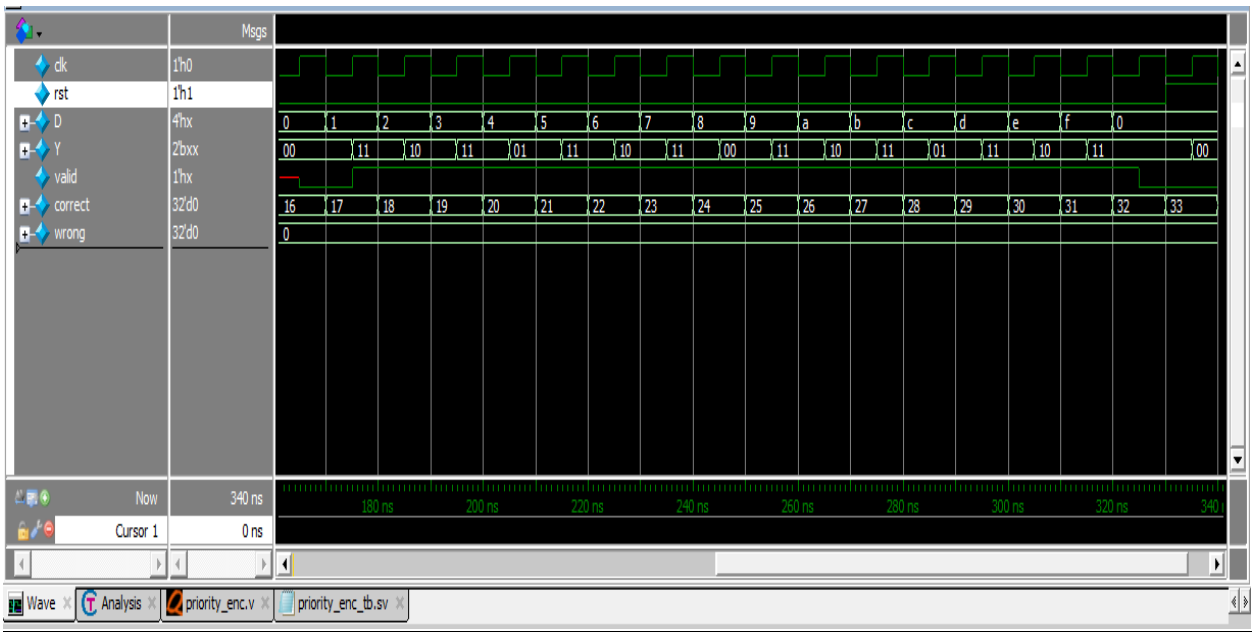
```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                      0         0         0     100.0

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

Toggle Coverage for File priority_enc.sv --

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

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

Toggle Coverage      =      100.0% (0 of 0 bins)

================================================================================
=== File: priority_enc_tb.sv
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           14        14         0     100.0
```

---

assertions

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Count Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/top/ASV/rst_check_cover                asser  Verilog  SVA  asser.sv(39)        2 Covered
/top/ASV/valid_bit_check_cover          asser  Verilog  SVA  asser.sv(40)       14 Covered
/top/ASV/output_check_cover0            asser  Verilog  SVA  asser.sv(42)        7 Covered
/top/ASV/output_check_cover1            asser  Verilog  SVA  asser.sv(43)        4 Covered
/top/ASV/output_check_cover2            asser  Verilog  SVA  asser.sv(44)        2 Covered
/top/ASV/output_check_cover3            asser  Verilog  SVA  asser.sv(45)        1 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 6

ASSERTION RESULTS:
---------------------------------------------------------
Name                 File(Line)           Failure Pass
                                          Count   Count
---------------------------------------------------------
/top/ASV/rst_check   asser.sv(31)               0     1
/top/ASV/valid_bit_check
                     asser.sv(32)               0     1
/top/ASV/output_c0   asser.sv(34)               0     1
/top/ASV/output_c1   asser.sv(35)               0     1
/top/ASV/output_c2   asser.sv(36)               0     1
/top/ASV/output_c3   asser.sv(37)               0     1

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





ALU

Design

```
import pack_file::*;
module ALU_4_bit(if_d.DUT if_t);
   reg signed [4:0]Alu_out;
   // Do the operation
   always @* begin
      case (if_t.Opcode)
         if_t.Add:     Alu_out = if_t.A + if_t.B;
         if_t.Sub:     Alu_out = if_t.A - if_t.B;
         if_t.Not_A:   Alu_out = ~if_t.A;
         if_t.ReductionOR_B:  Alu_out = |if_t.B;
       default:  Alu_out = 5'b0;
      endcase
   end // always @ *

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

endmodule
```

Tb

```
import pack_file::*;
module ALU_tb (if_d.TB if_t);

transaction tr=new();
initial begin
    if_t.reset=1;
    @(negedge if_t.clk);
    if_t.A=0;if_t.B=0;if_t.Opcode=0;
    if_t.reset=0;
    repeat(50) begin
        assert(tr.randomize());
        if_t.reset=tr.reset;
        if_t.Opcode=tr.Opcode;
        if_t.A=tr.A;
        if_t.B=tr.B;
    @(negedge if_t.clk);


    end
$stop;
end



endmodule
```

VERIFICATION PLAN

	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	check with concurent assertion on reset
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
9	ADD_4	negative number and B asserted to Max postive number OP=ADD the output C should be equal -1	simulation at time 70	on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition

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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on addition
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	check with concurent assertion for output on subtraction
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	check with concurent assertion for output on NOT
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	check with concurent assertion for output on NOT
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	check with concurent assertion for output on NOT
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	check with concurent assertion for output on NOT
25	Reduction_	When B is asserted to Zero OP=REDUCTION_OR the output C should be equal ZERO	Directed at the simulation at time 230	check with concurent assertion for output on REDUCTION
26	Reduction_	When B is asserted to MAXPOS OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 240	check with concurent assertion for output on REDUCTION
27	Reduction_	When B is asserted to MAXNEG OP=REDUCTION_OR the output C should be equal ONE	Directed at the simulation at time 250	check with concurent assertion for output on REDUCTION
28	Reduction_	When B is asserted to -1 OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 260	check with concurent assertion for output on REDUCTION
29	Reduction_	When B is asserted to 1 OP=REDUCTION_OR the output C should be equal 1	Directed at the simulation at time 270	check with concurent assertion for output on REDUCTION
30	RESET	When the reset is asserted, the output adder value should be low	Directed at the simulation at time 280	check with concurent assertion on reset
31	STOP	we end simulation here by $stop	Directed at the simulation at time 290	

interface

```
interface if_d(clk);
input bit clk;
logic reset;
logic [1:0]Opcode;
logic signed [3:0] A;
logic signed [3:0] B;
logic signed [4:0] C;


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


modport DUT (input clk,reset,Opcode,A,B,output C);
modport TB (output clk,reset,Opcode,A,B,input C);
modport Assertionss (input clk,reset,Opcode,A,B,output C);
endinterface

```

Top

```
module top();
bit clk=0;

always #50 clk=!clk;

if_d if_t(clk);
ALU_4_bit dut(if_t);
ALU_tb tb(if_t);
Asser ASV(if_t);


endmodule
```

assertions

```
1
2   module Asser(if_d.Assertionss if_t);
3
4   property rst_check;
5   @(posedge if_t.clk or posedge if_t.reset)  $rose(if_t.reset) |->if_t.C==0 ;
6   endproperty
7
8
9   property addition_check;
10    @(negedge if_t.clk) disable iff (if_t.reset)
11      (if_t.Opcode ==0) |-> (if_t.C == (if_t.A + if_t.B));
12  endproperty
13
14
15  property subtraction_check;
16   @(negedge if_t.clk) disable iff (if_t.reset)
17      (if_t.Opcode==if_t.Sub) |-> (if_t.C == (if_t.A-if_t.B));
18  endproperty
19
20  property NotA_check;
21  @(negedge if_t.clk) disable iff (if_t.reset)  (if_t.Opcode==if_t.Not_A |->( if_t.C== ~if_t.A));
22  endproperty
23
24  property ReductionOR_B_check;
25  @(negedge if_t.clk) disable iff (if_t.reset)  (if_t.Opcode==if_t.ReductionOR_B |-> (if_t.C== |if_t.B));
26  endproperty
27  |
28
29  rst:assert property(rst_check)$display("done"); else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
30                                                                                                          ,if_t.C);
31  rst_co:cover property(rst_check);
32
```

```
31  rst_co:cover property(rst_check);
32
33  addition:assert property(addition_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
34                                                                                                  ,if_t.C);
35  addition_co:cover property(addition_check);
36
37  subtraction:assert property(subtraction_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
38                                                                                                  ,if_t.C);
39  subtraction_co:cover property(subtraction_check);
40
41  not_ch:assert property(NotA_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
42                                                                                              ,if_t.C);
43  not_ch_co:cover property(NotA_check);
44
45  oring:assert property(ReductionOR_B_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
46                                                                                                  ,if_t.C);
47  oring_co:cover property(ReductionOR_B_check);
48
49  endmodule
```

Do file

```
vlib work
vlog *v +cover
vsim -voptargs=+acc work.top -cover
add wave *
coverage save ALU_tb.ucdb -onexit
run -all
coverage exclude -src {E:/study/kareem wassem/ASSIGMENTS/4/ALU/ALU.sv} -line 11

quit -sim

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

Code coverage

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

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

Statement Coverage for file ALU.sv --

    1                                           import pack_file::*;
    2                                           module ALU_4_bit(if_d.DUT if_t);
    3                                              reg signed [4:0]Alu_out;
    4                                              // Do the operation
    5               1                       1      always @* begin
    6                                                 case (if_t.Opcode)
    7               1                      15              if_t.Add:     Alu_out = if_t.A + if_t.B;
    8               1                      19              if_t.Sub:     Alu_out = if_t.A - if_t.B;
    9               1                       5              if_t.Not_A:   Alu_out = ~if_t.A;
    10              1                      10              if_t.ReductionOR_B:  Alu_out = |if_t.B;
    11              1                       E            default:  Alu_out = 5'b0;
    12                                                 endcase
    13                                             end // always @ *
    14
    15                                             // Register output C
    16              1                      54      always @(posedge if_t.clk or posedge if_t.reset) begin
    17                                                if (if_t.reset)
    18              1                       6           if_t.C <= 5'b0;
    19                                                else
    20              1                      48           if_t.C<= Alu_out;
    21                                             end
    22


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

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

Branch Coverage for file ALU.sv --

------------------------------------CASE Branch------------------------------------
    6                                       49     Count coming in to CASE
    7               1                       15              if_t.Add:     Alu_out = if_t.A + if_t.B;
    8               1                       19              if_t.Sub:     Alu_out = if_t.A - if_t.B;
    9               1                        5              if_t.Not_A:   Alu_out = ~if_t.A;
    10              1                       10              if_t.ReductionOR_B:  Alu_out = |if_t.B;
    11              1                        E            default:  Alu_out = 5'b0;
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    17                                      54     Count coming in to IF
    17              1                        6           if (if_t.reset)
    19              1                       48           else
Branch totals: 2 hits of 2 branches = 100.0%


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

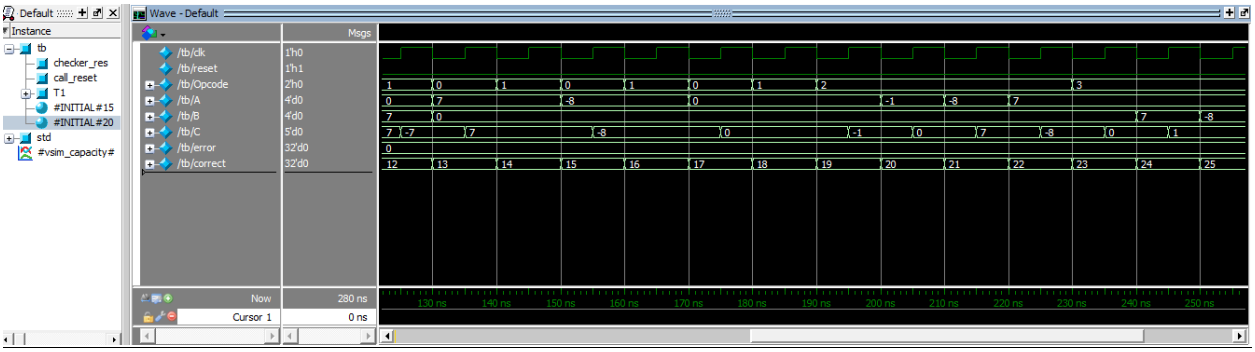
```
Toggle Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Toggle Bins                     10        10         0     100.0

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

Toggle Coverage for File ALU.sv --

       Line                                  Node      1H->0L      0L->1H  "Coverage"
--------------------------------------------------------------------------------------
          3                             Alu_out[4]           1           1      100.00
          3                             Alu_out[3]           1           1      100.00
          3                             Alu_out[2]           1           1      100.00
          3                             Alu_out[1]           1           1      100.00
          3                             Alu_out[0]           1           1      100.00

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

Toggle Coverage      =      100.0% (10 of 10 bins)
```

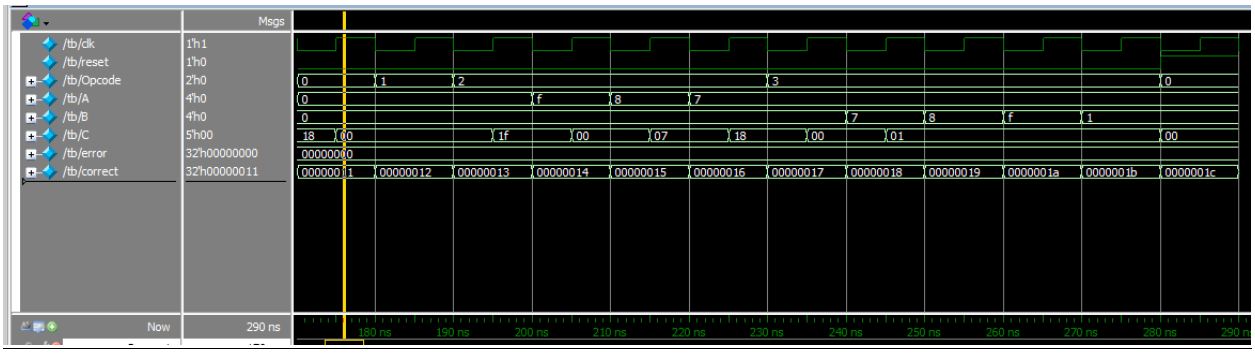
Assertions

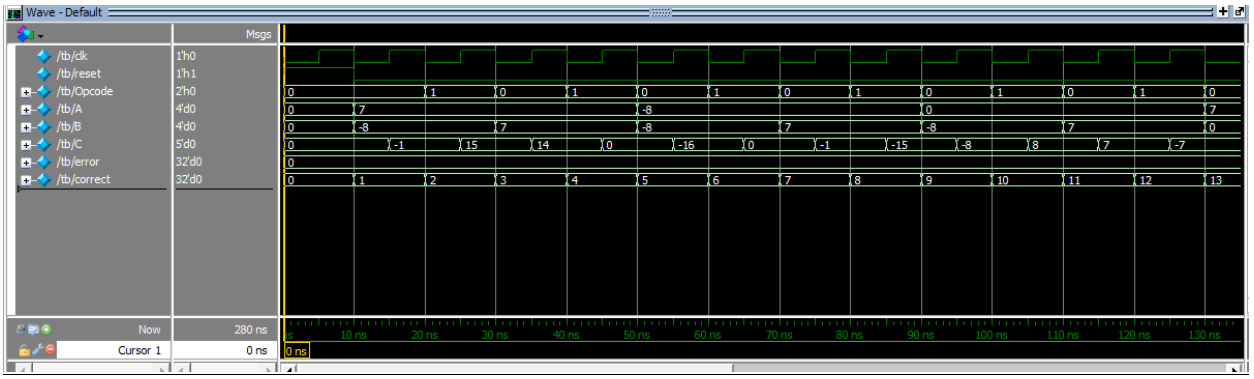
```
1
2   module Asser(if_d.Assertionss if_t);
3
4   property rst_check;
5   @(posedge if_t.clk or posedge if_t.reset)  $rose(if_t.reset) |->if_t.C==0 ;
6   endproperty
7
8
9   property addition_check;
10    @(negedge if_t.clk) disable iff (if_t.reset)
11      (if_t.Opcode ==0) |-> (if_t.C == (if_t.A + if_t.B));
12  endproperty
13
14
15  property subtraction_check;
16   @(negedge if_t.clk) disable iff (if_t.reset)
17      (if_t.Opcode==if_t.Sub) |-> (if_t.C == (if_t.A-if_t.B));
18  endproperty
19
20  property NotA_check;
21  @(negedge if_t.clk) disable iff (if_t.reset)  (if_t.Opcode==if_t.Not_A |->( if_t.C== ~if_t.A));
22  endproperty
23
24  property ReductionOR_B_check;
25  @(negedge if_t.clk) disable iff (if_t.reset)  (if_t.Opcode==if_t.ReductionOR_B |-> (if_t.C== |if_t.B));
26  endproperty
27
28
29  rst:assert property(rst_check)$display("done"); else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
30                                                                                                          ,if_t.C);
31  rst_co:cover property(rst_check);
32
```

```
31  rst_co:cover property(rst_check);
32
33  addition:assert property(addition_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
34                                                                                                  ,if_t.C);
35  addition_co:cover property(addition_check);
36
37  subtraction:assert property(subtraction_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
38                                                                                                  ,if_t.C);
39  subtraction_co:cover property(subtraction_check);
40
41  not_ch:assert property(NotA_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
42                                                                                                ,if_t.C);
43  not_ch_co:cover property(NotA_check);
44
45  oring:assert property(ReductionOR_B_check)else $error("reset=%0d ,Opcode=%0d,A=%0d ,B=%0d ,c=%0d ",if_t.reset,if_t.Opcode,if_t.A,if_t.B
46                                                                                                  ,if_t.C);
47  oring_co:cover property(ReductionOR_B_check);
48
49  endmodule
```

SIMULATION







counter

Design

```
module counter (if_counter.DUT if_t);

always @(posedge if_t.clk or negedge if_t.rst_n) begin
    if (!if_t.rst_n)
        if_t.count_out <= 0;
    else if (!if_t.load_n)
        if_t.count_out <= if_t.data_load;
    else if (if_t.ce)
        if (if_t.up_down)
            if_t.count_out <= if_t.count_out + 1;
        else
            if_t.count_out <= if_t.count_out - 1;
end

assign if_t.max_count = (if_t.count_out == {if_t.WIDTH{1'b1}})? 1:0;
assign if_t.zero = (if_t.count_out == 0)? 1:0;

endmodule
```

Tb

```
import pack::*;
module counter_tb(if_counter if_t);

transaction tr=new();

always_comb begin tr.clk=if_t.clk; end

initial begin

    if_t.rst_n=0;
    @(negedge if_t.clk);
    repeat(200)begin
        assert (tr.randomize());
        if_t.rst_n=tr.rst_n;
        if_t.load_n=tr.load_n;
        if_t.up_down=tr.up_down;
        if_t.ce=tr.ce;
        if_t.data_load=tr.data_load ;
        tr.count_out=if_t.count_out;
        tr.max_count=if_t.max_count;
        tr.zero=if_t.zero;
        @(negedge if_t.clk);#1;
    end

$stop;
end
endmodule
```

interface

```
interface if_counter(clk);

parameter WIDTH = 4;

//input bit clk;
input bit clk;


logic rst_n,load_n,up_down,ce;
logic  [WIDTH-1:0] data_load;
logic  [WIDTH-1:0] count_out;
logic max_count,zero;


modport DUT (input clk,rst_n,load_n,up_down,ce,data_load,output max_count,zero,count_out);
modport TEST (output rst_n,load_n,up_down,ce,data_load,input clk,max_count,zero,count_out);
modport Assertions (input clk,rst_n,load_n,up_down,ce,data_load,output max_count,zero,count_out);

endinterface
```

Top

```
module top();

bit clk=0;

initial begin
forever
    begin
        #100 clk =!clk;
    end
end


if_counter if_t(clk);
counter dut(if_t);
counter_tb tb(if_t);
bind counter asser SVA(if_t);

endmodule
```

Do file

```
vlib work
vlog *v +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave *
coverage save counter_tb.ucdb -onexit
run -all
quit -sim

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

Assertions

```
module asser(if_counter.Assertions if_t);

property load_check;
@(negedge if_t.clk) disable iff (!if_t.rst_n) (! if_t.load_n  |-> if_t.count_out == if_t.data_load );
endproperty

property turn_off_all;
@(negedge if_t.clk) disable iff (!if_t.rst_n) ( (if_t.load_n && !if_t.ce)  |-> $stable(if_t.count_out) );
endproperty

property count_up;
@(negedge if_t.clk) disable iff (!if_t.rst_n) ( (if_t.load_n && if_t.ce &&  if_t.up_down)  |-> if_t.count_out == $past(if_t.count_out) + 1'b1 );
endproperty

property count_down;
@(negedge if_t.clk) disable iff (!if_t.rst_n) ( (if_t.load_n && if_t.ce &&  !if_t.up_down) |-> if_t.count_out == $past(if_t.count_out) - 1'b1 );
endproperty


property maximum;
@(negedge if_t.clk) disable iff (!if_t.rst_n) ( if_t.count_out==4'b1111 |-> if_t.max_count== 1'b1 );
endproperty

property minimum;
@(negedge if_t.clk) disable iff (!if_t.rst_n) ( if_t.count_out==4'b0000 |-> if_t.zero== 1'b1 );
endproperty


// immediate assertion
```

```
always_comb begin
    if(!if_t.rst_n)begin
    rst_check:assert final ( if_t.count_out==0 && if_t.zero==1 && if_t.max_count==0 ) ;
    reset_cover:cover (if_t.count_out==0 && if_t.zero==1 && if_t.max_count==0);
    end
end


//concurent assertion

load_assert:assert property(load_check) else $display("@%0t reset=%0d",$time,if_t.rst_n);
turn_off_all_assert:assert property(turn_off_all)else $display("@%0t reset=%0d",$time,if_t.rst_n);
count_up_assert:assert property(count_up)else $display("@%0t reset=%0d",$time,if_t.rst_n);
count_down_assert:assert property(count_down)else $display("@%0t reset=%0d",$time,if_t.rst_n);
maximum_assert:assert property(maximum)else $display("@%0t reset=%0d",$time,if_t.rst_n);
minimum_assert:assert property(minimum)else $display("@%0t reset=%0d",$time,if_t.rst_n);

//coverage assertion
load_cover:cover property(load_check);
turn_off_all_cover:cover property(turn_off_all);
count_up_cover:cover property(count_up);
count_down_cover:cover property(count_down);
maximum_cover:cover property(maximum);
minimum_cover:cover property(minimum);

endmodule
```

Package

```
package pack;
    parameter WIDTH=4;
    class transaction;

    rand bit rst_n,load_n,up_down,ce;
    rand bit [WIDTH-1:0] data_load;
    bit [WIDTH-1:0] count_out;
    bit zero,max_count,clk;
    constraint x {
        rst_n dist {0:=5,1:=100};
        load_n dist {0:=70,1:=30};
        ce dist {1:=70,0:=30};
    }
    covergroup covgup @(posedge clk) ;
        coverpoint data_load iff(!load_n);
        COUNT_UP:coverpoint count_out iff(rst_n && ce && up_down){
            bins all_values_up[]={[0:15]};
            bins trans_zero=(15=>0);
        }
        COUNT_DOWN:coverpoint count_out iff(rst_n && ce && !up_down){
            bins all_values_down[]={[0:15]};
            bins trans_max=(0=>15);
        }
        max_check :cross count_out,max_count{
            option.cross_auto_bin_max=0;
            bins max_detect=binsof(count_out) intersect{15} && binsof(max_count) intersect {1};
            illegal_bins invalid=binsof(count_out) intersect{[0:14]} && binsof(max_count) intersect {1};
        }
        zeros_check :cross count_out,zero{
            option.cross_auto_bin_max=0;
            bins zero_detect=binsof(count_out) intersect{0} && binsof(zero) intersect {1};
            illegal_bins invalid=binsof(count_out) intersect{[1:15]} && binsof(zero) intersect {1};
```

```
            illegal_bins invalid=binsof(count_out) intersect{[1:15]} && binsof(zero) intersect {1};
        }
    endgroup




    function new();
       covgup=new();
    endfunction

endclass
endpackage
```

Code coverage

```
=== File: counter.sv
============================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            7         7         0     100.0

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

Statement Coverage for file counter.sv --

    1                                           ////////////////////////////////////////////////////////////////////////////////
    2                                           // Author: Kareem Waseem
    3                                           // Course: Digital Verification using SV & UVM
    4                                           //
    5                                           // Description: Counter Design
    6                                           //
    7                                           ////////////////////////////////////////////////////////////////////////////////
    8
    9                                           module counter (if_counter.DUT if_t);
    10
    11              1                    212    always @(posedge if_t.clk or negedge if_t.rst_n) begin
    12                                              if (!if_t.rst_n)
    13              1                     24            if_t.count_out <= 0;
    14                                              else if (!if_t.load_n)
    15              1                    134            if_t.count_out <= if_t.data_load;
    16                                              else if (if_t.ce)
    17                                                  if (if_t.up_down)
    18              1                     18                if_t.count_out <= if_t.count_out + 1;
    19                                                  else
    20              1                     19                if_t.count_out <= if_t.count_out - 1;
    21                                          end
    22
    23              1                    175    assign if_t.max_count = (if_t.count_out == {if_t.WIDTH{1'b1}})? 1:0;
    24              1                    175    assign if_t.zero = (if_t.count_out == 0)? 1:0;
```

```
Branch Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Branches                        10        10         0     100.0

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

Branch Coverage for file counter.sv --

------------------------------------IF Branch------------------------------------
    12                                      212     Count coming in to IF
    12              1                        24         if (!if_t.rst_n)
    14              1                       134         else if (!if_t.load_n)
    16              1                        37         else if (if_t.ce)
                                             17     All False Count
Branch totals: 4 hits of 4 branches = 100.0%

------------------------------------IF Branch------------------------------------
    17                                       37     Count coming in to IF
    17              1                        18             if (if_t.up_down)
    19              1                        19             else
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    23                                      174     Count coming in to IF
    23              1                         6     assign if_t.max_count = (if_t.count_out == {if_t.WIDTH{1'b1}})? 1:0;
    23              2                       168     assign if_t.max_count = (if_t.count_out == {if_t.WIDTH{1'b1}})? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%

------------------------------------IF Branch------------------------------------
    24                                      174     Count coming in to IF
    24              1                        18     assign if_t.zero = (if_t.count_out == 0)? 1:0;
    24              2                       156     assign if_t.zero = (if_t.count_out == 0)? 1:0;
Branch totals: 2 hits of 2 branches = 100.0%
```

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

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

Toggle Coverage for File counter.sv --

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

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

Toggle Coverage      =      100.0% (0 of 0 bins)

================================================================================
=== File: counter_tb.sv
================================================================================
Statement Coverage:
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           17        17         0     100.0
```

Assertions

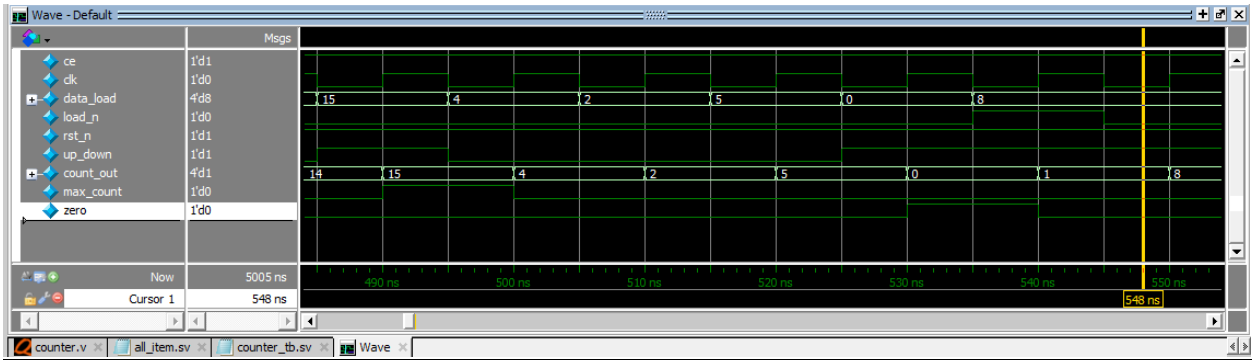
```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Count Status
                                      Unit   UnitType
--------------------------------------------------------------------------------------------
/top/dut/SVA/reset_cover              asser  Verilog  SVA  asser.sv(33)       11 Covered
/top/dut/SVA/load_cover               asser  Verilog  SVA  asser.sv(48)      134 Covered
/top/dut/SVA/turn_off_all_cover       asser  Verilog  SVA  asser.sv(49)       17 Covered
/top/dut/SVA/count_up_cover           asser  Verilog  SVA  asser.sv(50)       18 Covered
/top/dut/SVA/count_down_cover         asser  Verilog  SVA  asser.sv(51)       19 Covered
/top/dut/SVA/maximum_cover            asser  Verilog  SVA  asser.sv(52)        6 Covered
/top/dut/SVA/minimum_cover            asser  Verilog  SVA  asser.sv(53)       13 Covered

TOTAL DIRECTIVE COVERAGE: 100.0%  COVERS: 7

ASSERTION RESULTS:
-----------------------------------------------------
Name                 File(Line)           Failure Pass
                                          Count   Count
-----------------------------------------------------
/top/dut/SVA/rst_check
                     asser.sv(32)               0     1
/top/dut/SVA/load_assert
                     asser.sv(40)               0     1
/top/dut/SVA/turn_off_all_assert
                     asser.sv(41)               0     1
/top/dut/SVA/count_up_assert
                     asser.sv(42)               0     1
/top/dut/SVA/count_down_assert
                     asser.sv(43)               0     1
/top/dut/SVA/maximum_assert
                     asser.sv(44)               0     1
/top/dut/SVA/minimum_assert
                     asser.sv(45)               0     1
/top/tb/#ublk#95084642#12/immed__13
```

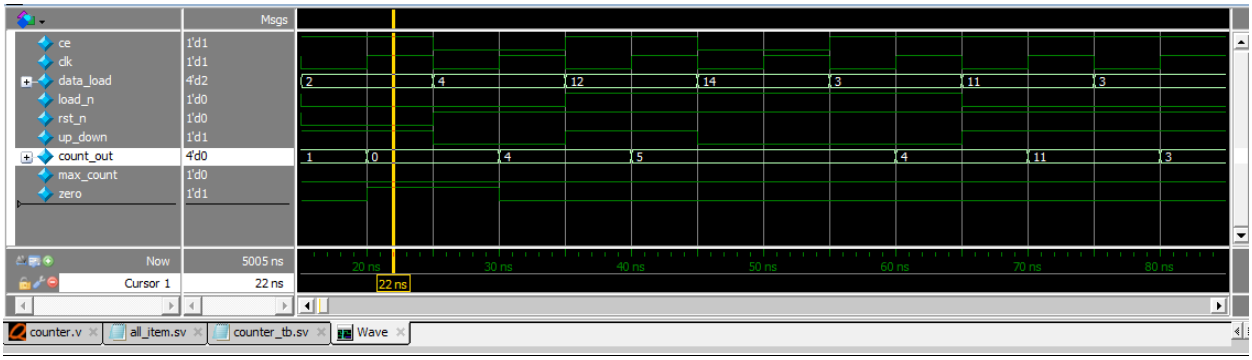
```
-----------------------------------------------------------
Name                   File(Line)           Failure Pass
                                            Count   Count
-----------------------------------------------------------
/top/dut/SVA/rst_check
                       asser.sv(32)               0     1
/top/dut/SVA/load_assert
                       asser.sv(40)               0     1
/top/dut/SVA/turn_off_all_assert
                       asser.sv(41)               0     1
/top/dut/SVA/count_up_assert
                       asser.sv(42)               0     1
/top/dut/SVA/count_down_assert
                       asser.sv(43)               0     1
/top/dut/SVA/maximum_assert
                       asser.sv(44)               0     1
/top/dut/SVA/minimum_assert
                       asser.sv(45)               0     1
/top/tb/#ublk#95084642#12/immed__13
                       counter_tb.sv(13)          0     1

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

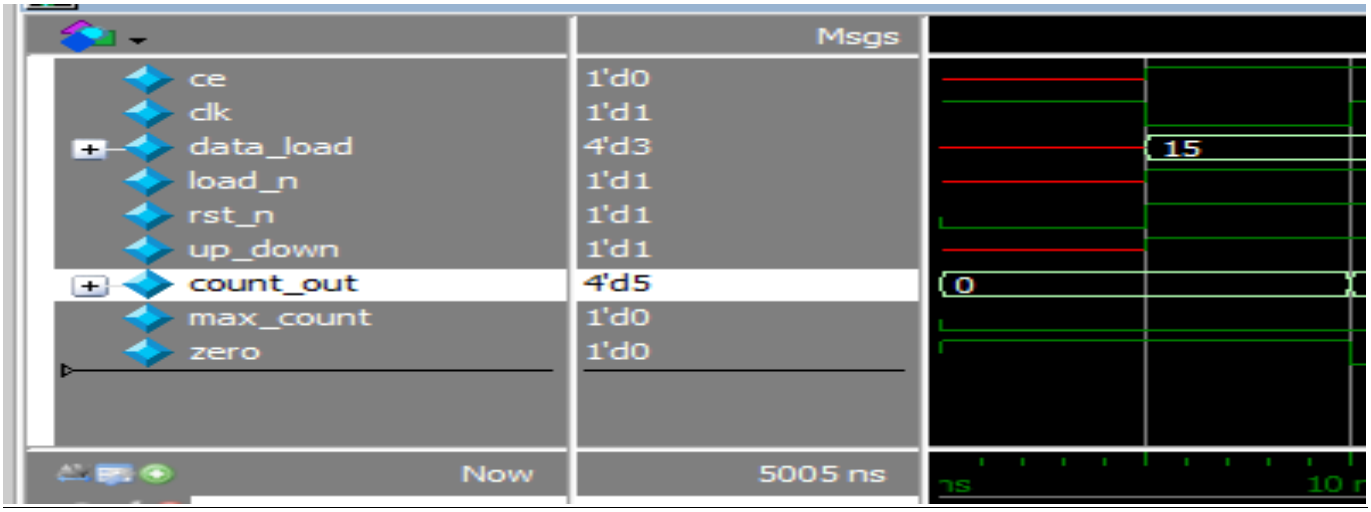
Functional coverage

```
Toggle Coverage      =      100.0% (2 of 2 bins)

COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal    Status

----------------------------------------------------------------------------------------------
 TYPE /pack/transaction/covgup                      100.0%        100    Covered
    covered/total bins:                                 72         72
    missing/total bins:                                  0         72
    % Hit:                                          100.0%        100
    Coverpoint covgup::data_load                    100.0%        100    Covered
        covered/total bins:                             16         16
        missing/total bins:                              0         16
        % Hit:                                      100.0%        100
    Coverpoint covgup::COUNT_UP                     100.0%        100    Covered
        covered/total bins:                             17         17
        missing/total bins:                              0         17
        % Hit:                                      100.0%        100
    Coverpoint covgup::COUNT_DOWN                   100.0%        100    Covered
        covered/total bins:                             17         17
        missing/total bins:                              0         17
        % Hit:                                      100.0%        100
    Coverpoint covgup::count_out                    100.0%        100    Covered
        covered/total bins:                             16         16
        missing/total bins:                              0         16
        % Hit:                                      100.0%        100
    Coverpoint covgup::zero                         100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
        % Hit:                                      100.0%        100
    Coverpoint covgup::max_count                    100.0%        100    Covered
        covered/total bins:                              2          2
        missing/total bins:                              0          2
```

SIMULATION







VERIFICATION PLAN

	Label	Description	Stimulus Generation	Functionality Check	functionality coverage
2	COUNTER_1	When the reset is asserted, the output counter value should be low then deassert reset	Directed at the start of the simulation then it randomized with cosntraint to be of high 95 % from time	check with immediate assertion for asyn reset	-
3	COUNTER_2	when load_n is asserted to low -> the output count out should take same value of load data	Randomization with constrain that Load_n should be low(active) for 70% of time,we can check this on time 75 ns	check with concurent assertion on out with data load	
4	COUNTER_3	when load_n is asserted to low and load data is equal to 15 -> the output count out should take 15 and max count output should be 1	Randomization,we can check this on time 495 ns	check with concurent assertion on out with data load	cover all values of load data when load_n asserted
5	COUNTER_4	when load_n is asserted to low and load data is equal to 0 -> the output count out should take 0 and zero output should be 1	Randomization,we can check this on time 535 ns	check with concurent assertion on out with data load	
6	COUNTER_5	we assert load_n to high and we assert enaple input to high and count up -> the output count _out shoult increase every clock cycle	Randomization we can check this in most of simulation	check with concurent assertion on output increment	
7	COUNTER_6	we assert load_n to high and we assert enaple input to high and count down-> the output count _out shoult decrease every clock cycle	Randomization under constrain that ec should be HIgh(active) for 70% of time,	check with concurent assertion on output decrement	we cover all values of count out in count up and transation from 15 to 0
8	COUNTER_7	we check on state when counter reach 15 through counting and check output max count if =1	Randomization	check with concurent assertion on max state	
9	COUNTER_8	we check on state when counter reach 0 through counting and check output Zero count if =1	Randomization,we can check this on time 790 ns	check with concurent assertion on zero state	we cover all values of count out in count down and transation from 0 to 15
10	COUNTER_9	we check when reset is high and load_n is high and enaple is low the output shouldnt change and keep previous value	Randomization through simmulation	check with immediate assertion for asyn reset	