

ALSU

Design

```
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 assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
19 assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
20 assign invalid = invalid_red_op | invalid_opcode;
21
22 //Registering input signals
23 always @(posedge clk or posedge rst) begin
24     if(rst) begin
25         cin_reg <= 0;
26         red_op_B_reg <= 0;
27         red_op_A_reg <= 0;
28         bypass_B_reg <= 0;
29         bypass_A_reg <= 0;
30         direction_reg <= 0;
31         serial_in_reg <= 0;
32         opcode_reg <= 0;
```

```
32         opcode_reg <= 0;
33         A_reg <= 0;
34         B_reg <= 0;
35
36     end else begin
37         cin_reg <= cin;
38         red_op_B_reg <= red_op_B;
39         red_op_A_reg <= red_op_A;
40         bypass_B_reg <= bypass_B;
41         bypass_A_reg <= bypass_A;
42         direction_reg <= direction;
43         serial_in_reg <= serial_in;
44         opcode_reg <= opcode;
45         A_reg <= A;
46         B_reg <= B;
47
48     end
49 end
50
51 //leds output blinking
52 always @(posedge clk or posedge rst) begin
53     if(rst) begin
54         leds <= 0;
55     end else begin
56         if (invalid)
57             leds <= ~leds;
58         else
59             leds <= 0;
60     end
61 end
```

```

63 //ALU 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                 end
106                 3'h3: out <= A_reg * B_reg;
107                 3'h4: begin
108                     if (direction_reg)
109                         out <= {out_next[4:0], serial_in_reg};
110                     else
111                         out <= {serial_in_reg, out_next[5:1]};
112                 end
113                 3'h5: begin
114                     if (direction_reg)
115                         out <= {out_next[4:0], out_next[5]};
116                     else
117                         out <= {out_next[0], out_next[5:1]};
118                 end

```

```

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

```
1  import pack_alu::*;
2
3  module ALSU_tb ();
4  parameter INPUT_PRIORITY = "B";
5  parameter FULL_ADDER = "ON";
6
7  bit clk=0, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
8  Opcode_e opcode;
9  bit signed [2:0] A, B;
10 bit [15:0] leds;
11 bit signed [5:0] out;
12 bit [15:0] leds_exp;
13 bit signed [5:0] out_exp;
14
15 //register internal
16 reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
17 reg signed [1:0] cin_reg;
18 reg [2:0] opcode_reg;
19 reg signed [2:0] A_reg, B_reg; //change to signed
20
21
22
23 int error =0, correct=0;
24 bit invalid_t, x1, x2;
25 bit signed [5:0] last_out=0;
26
27 ALSU #(.INPUT_PRIORITY(INPUT_PRIORITY), .FULL_ADDER(FULL_ADDER)) tb (.*);
28
```

```
30 always #10 clk=!clk;
31 transaction tr=new();
32
33
34 initial begin
35     assert_rst();
36     check_rst();
37
38     tr.constraint_mode(0);
39     tr.x.constraint_mode(1);
40     repeat(300) begin
41         assert(tr.randomize());
42         init(tr);
43         if(rst)begin
44             check_rst();
45             reset_internal();
46         end
47         else begin
48             check_result();
49             sampling(tr);
50         end
51     end
52 end
53
54 tr.constraint_mode(0);
55 tr.y.constraint_mode(1);
56 rst=0; bypass_A=0; bypass_B=0; red_op_A=0; red_op_B=0;
57 tr.rst.rand_mode(0); tr.bypass_A.rand_mode(0); tr.bypass_B.rand_mode(0); tr.red_op_A.rand_mode(0);
58 tr.red_op_B.rand_mode(0);
59 init(tr);
60
```

```

61  ✓   for(int i=0;i<1000;i++)begin
62      assert(tr.randomize());
63      cin=tr.cin;
64      direction=direction_reg;
65      serial_in=serial_in_reg;
66      A=tr.A;
67      B=tr.B;
68  ✓   if('{OR,XOR,ADD,MULT,SHIFT,ROTATE} ==tr.arr)$display("@%0t the wanted sequence is %p",$time,tr.arr);
69  ✓       foreach(tr.arr[j])begin
70           tr.opcode=tr.arr[j];
71           opcode=tr.arr[j];
72           tr.out=out;
73           tr.leds=leds;
74           check_result();
75           sampling(tr);
76       end
77   end
78   $display("number of correct =%0d ,error=%0d",correct,error);
79 $stop;
80
81 end

```

```

83 task check_result();
84
85     golden_model();
86     @(negedge clk);
87     if(out_exp!= out && leds_exp != leds)begin
88         $display("@%0t there is error out=%0b ,leds=%0b " , $time ,tr.out,tr.leds);error++;
89     end
90     else
91         correct++;
92
93
94 endtask
95
96 task golden_model();
97     if(is_invalid())
98         leds_exp= ~leds_exp;
99     else
100         leds_exp=0;
101     if(bypass_A_reg && bypass_B_reg)begin
102         if (INPUT_PRIORITY== "A")
103             out_exp = A_reg;
104         else if (INPUT_PRIORITY== "B")
105             out_exp = B_reg;
106         end
107         // check on pypass operations
108         else if(bypass_A) //check on bypass
109             out_exp = A_reg;
110         else if(bypass_B)
111             out_exp = B_reg;
112         //check on invalid_t condition output
113         else if(is_invalid()) begin

```

```

114         out_exp=0;
115     end
116     else begin
117         //here we check on OP code
118         case (opcode_reg)
119         OR:begin// check on priority first
120             if(red_op_A_reg && red_op_B_reg && INPUT_PRIORITY== "A")
121                 out_exp = A_reg;
122             else if(red_op_A_reg && red_op_B_reg&& INPUT_PRIORITY== "B")
123                 out_exp = B_reg;
124             else if(red_op_A_reg)
125                 out_exp = (|A_reg);
126             else if(red_op_B_reg)
127                 out_exp = (|B_reg);
128             else
129                 out_exp = (A_reg|B_reg);
130         end
131         XOR:begin
132             if(red_op_A_reg && red_op_B_reg && INPUT_PRIORITY== "A")
133                 out_exp = A_reg;
134             else if(red_op_A_reg && red_op_B_reg&& INPUT_PRIORITY== "B")
135                 out_exp = B_reg;
136             else if(red_op_A_reg)
137                 out_exp = (^A_reg);
138             else if(red_op_B_reg)
139                 out_exp = (^B_reg);
140             else
141                 out_exp = (A_reg^B_reg);
142         end
143         ADD:begin
144             if(FULL_ADDER == "ON")

```

```

145                 out_exp= A_reg+B_reg+cin_reg;
146             else if(FULL_ADDER == "OFF")
147                 out_exp= A_reg+B_reg;
148         end
149         MULT:begin
150             out_exp= A_reg*B_reg;
151         end
152         SHIFT:begin
153             if(direction_reg)
154                 out_exp = {out_exp[4:0],serial_in_reg};
155             else if(!direction_reg)
156                 out_exp = {serial_in_reg,out_exp[5:1]};
157         end
158         ROTATE:begin
159             if(direction_reg)
160                 out_exp = {out_exp[4:0],out_exp[5]};
161             else if(!direction_reg)
162                 out_exp = {out_exp[0],out_exp[5:1]};
163         end
164     endcase
165 end
166     update_internals();
167 endtask
168
169
170 task update_internals();
171     red_op_A_reg=red_op_A; red_op_B_reg=red_op_B; bypass_A_reg=bypass_A;
172     bypass_B_reg=bypass_B; direction_reg=direction; serial_in_reg=serial_in;
173     cin_reg=cin;
174     opcode_reg=opcode;
175     A_reg=A;B_reg=B;
176 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

```
1 package pack_alu;
2 typedef enum { OR=0,XOR,ADD,MULT,SHIFT,ROTATE,INVALID6,INVALID7 } Opcode_e;
3 typedef enum {MAXPOS=3,MAXNEG=-4,ZERO=0}corner_state_e;
4
5 class transaction;
6     rand bit clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7     rand Opcode_e opcode;
8     rand bit signed [2:0] A, B;
9     bit [2:0] ones_number={3'b001,3'b010,3'b100};
10    rand bit [2:0] found,notfound;
11    rand corner_state_e a_state;
12    rand bit [2:0] rem_numbers;
13    bit signed [5:0] out;
14    bit [15:0] leds;
15
16    rand Opcode_e arr[6];
17
18    constraint x {
19
20        rem_numbers!= MAXPOS||MAXNEG||ZERO;
21
22        rst dist {1:=5 , 0:=95};
23
24        found inside {ones_number};
25        !(notfound inside {ones_number});
26
27        if (opcode ==ADD || opcode== MULT){
28            A dist {a_state:=80,rem_numbers:=20};
29            B dist {a_state:=80,rem_numbers:=20};
30        }
31        if (opcode ==OR || opcode== XOR ){
32            if(red_op_A){
```

```
32            if(red_op_A){
33                A dist {found:=80,notfound:=20};
34                B==3'b000;
35            }
36            else if (red_op_B){
37                B dist {found:=80,notfound:=20};
38                A==3'b000;
39            }
40        }
41
42
43        opcode dist {[OR:ROTATE]:=80,[INVALID6:INVALID7]};
44
45        bypass_A dist {0:=90,1:=10};
46        bypass_B dist {0:=90,1:=10};
47    }
48    constraint y{
49        unique{arr};
50        foreach(arr[i])
51            arr[i] inside {[OR:ROTATE]};
52    }
53
54
55    covergroup cvr_gp;
56        special:coverpoint opcode{
57            option.weight=0;
58            bins operataions[]={[OR:ROTATE]};
59        }
60        CB1:coverpoint A{
61            bins A_data_0={0};
```

```

60 CB1:coverpoint A{
61     bins A_data_0={0};
62     bins A_data_max={MAXPOS};
63     bins A_data_min={MAXNEG};
64     bins A_data_walkingones[] = {3'b001,3'b010,3'b100} iff (red_op_A);
65     bins A_data_default=default;
66 }
67 CB2:coverpoint B{
68     bins B_data_0={0};
69     bins B_data_max={MAXPOS};
70     bins B_data_min={MAXNEG};
71     bins B_data_walkingones[] = {3'b001,3'b010,3'b100} iff (red_op_B);
72     bins B_data_default=default;
73 }
74 A_M:coverpoint opcode{
75     bins Bins_arith[] = {ADD,MULT};
76 }
77 CB3:coverpoint opcode{
78     bins Bins_shift[] = {SHIFT,ROTATE};
79     bins Bins_arith[] = {ADD,MULT};
80     bins Bins_bitwise[] = {OR,XOR};
81     illegal_bins Bins_invalid = {INVALID6,INVALID7};
82     bins Bins_trans=(OR=>XOR=>ADD=>MULT=>SHIFT=>ROTATE);
83 }
84 //1
85 corner_case:cross A_M,CB2,CB1{
86     ignore_bins walkingA=binsof(CB1.A_data_walkingones) ;
87     ignore_bins walkingB=binsof(CB2.B_data_walkingones) ;
88 }
89 //2

```

```

90 Addition:cross cin,special{
91     option.cross_auto_bin_max=0;
92     bins Add_cin0=binsof(special) intersect {ADD} && binsof(cin) intersect {0};
93     bins Add_cin1=binsof(special) intersect {ADD}&& binsof(cin) intersect {1};
94 }
95 //3
96 shift:cross special,serial_in{
97     option.cross_auto_bin_max=0;
98     bins shift_Si0=binsof(special) intersect {SHIFT} && binsof(serial_in) intersect {0};
99     bins shift_Si1=binsof(special) intersect {SHIFT} && binsof(serial_in) intersect {1};
100 }
101 //4
102 shift_rotate:cross CB3,direction{
103     option.cross_auto_bin_max=0;
104     bins shu_rot_d0=binsof(CB3.Bins_shift) && binsof(direction) intersect {0} ;
105     bins shu_rot_d1=binsof(CB3.Bins_shift) && binsof(direction) intersect {1} ;
106 }
107 //5
108 walkingones:cross CB3,CB2,CB1{
109     option.cross_auto_bin_max=0;
110     bins arithA=binsof(CB3.Bins_bitwise) && binsof(CB2.B_data_0) &&binsof(CB1.A_data_walkingones) ;
111     bins arithB=binsof(CB3.Bins_bitwise) && binsof(CB1.A_data_0) &&binsof(CB2.B_data_walkingones) ;
112 }
113 invalidation:cross red_op_A,red_op_B,special{
114     option.cross_auto_bin_max=0;
115     bins ROpA_notXoR=binsof(special) intersect {[ADD:ROTATE]} && binsof(red_op_A) intersect{1};
116     bins ROpB_notXoR=binsof(special) intersect {[ADD:ROTATE]} && binsof(red_op_B) intersect{1};
117 }
118
119 endgroup

```

```

121 function new();
122     cvr_gp=new();
123 endfunction
124
125 endclass
126 endpackage

```


Do file

```

\lib work
vlog ALSU.v ALSU_tb.sv pack_alu.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 ignore 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 constraints 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 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 coverage 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_alu/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	
Crbss 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::Addition	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	
<hr/>			
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_alu::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

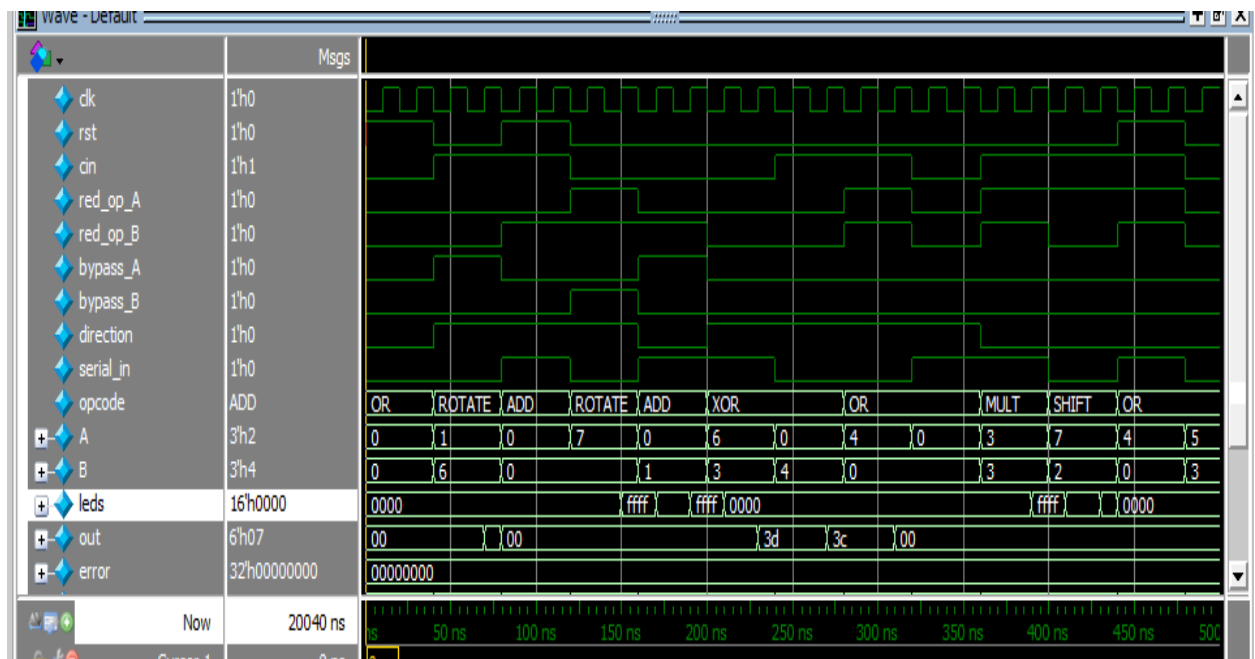
[1] - Does not contribute coverage as weight is 0

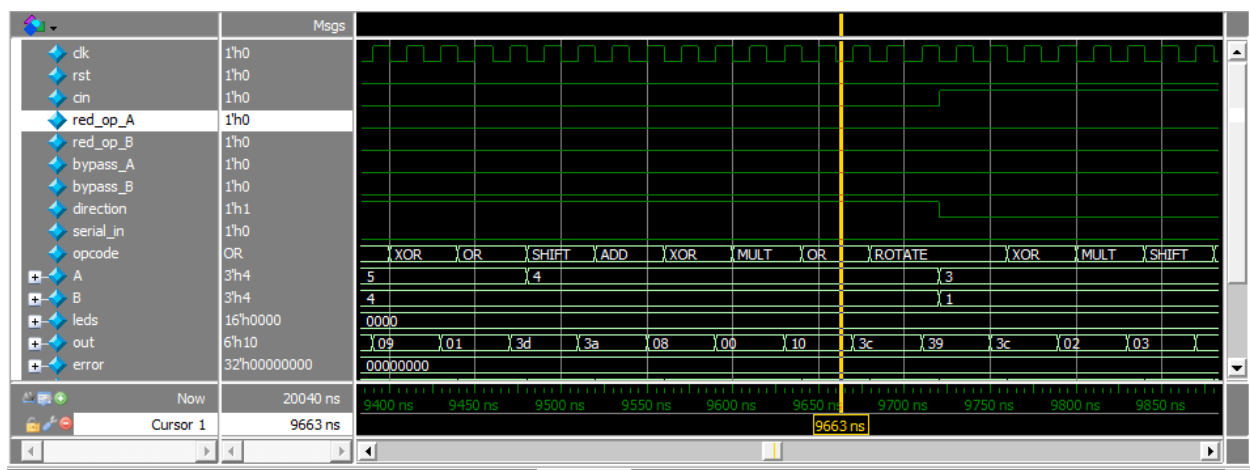
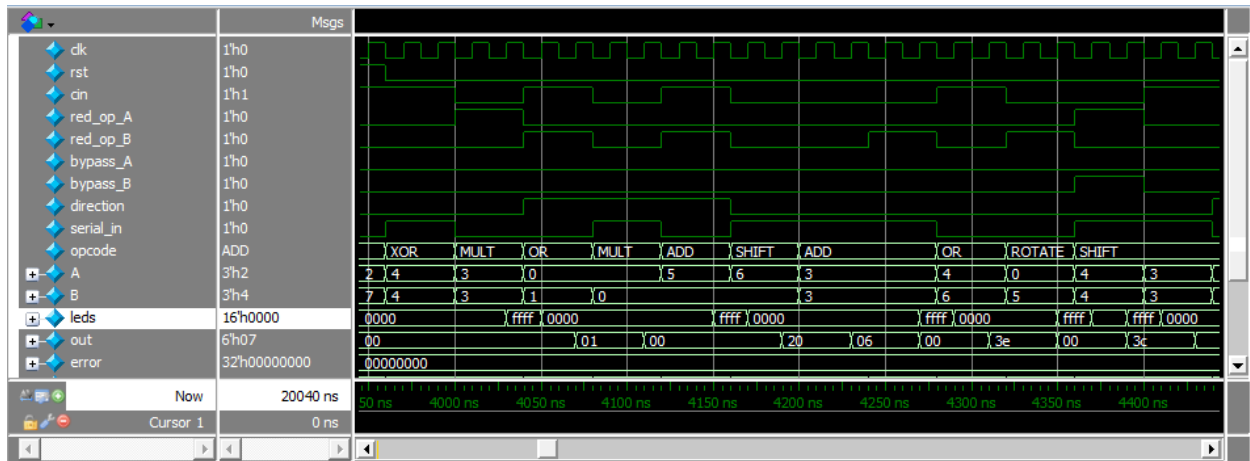
TOTAL COVERGROUP COVERAGE: 100.0% COVERGROUP TYPES: 1

ASSERTION RESULTS:

Name	File(Line)	Failure Count	Pass Count
------	------------	---------------	------------

Simulation





Ex2

```
1  module Assertion_ex2 ();
2  logic a,b,c,valid;
3  logic [3:0]d;
4  bit clk=0;
5  logic[7:0]V;
6
7  always #100 clk=!clk;
8
9  property x1;
10 |   @(posedge clk)(a |-> ##2 b);
11 endproperty
12
13 property x2;
14 @(posedge clk)(a && b |-> ##[1:3] c);
15 endproperty
16
17 sequence s11b;
18 ##2 !b;
19 endsequence
20
21 property x3;
22 @(posedge clk) s11b;
23 endproperty
24
25 property x4;
26 @(posedge clk) $onehot(V);
27 endproperty
28
29 property x5;
30 @(posedge clk) ($countones(d)==0 |-> ##1 !valid);
31
32 endproperty
```

```
32 endproperty
33 a1:assert property(x1) ;
34 a1_c:cover property(x1);
35
36 a2:assert property(x2) ;
37 a2_c:cover property(x2);
38
39 a3:assert property(x3) ;
40 a3_c:cover property(x3);
41
42 a4:assert property(x4) ;
43 a4_c:cover property(x4);
44
45 a5:assert property(x5) ;
46 a5_c:cover property(x5);
47 |
48 initial begin
49 |   a=1;b=0;c=0;V='d1;
50 |   repeat(2)@(negedge clk);
51 |   a=1;b=1;
52 |   repeat(2)@(negedge clk);
53 |   c=1;V=0;d=4'b0000;valid=1;
54 |   repeat(2)@(negedge clk);
55 |   valid=0;
56 |   repeat(2)@(negedge clk);
57 |   $stop;
58 end
59 endmodule
60
```

▼	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

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

Tb

```
1  module priority_enc_tb(if_pe.TB if_t);
2
3
4      initial begin
5
6          if_t.D=1;
7          if_t.rst=1;
8          @(negedge if_t.clk);#2;
9          if_t.rst=0;
10
11      for(int i=0;i<16;i++)begin
12          if_t.D=i;
13          @(negedge if_t.clk);
14      end
15      if_t.D=0;
16      if_t.rst=1;
17
18      repeat(2) @(negedge if_t.clk);
19      $stop;
20  end
21
22
23  endmodule
```

VERIFICATION PLAN

1	Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
2	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 concurrent assertion on output	A checker in the testbench to make sure the output is correct
3	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 concurrent assertion on output	A checker in the testbench to make sure the output is correct
4	PRIORTY_	we assert the reset to 0 to see the stimulus on Output Y and valid	Directed at the simulation	check with concurrent assertion on output	A checker in the testbench to make sure the output is correct
5	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 concurrent assertion on output	A checker in the testbench to make sure the output is correct

interface

```

1  interface if_pe(clk);
2  input bit clk;
3  logic rst;
4  logic [3:0] D;
5  logic [1:0] Y;
6  logic valid;
7
8  modport DUT (input clk,rst,D,output Y,valid);
9  modport TB (output clk,rst,D,input Y,valid);
10 modport Assertions (input clk,rst,D,output Y,valid);
11 endinterface

```

assertions

```
1 module asser(if_pe.Assertions if_t);
2
3 property rst;
4 @(negedge if_t.clk) if_t.rst==1'b1 |-> ( if_t.Y==2'b00 && if_t.valid ==1'b0);
5 endproperty
6
7 property valid_bit;
8 @(negedge if_t.clk) disable iff(if_t.rst) (($countones(if_t.D) > 0) |-> if_t.valid);
9 endproperty
10
11 property Bit_0;
12 @(negedge if_t.clk) disable iff(if_t.rst)
13 | if_t.D[0] |-> if_t.Y==2'b11;
14 endproperty
15
16 property Bit_1;
17 @(negedge if_t.clk) disable iff(if_t.rst)
18 |
19 | (if_t.D[1] && ! if_t.D[0]) |-> if_t.Y==2'b10 ;
20 endproperty
21
22 property Bit_2;
23 @(negedge if_t.clk) disable iff(if_t.rst)
24 | (if_t.D[2] && !if_t.D[1] && ! if_t.D[0] ) |-> if_t.Y==2'b01 ;
25 endproperty
26
27 property Bit_3;
28 @(negedge if_t.clk) disable iff(if_t.rst)
29 | ( if_t.D[3] && !if_t.D[2] && !if_t.D[1] && ! if_t.D[0] ) |-> if_t.Y==2'b00 ;
30 endproperty
```

```
31 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);
32 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);
33
34 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);
35 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);
36 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);
37 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);
38
39 rst_check_cover:cover property (rst);
40 valid_bit_check_cover:cover property (valid_bit);
41
42 output_check_cover0:cover property (Bit_0);
43 output_check_cover1:cover property (Bit_1);
44 output_check_cover2:cover property (Bit_2);
45 output_check_cover3:cover property (Bit_3);
46 endmodule
```

Top

```
1 module top();
2 bit clk=0;
3
4 always #5 clk=!clk;
5
6 if_pe if_t(clk);
7 priority_enc dut(if_t);
8 priority_enc_tb tb(if_t);
9 asser ASV(if_t);
10
11
12 endmodule
```

Do file

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

```
vcover report prioty_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:

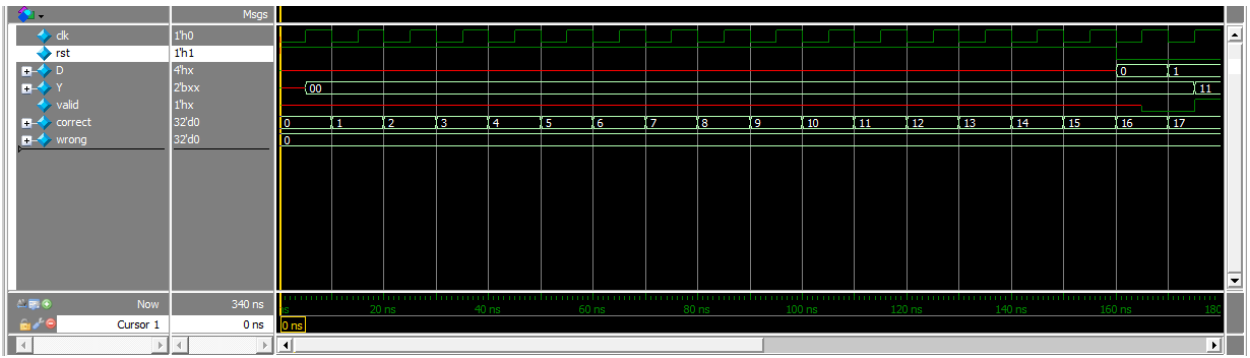
Name	Design Unit	Design UnitType	Lang	File(Line)	Count	Status
/top/ASV/rst_check_cover	asser	Verilog	SVA	asser.sv(39)	2	Covered
/top/ASV/valid_bit_check_cover	asser	Verilog	SVA	asser.sv(40)	14	Covered
/top/ASV/output_check_cover0	asser	Verilog	SVA	asser.sv(42)	7	Covered
/top/ASV/output_check_cover1	asser	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 Count	Pass 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

```
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      always @* begin
6          case (if_t.Opcode)
7              if_t.Add:    Alu_out = if_t.A + if_t.B;
8              if_t.Sub:    Alu_out = if_t.A - if_t.B;
9              if_t.Not_A:   Alu_out = ~if_t.A;
10             if_t.ReductionOR_B: Alu_out = |if_t.B;
11             default:      Alu_out = 5'b0;
12         endcase
13     end // always @ *
14
15     // Register output C
16     always @(posedge if_t.clk or posedge if_t.reset) begin
17         if (if_t.reset)
18             if_t.C <= 5'b0;
19         else
20             if_t.C <= Alu_out;
21         end
22     end
23 endmodule
```

Tb

```
1  import pack_file::*;
2  module ALU_tb (if_d.TB if_t);
3
4  transaction tr=new();
5  initial begin
6      if_t.reset=1;
7      @(negedge if_t.clk);
8      if_t.A=0;if_t.B=0;if_t.Opcode=0;
9      if_t.reset=0;
10     repeat(50) begin
11         assert(tr.randomize());
12         if_t.reset=tr.reset;
13         if_t.Opcode=tr.Opcode;
14         if_t.A=tr.A;
15         if_t.B=tr.B;
16         @(negedge if_t.clk);
17     end
18
19     end
20 $stop;
21 end
22
23
24
25 endmodule
26
```

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	check with concurrent assertion on reset
3	ADD_1	When A is asserted to Max positive 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 concurrent assertion for output on addition
4	SUB_1	When A is asserted to Max positive 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 concurrent assertion for output on subtraction
5	ADD_2	When A is asserted to Max positive number and B asserted to Max positive number and OPcode is ADD the output C should be equal 14	Directed at the simulation at time 30	check with concurrent assertion for output on addition
6	SUB_2	When A is asserted to Max positive number and B asserted to Max positive number and OPcode is SUB the output C should be equal 0	Directed at the simulation at time 40	check with concurrent 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 concurrent 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 concurrent assertion for output on subtraction
9	ADD_4	When A is asserted to Max negative number and B asserted to Max positive number OP=ADD the output C should be equal -1	Directed at the simulation at time 70	check with concurrent assertion for output on addition
10	SUB_4	When A is asserted to Max negative number and B asserted to Max positive number OP=SUB the output C should be equal -15	Directed at the simulation at time 80	check with concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent 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 concurrent assertion on reset
31	STOP	we end simulation here by \$stop	Directed at the simulation at time 290	

interface

```
1
2  interface if_d(clk);
3  input bit clk;
4  logic reset;
5  logic [1:0]Opcode;
6  logic signed [3:0] A;
7  logic signed [3:0] B;
8  logic signed [4:0] C;
9
10
11  localparam Add          = 2'b00; // A + B
12  localparam Sub          = 2'b01; // A - B
13  localparam Not_A        = 2'b10; // ~A
14  localparam ReductionOR_B = 2'b11; // |B
15
16
17  modport DUT (input clk,reset,Opcode,A,B,output C);
18  modport TB (output clk,reset,Opcode,A,B,input C);
19  modport Assertionss (input clk,reset,Opcode,A,B,output C);
20  endinterface
21
```

Top

```
1  module top();
2  bit clk=0;
3
4  always #50 clk=!clk;
5
6  if_d if_t(clk);
7  ALU_4_bit dut(if_t);
8  ALU_tb tb(if_t);
9  Asser ASV(if_t);
10
11
12  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

```
\\lib 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
```

SIMULATION



counter

Design

```
9  module counter (if_counter.DUT if_t);
10
11  always @(posedge if_t.clk or negedge if_t.rst_n) begin
12      if (!if_t.rst_n)
13          if_t.count_out <= 0;
14      else if (!if_t.load_n)
15          if_t.count_out <= if_t.data_load;
16      else if (if_t.ce)
17          if (if_t.up_down)
18              if_t.count_out <= if_t.count_out + 1;
19          else
20              if_t.count_out <= if_t.count_out - 1;
21  end
22
23  assign if_t.max_count = (if_t.count_out == {if_t.WIDTH{1'b1}})? 1:0;
24  assign if_t.zero = (if_t.count_out == 0)? 1:0;
25
26  endmodule
```

Tb

```
1  import pack::*;
2  module counter_tb(if_counter if_t);
3
4  transaction tr=new();
5
6  always_comb begin tr.clk=if_t.clk; end
7
8  initial begin
9
10     if_t.rst_n=0;
11     @(negedge if_t.clk);
12     repeat(200)begin
13         assert (tr.randomize());
14         if_t.rst_n=tr.rst_n;
15         if_t.load_n=tr.load_n;
16         if_t.up_down=tr.up_down;
17         if_t.ce=tr.ce;
18         if_t.data_load=tr.data_load ;
19         tr.count_out=if_t.count_out;
20         tr.max_count=if_t.max_count;
21         tr.zero=if_t.zero;
22         @(negedge if_t.clk);#1;
23     end
24
25     $stop;
26 end
27 endmodule
```

interface

```
1  interface if_counter(clk);
2
3  |
4
5  parameter WIDTH = 4;
6
7  //input bit clk;
8  input bit clk;
9
10
11  logic rst_n,load_n,up_down,ce;
12  logic [WIDTH-1:0] data_load;
13  logic [WIDTH-1:0] count_out;
14  logic max_count,zero;
15
16
17  modport DUT (input clk,rst_n,load_n,up_down,ce,data_load,output max_count,zero,count_out);
18  modport TEST (output rst_n,load_n,up_down,ce,data_load,input clk,max_count,zero,count_out);
19  modport Assertions (input clk,rst_n,load_n,up_down,ce,data_load,output max_count,zero,count_out);
20
21  endinterface
```

Top

```
2  module top();
3
4  bit clk=0;
5
6  initial begin
7  ✓ forever
8  ✓   begin
9     #100 clk =!clk;
10    end
11  end
12
13
14  if_counter if_t(clk);
15  counter dut(if_t);
16  counter_tb tb(if_t);
17  bind counter asser SVA(if_t);
18
19  endmodule
```

Do file

```
\lib 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

```
1  module asser(if_counter.Assertions if_t);
2
3  property load_check;
4  @(negedge if_t.clk) disable iff (!if_t.rst_n) ( !if_t.load_n |-> if_t.count_out == if_t.data_load );
5  endproperty
6
7  property turn_off_all;
8  @(negedge if_t.clk) disable iff (!if_t.rst_n) ( (if_t.load_n && !if_t.ce) |-> $stable(if_t.count_out) );
9  endproperty
10
11 property count_up;
12 @(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 );
13 endproperty
14
15 property count_down;
16 @(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 );
17 endproperty
18
19
20 property maximum;
21 @(negedge if_t.clk) disable iff (!if_t.rst_n) ( if_t.count_out==4'b1111 |-> if_t.max_count== 1'b1 );
22 endproperty
23
24 property minimum;
25 @(negedge if_t.clk) disable iff (!if_t.rst_n) ( if_t.count_out==4'b0000 |-> if_t.zero== 1'b1 );
26 endproperty
27
28
29 // immediate assertion
```

```
30 always_comb begin
31     if(!if_t.rst_n)begin
32         rst_check:assert final ( if_t.count_out==0 && if_t.zero==1 && if_t.max_count==0 );
33         reset_cover:cover (if_t.count_out==0 && if_t.zero==1 && if_t.max_count==0);
34     end
35 end
36
37
38 //concurrent assertion
39
40 load_assert:assert property(load_check) else $display("@%t reset=%0d", $time, if_t.rst_n);
41 turn_off_all_assert:assert property(turn_off_all) else $display("@%t reset=%0d", $time, if_t.rst_n);
42 count_up_assert:assert property(count_up) else $display("@%t reset=%0d", $time, if_t.rst_n);
43 count_down_assert:assert property(count_down) else $display("@%t reset=%0d", $time, if_t.rst_n);
44 maximum_assert:assert property(maximum) else $display("@%t reset=%0d", $time, if_t.rst_n);
45 minimum_assert:assert property(minimum) else $display("@%t reset=%0d", $time, if_t.rst_n);
46
47 //coverage assertion
48 load_cover:cover property(load_check);
49 turn_off_all_cover:cover property(turn_off_all);
50 count_up_cover:cover property(count_up);
51 count_down_cover:cover property(count_down);
52 maximum_cover:cover property(maximum);
53 minimum_cover:cover property(minimum);
54
55 endmodule
```

Package

```
1 package pack;
2 parameter WIDTH=4;
3 class transaction;
4
5 rand bit rst_n,load_n,up_down,ce;
6 rand bit [WIDTH-1:0] data_load;
7 bit [WIDTH-1:0] count_out;
8 bit zero,max_count,clk;
9
10 constraint x {
11     rst_n dist {0:=5,1:=100};
12     load_n dist {0:=70,1:=30};
13     ce dist {1:=70,0:=30};
14 }
15
16 covergroup covgup @(posedge clk) ;
17     coverpoint data_load iff(!load_n);
18     COUNT_UP:coverpoint count_out iff(rst_n && ce && up_down){
19         bins all_values_up[]={0:15};
20         bins trans_zero=(15=>0);
21     }
22     COUNT_DOWN:coverpoint count_out iff(rst_n && ce && !up_down){
23         bins all_values_down[]={0:15};
24         bins trans_max=(0=>15);
25     }
26
27 max_check :cross count_out,max_count{
28     option.cross_auto_bin_max=0;
29     bins max_detect=binsof(count_out) intersect{15} && binsof(max_count) intersect {1};
30     illegal_bins invalid=binsof(count_out) intersect{[0:14]} && binsof(max_count) intersect {1};
31 }
32
33 zeros_check :cross count_out,zero{
34     option.cross_auto_bin_max=0;
35     bins zero_detect=binsof(count_out) intersect{0} && binsof(zero) intersect {1};
36     illegal_bins invalid=binsof(count_out) intersect{[1:15]} && binsof(zero) intersect {1};
37 }
38
39 endgroup
40
41 function new();
42     covgup=new();
43 endfunction
44
45 endclass
46 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
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
```

```

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 Unit	Design UnitType	Lang	File(Line)	Count	Status
/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 Count	Pass 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 Count	Pass 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%

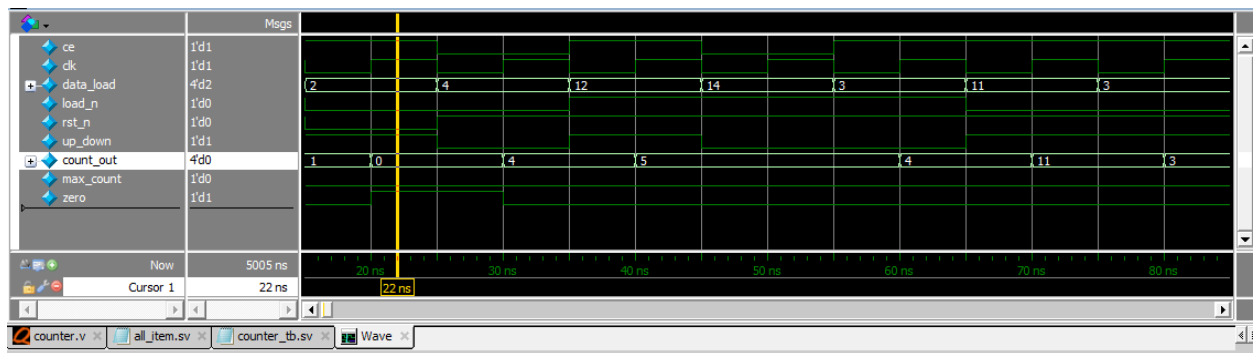
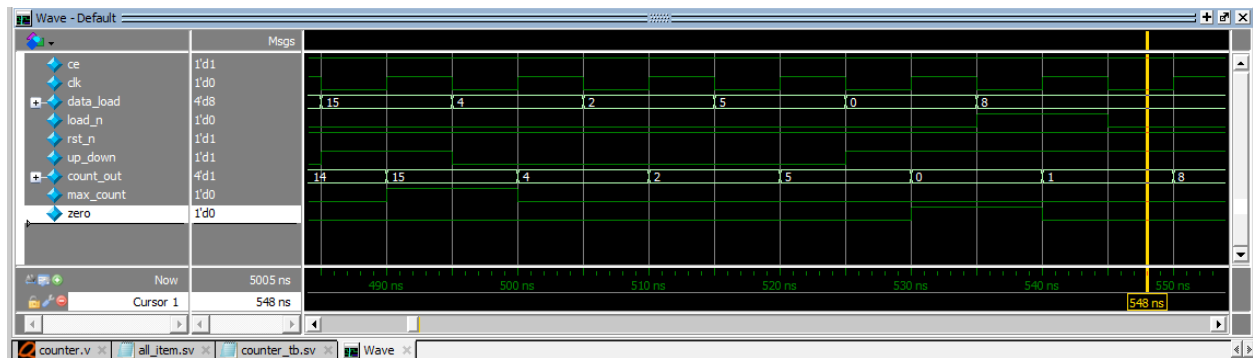
Functional coverage

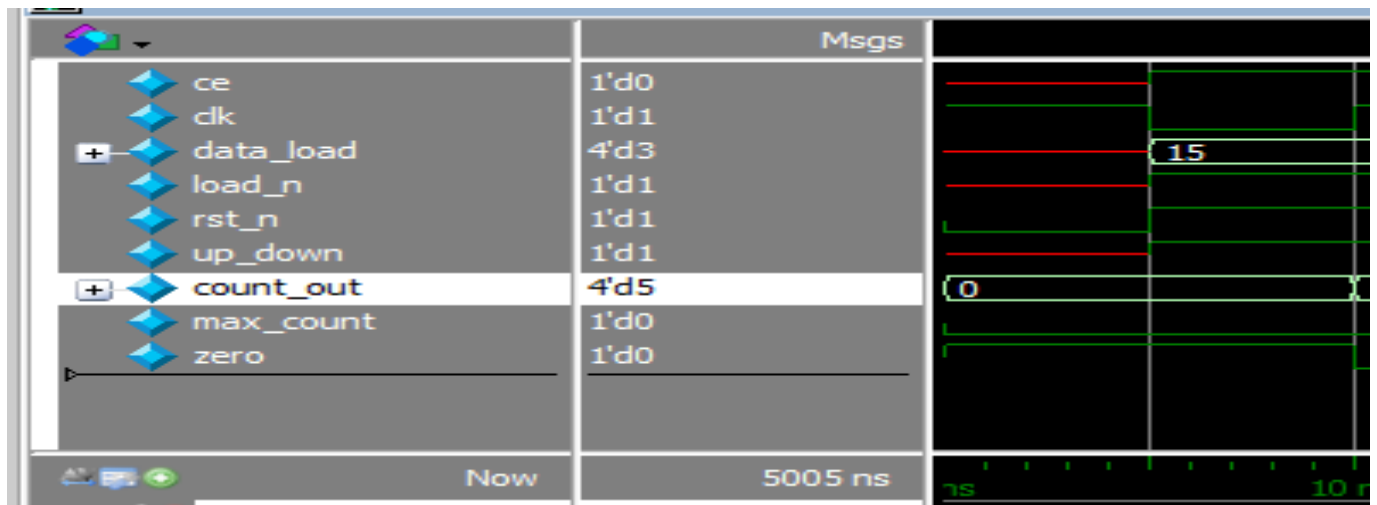
Toggle Coverage = 100.0% (2 of 2 bins)

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

1	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 concurrent 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 concurrent 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 concurrent assertion on out with data load	-
6	COUNTER_5	we assert load_n to high and we assert enable input to high and count up -> the output count_out should increase every clock cycle	Randomization we can check this in most of simulation	check with concurrent assertion on output increment	-
7	COUNTER_6	we assert load_n to high and we assert enable input to high and count down-> the output count_out should decrease every clock cycle	Randomization under constrain that ec should be High(active) for 70% of time,	check with concurrent 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 concurrent 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 concurrent 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 enable is low the output shouldnt change and keep previous value	Randomization through simmulation	check with immediate assertion for asyn reset	-