يوسف أحمد محمد ابراهيم

Assignment 4

- **Question 1:**

1. RTL 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, cin, rst, 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;

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;
      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
```

1. RTL design:

```
49   //leds output blinking
50   always @(posedge clk or posedge rst) begin
51     if(rst) begin
52        leds <= 0;
53     end else begin
54         if (invalid)
55           leds <= ~leds;
56         else
57           leds <= 0;
58     end
59   end
60
61   //ALSU output processing
62   always @(posedge clk or posedge rst) begin
63     if(rst) begin
64       out <= 0;
65     end
66     else begin
67       if (bypass_A_reg && bypass_B_reg)
68         out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
69       else if (bypass_A_reg)
70         out <= A_reg;
71       else if (bypass_B_reg)
72         out <= B_reg;
73       else if (invalid)
74           out <= 0;
75       else begin
76           case (opcode_reg)
77             3'h0: begin
78               if (red_op_A_reg && red_op_B_reg)
79                 out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
80               else if (red_op_A_reg)
81                 out <= |A_reg;
82               else if (red_op_B_reg)
83                 out <= |B_reg;
84               else
85                 out <= A_reg | B_reg;
86             end
87             3'h1: begin
88               if (red_op_A_reg && red_op_B_reg)
89                 out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
90               else if (red_op_A_reg)
91                 out <= ^A_reg;
92               else if (red_op_B_reg)
93                 out <= ^B_reg;
94               else
95                 out <= A_reg ^ B_reg;
96             end
97             3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
98             3'h3: out <= A_reg * B_reg;
99             3'h4: begin
```

1. RTL design:

```
100             if (direction_reg)
101               out <= {out[4:0], serial_in_reg};
102             else
103               out <= {serial_in_reg, out[5:1]};
104           end
105           3'h5: begin
106             if (direction_reg)
107               out <= {out[4:0], out[5]};
108             else
109               out <= {out[0], out[5:1]};
110           end
111           default : out <= 0;
112         endcase
113       end
114     end
115   end
116
117   endmodule
```

2. Testbench:

```
import ALSU_pkg::*;

module ALSU_tb();
    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    bit signed [2:0]  A;
    bit signed [2:0]  B;
    bit               cin;
    bit               serial_in;
    bit               red_op_A;
    bit               red_op_B;
    opcode_e          opcode;
    bit               bypass_A;
    bit               bypass_B;
    bit               clk;
    bit               rst;
    bit               direction;
    bit        [15:0] leds;
    bit 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         [2:0] opcode_reg;
    reg signed  [1:0] cin_reg;
    reg signed  [2:0] A_reg, B_reg;

    bit               invalid;

    rand_stimuls my_inputs; //handle

    bit [15:0]  leds_exp;
    bit [5:0]   out_exp;

    integer correct_count, error_count;

    ALSU dut(.*);

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


    initial begin
        correct_count = 0;
        error_count   = 0;
        A = 0;
        B = 0;
        cin = 0;
        serial_in = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;
        direction = 0;
```

2. Testbench:

```
57      //reset test
58       rst = 1;
59       check_result();
60
61       my_inputs = new(opcode);
62       //////random test ///////
63       //loop1
64       my_inputs.c2.constraint_mode(0); //disable
65       for(int i = 0; i<100; i++) begin
66           assert(my_inputs.randomize());
67
68           A = my_inputs.A;
69           rst = my_inputs.rst;
70           B = my_inputs.B;
71           cin = my_inputs.cin;
72           serial_in = my_inputs.serial_in;
73           red_op_A = my_inputs.red_op_A;
74           red_op_B = my_inputs.red_op_B;
75           opcode = my_inputs.opcode;
76           bypass_A = my_inputs.bypass_A;
77           bypass_B = my_inputs.bypass_B;
78           direction = my_inputs.direction;
79
80           check_result();
81       end
82
83       //loop2
84       my_inputs.constraint_mode(0); //disable
85       rst = 0;
86       bypass_A = 0;
87       bypass_B = 0;
88       red_op_A = 0;
89       red_op_B = 0;
90       my_inputs.c2.constraint_mode(1); //enable
91       for(int i = 0; i<10000; i++) begin
92           assert(my_inputs.randomize());
93
94           A = my_inputs.A;
95           B = my_inputs.B;
96           cin = my_inputs.cin;
97           serial_in = my_inputs.serial_in;
98           direction = my_inputs.direction;
99
100          foreach(my_inputs.opcode_array[j]) begin //this will loop 6 times
101              opcode = my_inputs.opcode_array[j];
102              check_result();
103          end
104      end
105
106      $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
107      $stop;
108  end
109
```

2. Testbench:

```
    //golden model
    assign invalid = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2])
        ||(opcode_reg[1] & opcode_reg[2]);

    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;
            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

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            leds_exp <= 0;
        end else begin
            if (invalid)
                leds_exp <= ~leds_exp;
            else
                leds_exp <= 0;
        end
    end

    always @(posedge clk or posedge rst) begin
        if(rst)
            out_exp <= 0;
        else begin
            if (bypass_A_reg && bypass_B_reg)
                out_exp <= A_reg;
            else if (bypass_A_reg)
                out_exp <= A_reg;
            else if (bypass_B_reg)
                out_exp <= B_reg;
```

2. Testbench:

```
160                 out_exp <= B_reg;
161             else if (invalid)
162                 out_exp <= 0;
163             else begin
164                 case (opcode_reg)
165                 3'h0: begin
166                     if (red_op_A_reg && red_op_B_reg)
167                         out_exp <= |A_reg;
168                     else if (red_op_A_reg)
169                         out_exp <= |A_reg;
170                     else if (red_op_B_reg)
171                         out_exp <= |B_reg;
172                     else
173                         out_exp <= A_reg | B_reg;
174                 end
175                 3'h1: begin
176                     if (red_op_A_reg && red_op_B_reg)
177                         out_exp <= ^A_reg;
178                     else if (red_op_A_reg)
179                         out_exp <= ^A_reg;
180                     else if (red_op_B_reg)
181                         out_exp <= ^B_reg;
182                     else
183                         out_exp <= A_reg ^ B_reg;
184                 end
185 
186                 3'h2: out_exp <= A_reg + B_reg + cin_reg;
187                 3'h3: out_exp <= A_reg * B_reg;
188                 3'h4: begin
189                     if (direction_reg)
190                         out_exp <= {out_exp[4:0], serial_in_reg};
191                     else
192                         out_exp <= {serial_in_reg, out_exp[5:1]};
193                 end
194                 3'h5: begin
195                     if (direction_reg)
196                         out_exp <= {out_exp[4:0], out_exp[5]};
197                     else
198                         out_exp <= {out_exp[0], out_exp[5:1]};
199                 end
200                 endcase
201             end
202         end
203     end
```

2. Testbench:

```
205     task check_result();
206         @(negedge clk);
207         @(negedge clk);
208         if(out != out_exp || leds != leds_exp) begin
209             $display("*** ERROR! at time %0t, out = %0d, Expected : %0d, leds = %0d, Expected : %0d ***"
210                 , $time, out, out_exp, leds, leds_exp);
211             error_count++;
212         end
213         else
214             correct_count++;
215 
216     endtask
217 
218     always @(posedge clk) @(posedge clk)
219         if (!rst && !(bypass_A || bypass_B))
220             my_inputs.cvr_gp.sample();
221 
222 endmodule
```

3. Package :

```
package ALSU_pkg;

    typedef enum bit[2:0] {
        OR,
        XOR,
        ADD,
        MULT,
        SHIFT,
        ROTATE,
        INVALID_6,
        INVALID_7
    } opcode_e;

    parameter MAXPOS = 3'b011;
    parameter ZERO = 3'b000;
    parameter MAXNEG = 3'b100;

    class rand_stimuls;
        rand bit  [2:0] A;
        rand bit  [2:0] B;
        rand bit        rst;
        rand bit        red_op_A;
        rand bit        red_op_B;
        rand bit        bypass_A;
        rand bit        bypass_B;
        rand bit        cin;
        rand bit        serial_in;
        rand bit        direction;
        rand opcode_e   opcode;
        randc opcode_e   opcode_array[6];

        //no need for constructor, they will be initialized to 0

        constraint c1 {
            rst dist {0 := 9, 1 := 1};


            opcode dist {[OR:ROTATE] := 5, [INVALID_6:INVALID_7] := 1};

            bypass_A dist {1 := 3, 0 := 7};
            bypass_B dist {1 := 3, 0 := 7};

            if(opcode == OR || opcode == XOR) {
                if(red_op_A) { //priority for A so red_op_B here doesn't matter
                    A dist {
                        [3'b000:3'b111] := 1,
                        3'b001 := 2,
                        3'b010 := 2,
                        3'b100 := 2
                        };
                    B == 3'b000;
                }
```

3. Package :

```
                else if(red_op_B){
                    A == 3'b000;
                    B dist {
                        [3'b000:3'b111] := 1,
                        3'b001 := 2,
                        3'b010 := 2,
                        3'b100 := 2
                        };
                }
            }

            else {
                red_op_A dist {0 := 7, 1 := 3};
                red_op_B dist {0 := 7, 1 := 3};
            }

            if(opcode == ADD || opcode == MULT) {
                A dist {[3'b001:3'b110] := 1, MAXPOS := 2, ZERO := 2, MAXNEG := 2};
                B dist {[3'b001:3'b110] := 1, MAXPOS := 2, ZERO := 2, MAXNEG := 2};
            }
        }

    constraint c2 {
        foreach (opcode_array[i])
            opcode_array[i] inside {SHIFT, ROTATE, ADD, MULT, OR, XOR};
    }


covergroup cvr_gp(ref opcode_e opcode_tb);

  A_cp : coverpoint A {
    option.comment = "If only the red_op_A is high";
    bins A_data_0        = {0};
    bins A_data_max      = {MAXPOS};
    bins A_data_min      = {MAXNEG};
    bins A_data_default  = default;
    bins A_data_walkingones[] = {3'b001, 3'b010, 3'b100}
      iff (red_op_A);

  }

  B_cp : coverpoint B {
    option.comment = "If only red_op_B is high and red_op_A is low";
    bins B_data_0        = {0};
    bins B_data_max      = {MAXPOS};
    bins B_data_min      = {MAXNEG};
    bins B_data_default  = default;
    bins B_data_walkingones[] = {3'b001, 3'b010, 3'b100}
      iff (red_op_B && !red_op_A);
  }
```

3. Package :

```
105     ALU_cp : coverpoint opcode_tb {
106         bins Bins_shift[]   = {SHIFT, ROTATE};
107         bins Bins_arith[]   = {ADD, MULT};
108         bins Bins_bitwise[] = {OR, XOR};
109         illegal_bins Bins_invalid   = {6, 7};
110         bins Bins_trans     = (OR => XOR => ADD => MULT => SHIFT => ROTATE);
111     }
112     op_arth : coverpoint opcode_tb {
113         option.weight = 0;
114         bins ADD_b = {ADD};
115         bins MULT_b = {MULT};
116         bins shift = {SHIFT};
117     }
118     c_B : coverpoint B {
119         option.weight = 0;
120         bins B_0        = {0};
121         bins B_max      = {MAXPOS};
122         bins B_min      = {MAXNEG};
123         bins walkingones1 = {3'b001};
124         bins walkingones2 = {3'b010};
125         bins walkingones3 = {3'b100};
126 
127     }
128     c_A : coverpoint A {
129         option.weight = 0;
130         bins A_0        = {0};
131         bins A_max      = {MAXPOS};
132         bins A_min      = {MAXNEG};
133         bins walkingones1 = {3'b001};
134         bins walkingones2 = {3'b010};
135         bins walkingones3 = {3'b100};
136     }
137 
138     red_A: coverpoint red_op_A;
139     red_B: coverpoint red_op_B;
140     op: coverpoint opcode_tb;
141 
142 
143     c1: cross op_arth, c_B, c_A{
144         ignore_bins b1 = binsof(op_arth) intersect {SHIFT};
145         ignore_bins b2 = binsof(c_A.walkingones1);
146         ignore_bins b3 = binsof(c_A.walkingones2);
147         ignore_bins b4 = binsof(c_A.walkingones3);
148         ignore_bins b5 = binsof(c_B.walkingones1);
149         ignore_bins b6 = binsof(c_B.walkingones2);
150         ignore_bins b7 = binsof(c_B.walkingones3);
151     }
152 
153     c2: cross op_arth, cin {
154         ignore_bins b1 = binsof(op_arth) intersect {MULT, SHIFT};
155 
```

3. Package :

```
156     }
157     c3: cross op_arth, direction {
158         ignore_bins b1 = binsof(op_arth) intersect {MULT, SHIFT};
159
160     }
161
162     c4: cross op_arth, serial_in {
163         ignore_bins b1 = binsof(op_arth) intersect {MULT, SHIFT};
164
165     }
166
167     c5: cross ALU_cp, red_A, c_A, c_B{
168         option.cross_auto_bin_max = 0;
169         bins b1 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
170                             binsof(red_A) intersect {1} &&
171                             binsof(c_B) intersect {0} &&
172                             binsof(c_A.walkingones1);
173         bins b2 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
174                             binsof(red_A) intersect {1} &&
175                             binsof(c_B) intersect {0} &&
176                             binsof(c_A.walkingones2);
177         bins b3 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
178                             binsof(red_A) intersect {1} &&
179                             binsof(c_B) intersect {0} &&
180                             binsof(c_A.walkingones3);
181         bins b4 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
182                             binsof(red_A) intersect {1} &&
183                             binsof(c_B) intersect {0} &&
184                             binsof(c_A.walkingones1);
185         bins b5 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
186                             binsof(red_A) intersect {1} &&
187                             binsof(c_B) intersect {0} &&
188                             binsof(c_A.walkingones2);
189         bins b6 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
190                             binsof(red_A) intersect {1} &&
191                             binsof(c_B) intersect {0} &&
192                             binsof(c_A.walkingones3);
193     }
194
195
196     c6: cross ALU_cp, red_B, c_A, c_B{
197         option.cross_auto_bin_max = 0;
198
199         bins b1 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
200                             binsof(red_B) intersect {1} &&
201                             binsof(c_A) intersect {0} &&
202                             binsof(c_B.walkingones1);
203
204         bins b2 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
205                             binsof(red_B) intersect {1} &&
206                             binsof(c_A) intersect {0} &&
207                             binsof(c_B.walkingones2);
```

3. Package :

```
208
209         bins b3 = binsof(ALU_cp.Bins_bitwise) intersect {OR} &&
210                           binsof(red_B) intersect {1} &&
211                           binsof(c_A) intersect {0} &&
212                           binsof(c_B.walkingones3);
213
214         bins b4 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
215                           binsof(red_B) intersect {1} &&
216                           binsof(c_A) intersect {0} &&
217                           binsof(c_B.walkingones1);
218
219         bins b5 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
220                           binsof(red_B) intersect {1} &&
221                           binsof(c_A) intersect {0} &&
222                           binsof(c_B.walkingones2);
223
224         bins b6 = binsof(ALU_cp.Bins_bitwise) intersect {XOR} &&
225                           binsof(red_B) intersect {1} &&
226                           binsof(c_A) intersect {0} &&
227                           binsof(c_B.walkingones3);
228     }
229
230
231     c7: cross red_A, red_B, op {
232         ignore_bins b1 = binsof(red_A) intersect {0};
233         ignore_bins b2 = binsof(red_B) intersect {0};
234         ignore_bins b3 = binsof(op) intersect {OR, XOR};
235
236     }
237
238     endgroup
239
240         function new (ref opcode_e opcode_tb);
241             cvr_gp = new(opcode_tb);
242         endfunction
243     endclass
244 endpackage
```

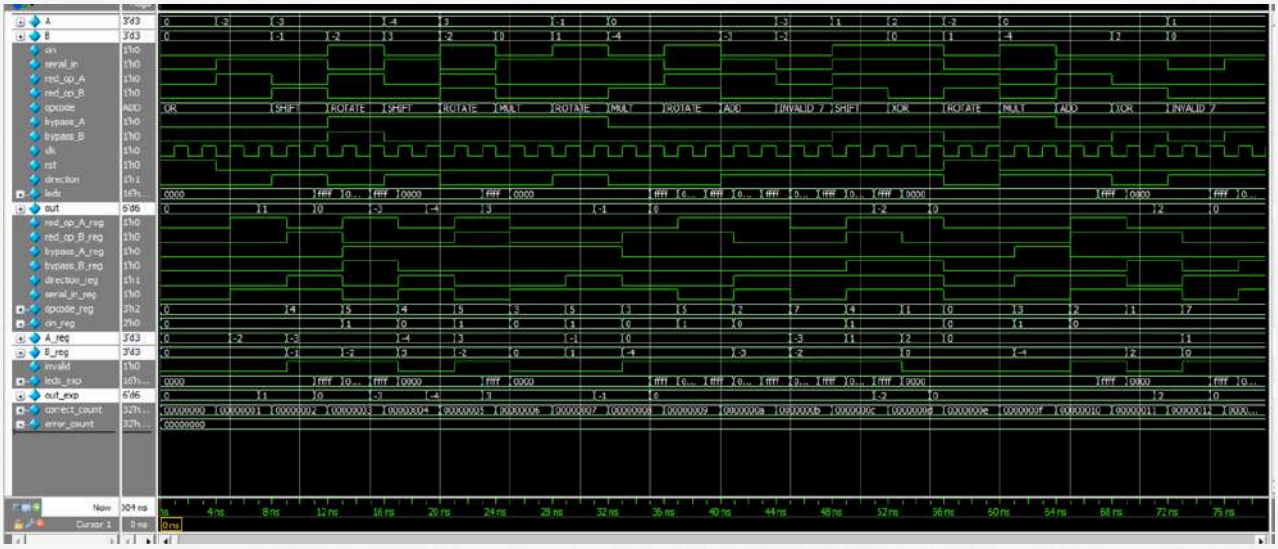
4. Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET_TEST	When the reset is asserted, outputs should be low.	Directed at the start of the simulation, and called in other radomization steps.	-	**check_result()** verifies correct output using a golden model().
RANDOMIZATION_TEST	Output of the dut design should equal to the output of the golden model with the rest of input randomized.	ALL inputs are randomized using the calss with the following constraints: **rst** is 90% low, any **OPCODE** have invalid value to valid one is 1:3, and **A/B** have one-hot value to any other value is 2:1 when **red_op_A/red_cp_B** is high and **OPCODE = OR/XOR.** And **A/B** have {MAXPOS, ZERO, MAXNG} value to any other value = 2 : 1 when **OPCODE** is MULT or ADD. **bypass_A/bypass_B** are 30% high. **red_op_A/red_op_B** are 30% low when **OPCODE** is not OR, XOR. an array of opcode_e type with 6 valid opcode sequences is constrainted to have unique values each time.	The covergroup includes the following coverpoints. A_cp monitors operand A with bins for value 0, MAXPOS, and MAXNEG, a default bin, and three walking-1 bins (001, 010, 100) sampled only when red_op_A is high. B_cp monitors operand B with bins for 0, MAXPOS, and MAXNEG, a default bin, and three walking-1 bins sampled only when red_op_B is high and red_op_A is low. ALU_cp monitors the opcode with bins grouped into shift operations (SHIFT and ROTATE), arithmetic operations (ADD and MULT), bitwise operations (OR and XOR), illegal bins for INVALID_6 and INVALID_7, and a transition bin that checks the ordered sequence OR -> XOR -> ADD -> MULT -> SHIFT -> ROTATE. Additional simple coverpoints include red_A (for red_op_A), red_B (for red_op_B), and cp (for the opcode value). The cross coverage is defined as follows. c5 checks ALU_cp crossed with red_A, A_cp, and B_cp. It creates explicit bins to verify that when OR or XOR operations are selected, and red_op_A is high while red_op_B is low, operand A must take each of the walking-1 values while operand B is0. c6 is similar, but for ALU_cp crossed with red_B, A_cp, and B_cp. It verifies that for OR or XOR operations, when red_op_B is high and red_op_A is low, operand B takes each walking-1 value while operand A is 0. Finally, c7 checks the cross of red_A, red_B, and the opcode. It ignores combinations where red_A or red_B are 0, and ignores opcode values OR and XOR, ensuring only valid interactions between the two reduction signals and the opcode are covered.	**check_result()** verifies correct output using a golden model().

5.Do file:

```
vlib work
vlog 3_pkg.sv 3_ALSU.v 3_ALSU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save 3_ALSU_tb.ucdb -onexit
coverage exclude -du ALSU -togglenode {cin_reg[1]}
coverage exclude -src 3_ALSU.v -line 111 -code b
coverage exclude -src 3_ALSU.v -line 111 -code s

run -all
```

6. Wave snippets :



```
#    Time: 39267 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 301.
#    Time: 39275 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_7. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 302.
#    Time: 39331 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_7. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 303.
#    Time: 39439 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 304.
#    Time: 39483 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 305.
#    Time: 39531 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 306.
#    Time: 39567 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 307.
#    Time: 39599 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 308.
#    Time: 39751 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_7. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 309.
#    Time: 39803 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 310.
#    Time: 39855 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_7. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 311.
#    Time: 39887 ns  Iteration: 1  Instance: /ALSU_tb
# *** ERROR count: 0, CORRECT count: 70001
# ** Note: $stop    : 3_ALSU_tb.sv(107)
#    Time: 280004 ns  Iteration: 1  Instance: /ALSU_tb
# Break in Module ALSU_tb at 3_ALSU_tb.sv line 107
```

7. Statement coverage :

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     48        48         0   100.00%

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

Statement Coverage for instance /ALSU_tb/dut --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File 3_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, cin, rst, 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;
    14
    15                                              wire invalid_red_op, invalid_opcode, invalid;
    16
    17                                              //Invalid handling
    18              1                       59371   assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
    19              1                       58457   assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
    20              1                        4762   assign invalid = invalid_red_op | invalid_opcode;
    21
```

7. Statement coverage :

```
22                                      //Registering input signals
23              1               123358  always @(posedge clk or posedge rst) begin
24                                        if(rst) begin
25              1                 1998      cin_reg <= 0;
26              1                 1998      red_op_B_reg <= 0;
27              1                 1998      red_op_A_reg <= 0;
28              1                 1998      bypass_B_reg <= 0;
29              1                 1998      bypass_A_reg <= 0;
30              1                 1998      direction_reg <= 0;
31              1                 1998      serial_in_reg <= 0;
32              1                 1998      opcode_reg <= 0;
33              1                 1998      A_reg <= 0;
34              1                 1998      B_reg <= 0;
35                                        end else begin
36              1               121360      cin_reg <= cin;
37              1               121360      red_op_B_reg <= red_op_B;
38              1               121360      red_op_A_reg <= red_op_A;
39              1               121360      bypass_B_reg <= bypass_B;
40              1               121360      bypass_A_reg <= bypass_A;
41              1               121360      direction_reg <= direction;
42              1               121360      serial_in_reg <= serial_in;
43              1               121360      opcode_reg <= opcode;
44              1               121360      A_reg <= A;
45              1               121360      B_reg <= B;
46                                        end
47                                      end
```

7. Statement coverage :

```
48
49                                      //leds output blinking
50          1                  140951   always @(posedge clk or posedge rst) begin
51                                        if(rst) begin
52          1                    3047           leds <= 0;
53                                        end else begin
54                                            if (invalid)
55          1                    6323             leds <= ~leds;
56                                            else
57          1                  131581             leds <= 0;
58                                        end
59                                      end
60
61                                      //ALSU output processing
62          1                  116777   always @(posedge clk or posedge rst) begin
63                                        if(rst) begin
64          1                    1898          out <= 0;
65                                        end
66                                        else begin
67                                          if (bypass_A_reg && bypass_B_reg)
68          1                    1427           out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
69                                          else if (bypass_A_reg)
70          1                    3385           out <= A_reg;
71                                          else if (bypass_B_reg)
72          1                    3353           out <= B_reg;
73                                          else if (invalid)
74          1                    2499             out <= 0;
```

7. Statement coverage :

```
75                                      else begin
76                                          case (opcode_reg)
77                                            3'h0: begin
78                                              if (red_op_A_reg && red_op_B_reg)
79          1                    200                out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
80                                              else if (red_op_A_reg)
81          1                    126                out <= |A_reg;
82                                              else if (red_op_B_reg)
83          1                    117                out <= |B_reg;
84                                              else
85          1                  17305                out <= A_reg | B_reg;
86                                            end
87                                            3'h1: begin
88                                              if (red_op_A_reg && red_op_B_reg)
89          1                    204                out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
90                                              else if (red_op_A_reg)
91          1                    103                out <= ^A_reg;
92                                              else if (red_op_B_reg)
93          1                    144                out <= ^B_reg;
94                                              else
95          1                  16688                out <= A_reg ^ B_reg;
96                                            end
97          1                  16712            3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
98          1                  17229            3'h3: out <= A_reg * B_reg;
99                                            3'h4: begin
100                                             if (direction_reg)
101         1                   9296                out <= {out[4:0], serial_in_reg};
102                                             else
103         1                   9128                out <= {serial_in_reg, out[5:1]};
104                                           end
105                                           3'h5: begin
106                                             if (direction_reg)
107         1                   8390                out <= {out[4:0], out[5]};
108                                             else
109         1                   8573                out <= {out[0], out[5:1]};
```

8. Branch coverage :

```
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        31        31         0   100.00%

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

Branch Coverage for instance /ALSU_tb/dut

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File 3_ALSU.v
------------------------------------IF Branch------------------------------------
    24                                   123358     Count coming in to IF
    24             1                       1998       if(rst) begin

    35             1                     121360       end else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    51                                   140951     Count coming in to IF
    51             1                       3047       if(rst) begin

    53             1                     137904       end else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    54                                   137904     Count coming in to IF
    54             1                       6323           if (invalid)

    56             1                     131581           else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                   116777     Count coming in to IF
    63             1                       1898       if(rst) begin

    66             1                     114879       else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                   114879     Count coming in to IF
    67             1                       1427         if (bypass_A_reg && bypass_B_reg)

    69             1                       3385         else if (bypass_A_reg)

    71             1                       3353         else if (bypass_B_reg)

    73             1                       2499         else if (invalid)

    75             1                     104215         else begin

Branch totals: 5 hits of 5 branches = 100.00%

-----------------------------------CASE Branch-----------------------------------
    76                                   104215     Count coming in to CASE
    77             1                      17748              3'h0: begin

    87             1                      17139              3'h1: begin

    97             1                      16712              3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
```

8. Branch coverage :

```
                                    CASE Branch
    97              1                   16712               3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;

    98              1                   17229               3'h3: out <= A_reg * B_reg;

    99              1                   18424               3'h4: begin

    105             1                   16963               3'h5: begin

Branch totals: 6 hits of 6 branches = 100.00%

----------------------------------IF Branch----------------------------------
    78                                  17748     Count coming in to IF
    78              1                     200                 if (red_op_A_reg && red_op_B_reg)

    80              1                     126                 else if (red_op_A_reg)

    82              1                     117                 else if (red_op_B_reg)

    84              1                   17305                 else

Branch totals: 4 hits of 4 branches = 100.00%

----------------------------------IF Branch----------------------------------
    88                                  17139     Count coming in to IF
    88              1                     204                 if (red_op_A_reg && red_op_B_reg)

    90              1                     103                 else if (red_op_A_reg)

    92              1                     144                 else if (red_op_B_reg)

    94              1                   16688                 else

Branch totals: 4 hits of 4 branches = 100.00%

----------------------------------IF Branch----------------------------------
    100                                 18424     Count coming in to IF
    100             1                    9296                 if (direction_reg)

    102             1                    9128                 else

Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------IF Branch----------------------------------
    106                                 16963     Count coming in to IF
    106             1                    8390                 if (direction_reg)

    108             1                    8573                 else

Branch totals: 2 hits of 2 branches = 100.00%
```

9. Toggle coverage:

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        118       118         0   100.00%

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

Toggle Coverage for instance /ALSU_tb/dut --

                                    Node      1H->0L      0L->1H                          "Coverage"
                                    ----------------------------------------------------------------
                                   A[0-2]           1           1                             100.00
                                A_reg[2-0]           1           1                             100.00
                                   B[0-2]           1           1                             100.00
                                B_reg[2-0]           1           1                             100.00
                                  bypass_A           1           1                             100.00
                               bypass_A_reg          1           1                             100.00
                                  bypass_B           1           1                             100.00
                               bypass_B_reg          1           1                             100.00
                                       cin           1           1                             100.00
                                 cin_reg[0]          1           1                             100.00
                                       clk           1           1                             100.00
                                 direction           1           1                             100.00
                              direction_reg          1           1                             100.00
                                   invalid           1           1                             100.00
                             invalid_opcode          1           1                             100.00
                              invalid_red_op         1           1                             100.00
                                 leds[15-0]          1           1                             100.00
                                opcode[0-2]          1           1                             100.00
                             opcode_reg[2-0]         1           1                             100.00
                                  out[5-0]           1           1                             100.00
                                  red_op_A           1           1                             100.00
                               red_op_A_reg          1           1                             100.00
                                  red_op_B           1           1                             100.00
                               red_op_B_reg          1           1                             100.00
                                       rst           1           1                             100.00
                                 serial_in           1           1                             100.00
                              serial_in_reg          1           1                             100.00

  Total Node Count     =         59
  Toggled Node Count   =         59
  Untoggled Node Count =          0

  Toggle Coverage      =     100.00% (118 of 118 bins)
```

10. Functional coverage:

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

----------------------------------------------------------------------------------------------------
 TYPE /ALSU_pkg/rand_stimuls/cvr_gp                100.00%        100          -    Covered
    covered/total bins:                                 94         94          -
    missing/total bins:                                  0         94          -
    % Hit:                                         100.00%        100          -
    Coverpoint A_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint B_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint ALU_cp                              100.00%        100          -    Covered
        covered/total bins:                              7          7          -
        missing/total bins:                              0          7          -
        % Hit:                                     100.00%        100          -
    Coverpoint op_arth                               0.00%        100          -    ZERO
        covered/total bins:                              3          3          -
        missing/total bins:                              0          3          -
        % Hit:                                     100.00%        100          -
    Coverpoint c_B                                   0.00%        100          -    ZERO
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint c_A                                   0.00%        100          -    ZERO
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint red_A                               100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint red_B                               100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint op                                  100.00%        100          -    Covered
        covered/total bins:                              8          8          -
        missing/total bins:                              0          8          -
        % Hit:                                     100.00%        100          -
    Coverpoint serial_in                           100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint direction                           100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint cin                                 100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Cross c1                                       100.00%        100          -    Covered
        covered/total bins:                             18         18          -
        missing/total bins:                              0         18          -
        % Hit:                                     100.00%        100          -
    Cross c2                                       100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Cross c3                                       100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Cross c4                                       100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
```

10. Functional coverage:

2172	Cross c5	100.00%	100	-	Covered
2173	covered/total bins:	6	6	-	
2174	missing/total bins:	0	6	-	
2175	% Hit:	100.00%	100	-	
2176	Cross c6	100.00%	100	-	Covered
2177	covered/total bins:	6	6	-	
2178	missing/total bins:	0	6	-	
2179	% Hit:	100.00%	100	-	
2180	Cross c7	100.00%	100	-	Covered
2181	covered/total bins:	6	6	-	
2182	missing/total bins:	0	6	-	
2183	% Hit:	100.00%	100	-	
2184	Covergroup instance \/ALSU_pkg::rand_stimuls::cvr_gp				
2185		100.00%	100	-	Covered
2186	covered/total bins:	94	94	-	
2187	missing/total bins:	0	94	-	
2188	% Hit:	100.00%	100	-	
2189	Coverpoint A_cp	100.00%	100	-	Covered
2190	covered/total bins:	6	6	-	
2191	missing/total bins:	0	6	-	
2192	% Hit:	100.00%	100	-	
2193	bin A_data_0	8567	1	-	Covered
2194	bin A_data_max	8253	1	-	Covered
2195	bin A_data_min	8320	1	-	Covered
2196	bin A_data_walkingones[1]	3674	1	-	Covered
2197	bin A_data_walkingones[2]	3834	1	-	Covered
2198	bin A_data_walkingones[4]	3897	1	-	Covered
2199	default bin A_data_default	23602		-	Occurred
2200	Coverpoint B_cp	100.00%	100	-	Covered
2201	covered/total bins:	6	6	-	
2202	missing/total bins:	0	6	-	
2203	% Hit:	100.00%	100	-	
2204	bin B_data_0	8654	1	-	Covered
2205	bin B_data_max	8106	1	-	Covered
2206	bin B_data_min	8081	1	-	Covered
2207	bin B_data_walkingones[1]	1855	1	-	Covered
2208	bin B_data_walkingones[2]	1927	1	-	Covered
2209	bin B_data_walkingones[4]	2186	1	-	Covered
2210	default bin B_data_default	23902		-	Occurred
2211	Coverpoint ALU_cp	100.00%	100	-	Covered
2212	covered/total bins:	7	7	-	
2213	missing/total bins:	0	7	-	
2214	% Hit:	100.00%	100	-	
2215	illegal_bin Bins_invalid	311		-	Occurred
2216	bin Bins_shift[SHIFT]	10680	1	-	Covered
2217	bin Bins_shift[ROTATE]	10616	1	-	Covered
2218	bin Bins_arith[ADD]	10667	1	-	Covered
2219	bin Bins_arith[MULT]	10659	1	-	Covered
2220	bin Bins_bitwise[OR]	10675	1	-	Covered
2221	bin Bins_bitwise[XOR]	10739	1	-	Covered
2222	bin Bins_trans	1	1	-	Covered
2223	Coverpoint op_arth [1]	100.00%	100	-	Covered
2224	covered/total bins:	3	3	-	
2225	missing/total bins:	0	3	-	
2226	% Hit:	100.00%	100	-	
2227	bin ADD_b	10667	1	-	Covered
2228	bin MULT_b	10659	1	-	Covered
2229	bin shift	10680	1	-	Covered
2230	Coverpoint c_B [1]	100.00%	100	-	Covered
2231	covered/total bins:	6	6	-	
2232	missing/total bins:	0	6	-	
2233	% Hit:	100.00%	100	-	
2234	bin B_0	8654	1	-	Covered
2235	bin B_max	8106	1	-	Covered
2236	bin B_min	8081	1	-	Covered
2237	bin walkingones1	7671	1	-	Covered
2238	bin walkingones2	7933	1	-	Covered
2239	bin walkingones3	8081	1	-	Covered
2240	Coverpoint c_A [1]	100.00%	100	-	Covered
2241	covered/total bins:	6	6	-	
2242	missing/total bins:	0	6	-	
2243	% Hit:	100.00%	100	-	
2244	bin A_0	8567	1	-	Covered
2245	bin A_max	8253	1	-	Covered
2246	bin A_min	8320	1	-	Covered
2247	bin walkingones1	7798	1	-	Covered

10. Functional coverage:

2247	bin walkingones1	7798	1	-	Covered
2248	bin walkingones2	7807	1	-	Covered
2249	bin walkingones3	8320	1	-	Covered
2250	Coverpoint red_A	100.00%	100	-	Covered
2251	covered/total bins:	2	2	-	
2252	missing/total bins:	0	2	-	
2253	% Hit:	100.00%	100	-	
2254	bin auto[0]	33692	1	-	Covered
2255	bin auto[1]	30655	1	-	Covered
2256	Coverpoint red_B	100.00%	100	-	Covered
2257	covered/total bins:	2	2	-	
2258	missing/total bins:	0	2	-	
2259	% Hit:	100.00%	100	-	
2260	bin auto[0]	33094	1	-	Covered
2261	bin auto[1]	31253	1	-	Covered
2262	Coverpoint op	100.00%	100	-	Covered
2263	covered/total bins:	8	8	-	
2264	missing/total bins:	0	8	-	
2265	% Hit:	100.00%	100	-	
2266	bin auto[OR]	10675	1	-	Covered
2267	bin auto[XOR]	10739	1	-	Covered
2268	bin auto[ADD]	10667	1	-	Covered
2269	bin auto[MULT]	10659	1	-	Covered
2270	bin auto[SHIFT]	10680	1	-	Covered
2271	bin auto[ROTATE]	10616	1	-	Covered
2272	bin auto[INVALID_6]	149	1	-	Covered
2273	bin auto[INVALID_7]	162	1	-	Covered
2274	Coverpoint serial_in	100.00%	100	-	Covered
2275	covered/total bins:	2	2	-	
2276	missing/total bins:	0	2	-	
2277	% Hit:	100.00%	100	-	
2278	bin auto[0]	32332	1	-	Covered
2279	bin auto[1]	32015	1	-	Covered
2280	Coverpoint direction	100.00%	100	-	Covered
2281	covered/total bins:	2	2	-	
2282	missing/total bins:	0	2	-	
2283	% Hit:	100.00%	100	-	
2284	bin auto[0]	32469	1	-	Covered
2285	bin auto[1]	31878	1	-	Covered
2286	Coverpoint cin	100.00%	100	-	Covered
2287	covered/total bins:	2	2	-	
2288	missing/total bins:	0	2	-	
2289	% Hit:	100.00%	100	-	
2290	bin auto[0]	32578	1	-	Covered
2291	bin auto[1]	31769	1	-	Covered
2292	Cross c1	100.00%	100	-	Covered
2293	covered/total bins:	18	18	-	
2294	missing/total bins:	0	18	-	
2295	% Hit:	100.00%	100	-	
2296	Auto, Default and User Defined Bins:				
2297	bin <MULT_b,B_min,A_min>	215	1	-	Covered
2298	bin <ADD_b,B_min,A_min>	200	1	-	Covered
2299	bin <MULT_b,B_max,A_min>	186	1	-	Covered
2300	bin <ADD_b,B_max,A_min>	237	1	-	Covered
2301	bin <MULT_b,B_0,A_min>	180	1	-	Covered
2302	bin <ADD_b,B_0,A_min>	182	1	-	Covered
2303	bin <MULT_b,B_min,A_max>	171	1	-	Covered
2304	bin <ADD_b,B_min,A_max>	192	1	-	Covered
2305	bin <MULT_b,B_min,A_0>	188	1	-	Covered
2306	bin <ADD_b,B_min,A_0>	215	1	-	Covered
2307	bin <MULT_b,B_max,A_max>	194	1	-	Covered
2308	bin <ADD_b,B_max,A_max>	225	1	-	Covered
2309	bin <MULT_b,B_max,A_0>	181	1	-	Covered
2310	bin <ADD_b,B_max,A_0>	183	1	-	Covered
2311	bin <MULT_b,B_0,A_max>	200	1	-	Covered
2312	bin <ADD_b,B_0,A_max>	176	1	-	Covered
2313	bin <MULT_b,B_0,A_0>	160	1	-	Covered
2314	bin <ADD_b,B_0,A_0>	204	1	-	Covered
2315	Illegal and Ignore Bins:				
2316	ignore_bin b7	2708		-	Occurred
2317	ignore_bin b6	2488		-	Occurred
2318	ignore_bin b5	2304		-	Occurred
2319	ignore_bin b4	2690		-	Occurred
2320	ignore_bin b3	2425		-	Occurred
2321	ignore_bin b2	2373		-	Occurred
2322	ignore_bin b1	4164		-	Occurred

10. Functional coverage:

```
2323    Cross c2                                          100.00%        100          -    Covered
2324        covered/total bins:                                 2          2          -
2325        missing/total bins:                                 0          2          -
2326        % Hit:                                        100.00%        100          -
2327        Auto, Default and User Defined Bins:
2328            bin <ADD_b,auto[1]>                          5266          1          -    Covered
2329            bin <ADD_b,auto[0]>                          5401          1          -    Covered
2330        Illegal and Ignore Bins:
2331            ignore_bin b1                               21339                     -    Occurred
2332    Cross c3                                          100.00%        100          -    Covered
2333        covered/total bins:                                 2          2          -
2334        missing/total bins:                                 0          2          -
2335        % Hit:                                        100.00%        100          -
2336        Auto, Default and User Defined Bins:
2337            bin <ADD_b,auto[1]>                          5291          1          -    Covered
2338            bin <ADD_b,auto[0]>                          5376          1          -    Covered
2339        Illegal and Ignore Bins:
2340            ignore_bin b1                               21339                     -    Occurred
2341    Cross c4                                          100.00%        100          -    Covered
2342        covered/total bins:                                 2          2          -
2343        missing/total bins:                                 0          2          -
2344        % Hit:                                        100.00%        100          -
2345        Auto, Default and User Defined Bins:
2346            bin <ADD_b,auto[1]>                          5263          1          -    Covered
2347            bin <ADD_b,auto[0]>                          5404          1          -    Covered
2348        Illegal and Ignore Bins:
2349            ignore_bin b1                               21339                     -    Occurred
2350    Cross c5                                          100.00%        100          -    Covered
2351        covered/total bins:                                 6          6          -
2352        missing/total bins:                                 0          6          -
2353        % Hit:                                        100.00%        100          -
2354        Auto, Default and User Defined Bins:
2355            bin b1                                       138          1          -    Covered
2356            bin b2                                       104          1          -    Covered
2357            bin b3                                       111          1          -    Covered
2358            bin b4                                       140          1          -    Covered
2359            bin b5                                        99          1          -    Covered
2360            bin b6                                       107          1          -    Covered
2361    Cross c6                                          100.00%        100          -    Covered
2362        covered/total bins:                                 6          6          -
2363        missing/total bins:                                 0          6          -
2364        % Hit:                                        100.00%        100          -
2365        Auto, Default and User Defined Bins:
2366            bin b1                                        83          1          -    Covered
2367            bin b2                                        99          1          -    Covered
2368            bin b3                                        79          1          -    Covered
2369            bin b4                                        92          1          -    Covered
2370            bin b5                                        93          1          -    Covered
2371            bin b6                                        98          1          -    Covered
2372    Cross c7                                          100.00%        100          -    Covered
2373        covered/total bins:                                 6          6          -
2374        missing/total bins:                                 0          6          -
2375        % Hit:                                        100.00%        100          -
2376        Auto, Default and User Defined Bins:
2377            bin <auto[1],auto[1],auto[INVALID_7]>           13          1          -    Covered
2378            bin <auto[1],auto[1],auto[ROTATE]>            2561          1          -    Covered
2379            bin <auto[1],auto[1],auto[INVALID_6]>           10          1          -    Covered
2380            bin <auto[1],auto[1],auto[SHIFT]>             2508          1          -    Covered
2381            bin <auto[1],auto[1],auto[MULT]>              2531          1          -    Covered
2382            bin <auto[1],auto[1],auto[ADD]>               2570          1          -    Covered
2383        Illegal and Ignore Bins:
2384            ignore_bin b3                               21414                     -    Occurred
2385            ignore_bin b2                               33094                     -    Occurred
2386            ignore_bin b1                               33692                     -    Occurred
2387
2388    [1] - Does not contribute coverage as weight is 0
2389
2390    TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

Question 2:

1. Code:

```
module asser_ex();
    p1: assert property @(posedge clk)
        a |-> ##2 b;
    endproperty

    p2: assert property @(posedge clk)
        a && b |-> ##[1:3] c;
    endproperty

    sequence s11b;
        ##2 !b;
    endsequence


    property p4;
        @(posedge clk)
        $onehot(Y);
    endproperty

    property p5;
        @(posedge clk)
        !D |-> ##1 !valid;
    endproperty
endmodule
```

Question 3:

1. RTL code:

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

2. Verification plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
COUNTER_1	When the reset is asserted, the output counter value should be low	Directed at the start of the simulation,then randomized with constraint to make the rest off	-	Assertion combinational to check asyncronos rst
COUNTER_2	When the load is asserted, the output count_out should take the value of the load_data input	Randomize wih constrain to make the load active 70% of time	cover all values of laod	Assertion to check load to check load functionality
COUNTER_3	(dock enable signal to increment or decrement the counter depending on the up_down)	Randomize wih constrain to make IT active 70% of time	all values of count out and trans bins from 0 to max	assertions to check enable func and counter increment, dcrement func ,max and zero flag

3. packege code:

```
1  package counter_pkg;
2      parameter WIDTH = 4;
3      parameter MAX_VALUE = 2**WIDTH - 1;
4      class rand_stimuls;
5          rand bit             rst_n;
6          rand bit             load_n;
7          rand bit             up_down;
8          rand bit             ce;
9          rand bit [WIDTH-1:0] data_load;
10
11         //no need for constructor, they will be initialized to 0 by default
12         constraint c1 {
13             rst_n dist {1 := 9, 0 := 1};
14             ce dist {1 := 9, 0 := 1};
15             load_n dist {0 := 7, 1 := 3}; //load_n active 70%
16         }
17
18         covergroup cg1(ref bit [WIDTH-1:0]count_out, ref bit clk) @(posedge clk);
19             data_load_cp : coverpoint data_load iff(rst_n && !load_n);
20             count_out_cp : coverpoint count_out iff(rst_n && up_down && ce);
21             count_out_cp2 : coverpoint count_out iff(rst_n && up_down && ce) {
22                 bins ovrflow = (MAX_VALUE => 0);
23             }
24             count_out_cp3 : coverpoint count_out iff(rst_n && !up_down && ce);
25             count_out_cp4 : coverpoint count_out iff(rst_n && up_down && ce) {
26                 bins ovrflow = (MAX_VALUE => 0);
27             }
28         endgroup
29
30         function new (ref bit [WIDTH-1:0]count_out, ref bit clk);
31             cg1 = new(count_out, clk);
32         endfunction
33     endclass
34 endpackage
35
```

4. Testbench code:

```
1   import counter_pkg::*;
2
3   module counter_tb(counter_if.TB intrf);
4       rand_stimuls my_inputs;
5
6       logic clk;
7       assign clk = intrf.clk;
8
9       initial begin
10
11          intrf.load_n = 0;
12          intrf.up_down = 0;
13          intrf.ce = 0;
14          intrf.data_load = 0;
15
16          //reset test
17          intrf.rst_n = 0;
18          @(negedge intrf.clk);
19
20          //randomized test
21          my_inputs = new(intrf.count_out, clk);
22          for(int i = 0; i < 5000; i++) begin
23              assert(my_inputs.randomize());
24              intrf.rst_n = my_inputs.rst_n;
25              intrf.load_n = my_inputs.load_n;
26              intrf.up_down = my_inputs.up_down;
27              intrf.ce = my_inputs.ce;
28              intrf.data_load = my_inputs.data_load;
29              @(negedge intrf.clk);
30          end
31          $stop;
32      end
33  endmodule
```

5. Interface:

```
interface counter_if #(parameter WIDTH = 4)(input clk);

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

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

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

    modport SVA (input clk, rst_n, load_n, up_down, ce, data_load,
                 count_out, max_count, zero);

endinterface
```

6. Top module:

```
1  module counter_top();
2
3      logic clk;
4
5      always #1 clk = ~clk;
6      initial clk = 0;
7
8      counter_if intrf(clk);
9
10
11     counter DUT(intrf);
12     counter_tb TB(intrf);
13
14
15     bind counter counter_SVA counter_SVA_inst(intrf);
16
17 endmodule
```

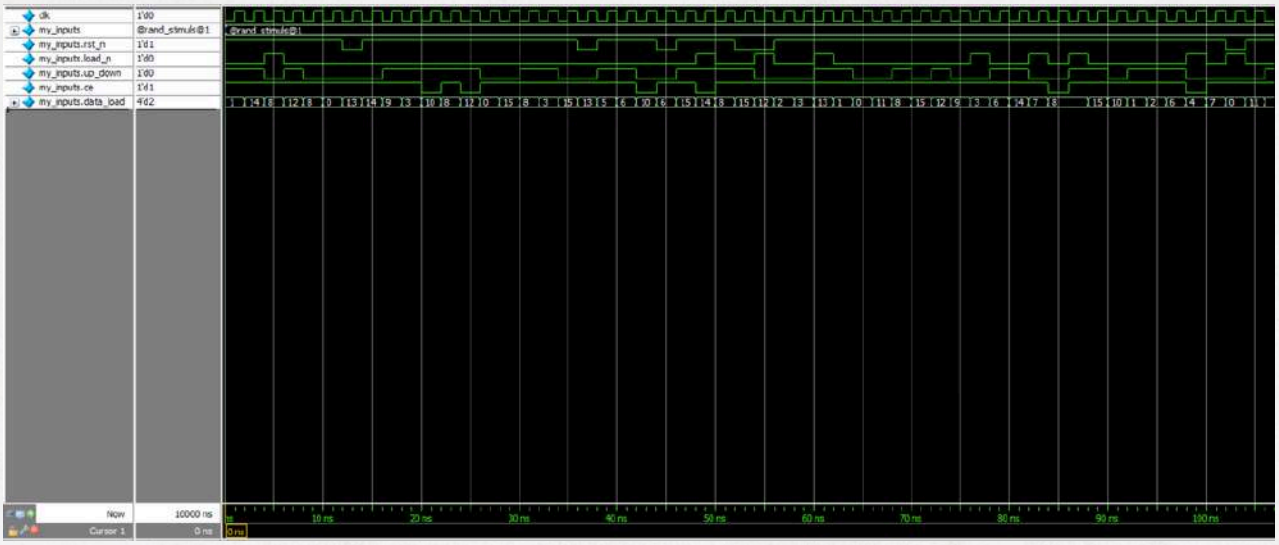
7. SVA:

```
module counter_SVA(counter_if.SVA intrf);

    property load_active;
        @(posedge intrf.clk) disable iff (!intrf.rst_n)
        (!intrf.load_n) |=> (intrf.count_out == $past(intrf.data_load));
    endproperty

    property no_load_no_cn;
        @(posedge intrf.clk) disable iff (!intrf.rst_n)
        (intrf.load_n && !intrf.ce) |=> (intrf.count_out == $past(intrf.count_out));
    endproperty

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

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

    property max_count_assert;
        @(posedge intrf.clk) disable iff (!intrf.rst_n)
        (intrf.count_out == 4'b1111) |-> intrf.max_count;
    endproperty

    property zero_assert;
        @(posedge intrf.clk) disable iff (!intrf.rst_n)
        (intrf.count_out == 4'b0000) |-> intrf.zero;
    endproperty



    always_comb begin
        if(!intrf.rst_n)
        async_rst: assert final (intrf.count_out == 0);
    end

    a1: assert property (load_active);
    a2: assert property (no_load_no_cn);
    a3: assert property (count_up);
    a4: assert property (count_down);
    a5: assert property (max_count_assert);
    a6: assert property (zero_assert);

    c1: cover property (load_active);
    c2: cover property (no_load_no_cn);
    c3: cover property (count_up);
    c4: cover property (count_down);
    c5: cover property (max_count_assert);
    c6: cover property (zero_assert);

endmodule
```

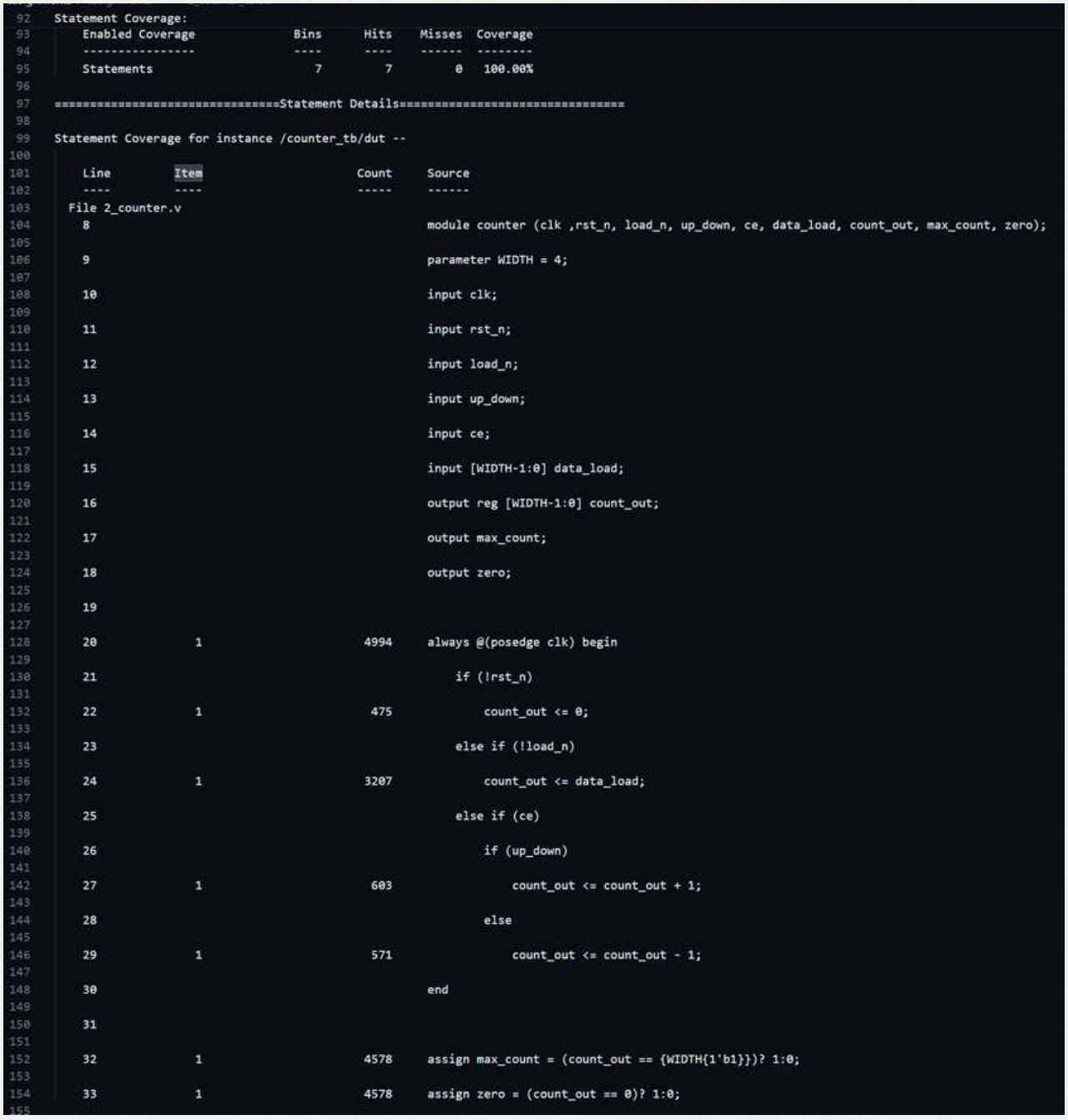
8. Do file :

```
vlib work
vlog 3_pkg.sv 3_counter.sv 3_counter_tb.sv 3_counter_top.sv 3_counter_if.sv 3_counter_SVA.sv +cover -covercells
vsim -voptargs=+acc work.counter_top -cover
add wave *
add wave -position insertpoint  \
sim:/counter_top/TB/my_inputs \
sim:/counter_top/TB/my_inputs.rst_n \
sim:/counter_top/TB/my_inputs.load_n \
sim:/counter_top/TB/my_inputs.up_down \
sim:/counter_top/TB/my_inputs.ce \
sim:/counter_top/TB/my_inputs.data_load
coverage save 3_counter_tb.ucdb -onexit

run -all
```

9. Qesta sim wave snippets (Unsigned):



10. Statement coverage report :



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

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

Statement Coverage for instance /counter_tb/dut --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 2_counter.v
    8                                                module counter (clk ,rst_n, load_n, up_down, ce, data_load, count_out, max_count, zero);

    9                                                parameter WIDTH = 4;

    10                                               input clk;

    11                                               input rst_n;

    12                                               input load_n;

    13                                               input up_down;

    14                                               input ce;

    15                                               input [WIDTH-1:0] data_load;

    16                                               output reg [WIDTH-1:0] count_out;

    17                                               output max_count;

    18                                               output zero;

    19

    20              1                         4994     always @(posedge clk) begin

    21                                                   if (!rst_n)

    22              1                          475           count_out <= 0;

    23                                                   else if (!load_n)

    24              1                         3207           count_out <= data_load;

    25                                                   else if (ce)

    26                                                       if (up_down)

    27              1                          603               count_out <= count_out + 1;

    28                                                       else

    29              1                          571               count_out <= count_out - 1;

    30                                               end

    31

    32              1                         4578     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

    33              1                         4578     assign zero = (count_out == 0)? 1:0;
```

11. Branch coverage report:

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for instance /counter_tb/dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 2_counter.v
------------------------------------IF Branch------------------------------------
    21                                     4994     Count coming in to IF
    21             1                        475         if (!rst_n)

    23             1                       3207         else if (!load_n)

    25             1                       1174         else if (ce)

                                            138     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    26                                     1174     Count coming in to IF
    26             1                        603             if (up_down)

    28             1                        571             else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    32                                     4577     Count coming in to IF
    32             1                        334     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

    32             2                       4243     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                     4577     Count coming in to IF
    33             1                        650     assign zero = (count_out == 0)? 1:0;

    33             2                       3927     assign zero = (count_out == 0)? 1:0;

Branch totals: 2 hits of 2 branches = 100.00%
```

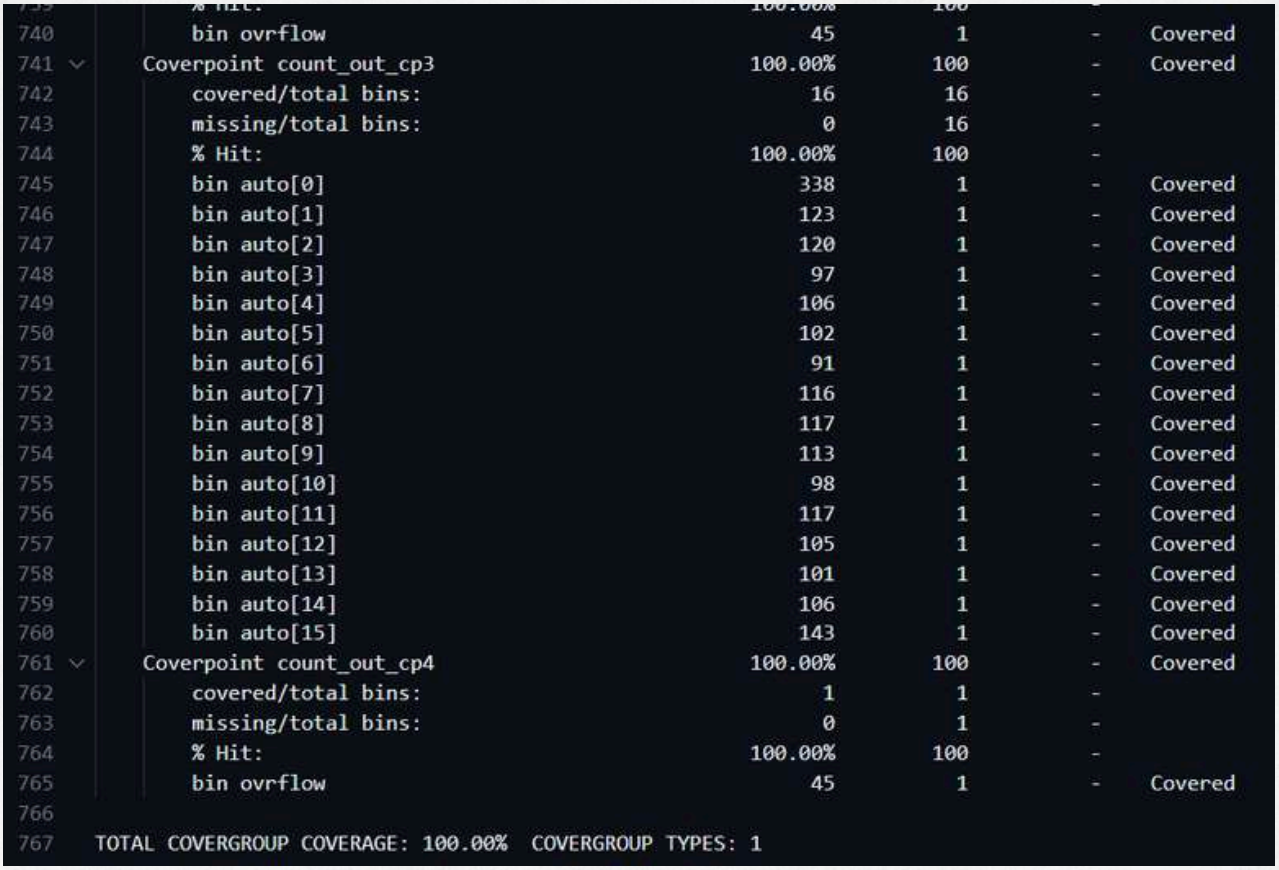
12. Toggle coverage:

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

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

Toggle Coverage for instance /counter_tb/dut --

                                            Node      1H->0L      0L->1H  "Coverage"
                                            ----------------------------------------
                                              ce           1           1      100.00
                                             clk           1           1      100.00
                                  count_out[3-0]           1           1      100.00
                                  data_load[0-3]           1           1      100.00
                                          load_n           1           1      100.00
                                       max_count           1           1      100.00
                                           rst_n           1           1      100.00
                                         up_down           1           1      100.00
                                            zero           1           1      100.00

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

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

13. Functional coverage:

```
662  COVERGROUP COVERAGE:
663  ----------------------------------------------------------------------------------------------------
664  Covergroup                                          Metric       Goal       Bins    Status
665
666  ----------------------------------------------------------------------------------------------------
667  TYPE /counter_pkg/rand_stimuls/cg1                  100.00%        100          -    Covered
668     covered/total bins:                                   50         50          -
669     missing/total bins:                                    0         50          -
670     % Hit:                                           100.00%        100          -
671     Coverpoint data_load_cp                          100.00%        100          -    Covered
672        covered/total bins:                                16         16          -
673        missing/total bins:                                 0         16          -
674        % Hit:                                        100.00%        100          -
675     Coverpoint count_out_cp                          100.00%        100          -    Covered
676        covered/total bins:                                16         16          -
677        missing/total bins:                                 0         16          -
678        % Hit:                                        100.00%        100          -
679     Coverpoint count_out_cp2                         100.00%        100          -    Covered
680        covered/total bins:                                 1          1          -
681        missing/total bins:                                 0          1          -
682        % Hit:                                        100.00%        100          -
683     Coverpoint count_out_cp3                         100.00%        100          -    Covered
684        covered/total bins:                                16         16          -
685        missing/total bins:                                 0         16          -
686        % Hit:                                        100.00%        100          -
687     Coverpoint count_out_cp4                         100.00%        100          -    Covered
688        covered/total bins:                                 1          1          -
689        missing/total bins:                                 0          1          -
690        % Hit:                                        100.00%        100          -
691  Covergroup instance \/counter_pkg::rand_stimuls::cg1
692                                                      100.00%        100          -    Covered
693     covered/total bins:                                   50         50          -
694     missing/total bins:                                    0         50          -
695     % Hit:                                           100.00%        100          -
696     Coverpoint data_load_cp                          100.00%        100          -    Covered
697        covered/total bins:                                16         16          -
698        missing/total bins:                                 0         16          -
699        % Hit:                                        100.00%        100          -
700        bin auto[0]                                       213          1          -    Covered
701        bin auto[1]                                       193          1          -    Covered
702        bin auto[2]                                       188          1          -    Covered
703        bin auto[3]                                       204          1          -    Covered
704        bin auto[4]                                       173          1          -    Covered
705        bin auto[5]                                       213          1          -    Covered
706        bin auto[6]                                       184          1          -    Covered
707        bin auto[7]                                       195          1          -    Covered
708        bin auto[8]                                       217          1          -    Covered
709        bin auto[9]                                       208          1          -    Covered
710        bin auto[10]                                      202          1          -    Covered
711        bin auto[11]                                      224          1          -    Covered
712        bin auto[12]                                      205          1          -    Covered
713        bin auto[13]                                      182          1          -    Covered
714        bin auto[14]                                      196          1          -    Covered
715        bin auto[15]                                      214          1          -    Covered
716     Coverpoint count_out_cp                          100.00%        100          -    Covered
717        covered/total bins:                                16         16          -
718        missing/total bins:                                 0         16          -
719        % Hit:                                        100.00%        100          -
720        bin auto[0]                                       284          1          -    Covered
721        bin auto[1]                                       143          1          -    Covered
722        bin auto[2]                                        92          1          -    Covered
723        bin auto[3]                                       130          1          -    Covered
724        bin auto[4]                                        98          1          -    Covered
725        bin auto[5]                                       125          1          -    Covered
726        bin auto[6]                                       114          1          -    Covered
727        bin auto[7]                                       116          1          -    Covered
728        bin auto[8]                                       106          1          -    Covered
729        bin auto[9]                                       111          1          -    Covered
730        bin auto[10]                                      120          1          -    Covered
731        bin auto[11]                                      127          1          -    Covered
732        bin auto[12]                                      119          1          -    Covered
733        bin auto[13]                                      118          1          -    Covered
734        bin auto[14]                                      117          1          -    Covered
735        bin auto[15]                                      148          1          -    Covered
736     Coverpoint count_out_cp2                         100.00%        100          -    Covered
737        covered/total bins:                                 1          1          -
738        missing/total bins:                                 0          1          -
739        % Hit:                                        100.00%        100          -
```

13. Functional coverage:



740		bin ovrflow	45	1	- Covered
741	Coverpoint count_out_cp3		100.00%	100	- Covered
742		covered/total bins:	16	16	-
743		missing/total bins:	0	16	-
744		% Hit:	100.00%	100	-
745		bin auto[0]	338	1	- Covered
746		bin auto[1]	123	1	- Covered
747		bin auto[2]	120	1	- Covered
748		bin auto[3]	97	1	- Covered
749		bin auto[4]	106	1	- Covered
750		bin auto[5]	102	1	- Covered
751		bin auto[6]	91	1	- Covered
752		bin auto[7]	116	1	- Covered
753		bin auto[8]	117	1	- Covered
754		bin auto[9]	113	1	- Covered
755		bin auto[10]	98	1	- Covered
756		bin auto[11]	117	1	- Covered
757		bin auto[12]	105	1	- Covered
758		bin auto[13]	101	1	- Covered
759		bin auto[14]	106	1	- Covered
760		bin auto[15]	143	1	- Covered
761	Coverpoint count_out_cp4		100.00%	100	- Covered
762		covered/total bins:	1	1	-
763		missing/total bins:	0	1	-
764		% Hit:	100.00%	100	-
765		bin ovrflow	45	1	- Covered
766					
767	TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1				

14. Assertion result:

```
788
789  ASSERTION RESULTS:
790  ---------------------------------------------------------------------
791  Name                 File(Line)                   Failure      Pass
792                                                    Count        Count
793  ---------------------------------------------------------------------
794  /counter_top/DUT/counter_SVA_inst/async_rst
795                       3_counter_SVA.sv(37)               0          1
796  /counter_top/DUT/counter_SVA_inst/a1
797                       3_counter_SVA.sv(40)               0          1
798  /counter_top/DUT/counter_SVA_inst/a2
799                       3_counter_SVA.sv(41)               0          1
800  /counter_top/DUT/counter_SVA_inst/a3
801                       3_counter_SVA.sv(42)               0          1
802  /counter_top/DUT/counter_SVA_inst/a4
803                       3_counter_SVA.sv(43)               0          1
804  /counter_top/DUT/counter_SVA_inst/a5
805                       3_counter_SVA.sv(44)               0          1
806  /counter_top/DUT/counter_SVA_inst/a6
807                       3_counter_SVA.sv(45)               0          1
808  /counter_top/TB/#anonblk#95084642#22#4#/#ublk#95084642#22/immed__23
809                       3_counter_tb.sv(23)                0          1
810
811  Total Coverage By Instance (filtered view): 100.00%
812
```

15. Assertion coverage:

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/counter_top/DUT/counter_SVA_inst/c1    counter_SVA Verilog  SVA  3_counter_SVA.sv(47)
                                                                          2900 Covered
/counter_top/DUT/counter_SVA_inst/c2    counter_SVA Verilog  SVA  3_counter_SVA.sv(48)
                                                                          128 Covered
/counter_top/DUT/counter_SVA_inst/c3    counter_SVA Verilog  SVA  3_counter_SVA.sv(49)
                                                                          543 Covered
/counter_top/DUT/counter_SVA_inst/c4    counter_SVA Verilog  SVA  3_counter_SVA.sv(50)
                                                                          530 Covered
/counter_top/DUT/counter_SVA_inst/c5    counter_SVA Verilog  SVA  3_counter_SVA.sv(51)
                                                                          326 Covered
/counter_top/DUT/counter_SVA_inst/c6    counter_SVA Verilog  SVA  3_counter_SVA.sv(52)
                                                                          708 Covered
```

Question 4:

1. Testbench:

```
module config_reg_tb();

    typedef enum {
        adc0_reg,
        adc1_reg,
        temp_sensor0_reg,
        temp_sensor1_reg,
        analog_test,
        digital_test,
        amp_gain,
        digital_config,
        break_loop
    } bank_reg;

    logic clk;
    logic reset;
    logic write;
    logic [15:0] data_in;
    bank_reg address;
    logic [15:0] data_out;
    config_reg dut( clk, reset, write, data_in, address, data_out);

    integer correct_count, error_count;

    logic [15:0] reset_assoc[string] = '{
        "adc0_reg" : 16'hFFFF,
        "adc1_reg" : 16'h0,
        "temp_sensor0_reg" : 16'h0,
        "temp_sensor1_reg" : 16'h0,
        "analog_test" : 16'hABCD,
        "digital_test" : 16'h0,
        "amp_gain" : 16'h0,
        "digital_config" : 16'h1
    };

    logic [15:0] data_out_exp;

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

    initial begin
        correct_count = 0;
        error_count = 0;
        write = 0;
        data_in = 0;

        assert_reset();
```

1. Testbench:

```
        //write / read test
        write = 1;

        for(address = address.first; address < address.last; address = address.next) begin
            data_in = 16'h0000;
            data_out_exp = 16'h0000;
            @(negedge clk);
            check_result();

            data_in = 16'hffff;
            data_out_exp = 16'hffff;
            @(negedge clk);
            check_result();

            data_in = 16'h8000;
            data_out_exp = 16'h8000;
            @(negedge clk);

            check_result();
            data_in = 16'h4000;
            data_out_exp = 16'h4000;
            @(negedge clk);
            check_result();

            data_in = 16'h0001;
            data_out_exp = 16'h0001;
            @(negedge clk);
            check_result();

            data_in = 16'h0010;
            data_out_exp = 16'h0010;
            @(negedge clk);
            check_result();

            $display("////////////////////////////////");
        end

        $display("Correct: %d, Error: %d", correct_count, error_count);
        $stop;
    end


    task assert_reset();
        reset = 1;
        @(negedge clk);
        check_reset();
        reset = 0;
    endtask
```

1. Testbench:

```
    task check_reset();
        $display("//////////////////RESET//////////////////");
        for (address = address.first(); address < address.last(); address = address.next()) begin
            @(negedge clk);
            if(data_out != reset_assoc[$sformatf("%s", address.name())]) begin
                $display("RESET ERROR: Address %s, expected %b, got %b", address, reset_assoc[$sformatf("%s", address.name())], data_out);
                error_count++;
            end
            else begin
                correct_count++;
            end
        end
        $display("////////////////////////////////////");

    endtask

    task check_result();
            if(data_out != data_out_exp) begin
                $display("ERROR: Address %s, expected %b, got %b", address, data_out_exp, data_out);
                error_count++;
            end
            else begin
                correct_count++;
                $display("ERROR: Address %s, expected %b, got %b", address, data_out_exp, data_out);
            end

    endtask

endmodule
```

2. Verification plan:

1	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	CONFIG_REG_1	After reset is asserted, all registers should take their default reset values from reset_assoc.	Apply reset=1, loop through all register addresses, compare data_out with reset_assoc values.		A checker compares data_out with expected reset values and increments correct/error counters.
3	CONFIG_REG_2	When write is enabled, each register should update with the given data_in value on the selected address.	Set write=1, data_in=16'hA, loop through all register addresses and check data_out matches data_in.		A checker compares data_out with written data_in values for each address.
4	CONFIG_REG_3	assert rst with write rst must dominate	Set write=1 ,rst=1, data_in=16'hFFFF, loop through all register addresses and check data_out matches data_in.		A checker ensures new data overwrites old data correctly.

3. Bug report:
   1. **adc0_reg**
      - Bug: MSB stuck at 1.
   2. **adc1_reg**
      - Bug: Performs rotate-right by 8 bits on write.
   3. **temp_sensor0_reg**
      - Bug: Shifts data left by 1 bit.
   4. **temp_sensor0_reg**
      - Bug: Writes occur during reset (should be blocked).
   5. **digital_test**
      - Bug: Updates only after the second positive clock edge.
   6. **analog_test**
      - Bug: Incorrect reset value.
   7. **amp_gain**
      - Bug: Behave combinationally.
   8. **digital_config**
      - Bug: MSB stuck at 0.

3. Questasim snippetes

```
# ////////////////////RESET////////////////////
# RESET ERROR: Address analog_test, expected 1010101111001101, got 1010101111001100
# ////////////////////////////////////////
# ERROR: Address break_loop, expected 0100000000000000, got 0000000000000001
# ERROR: Address adc0_reg, expected 0000000000000000, got 1000000000000000
# ERROR: Address adc0_reg, expected 1111111111111111, got 1111111111111111
# ERROR: Address adc0_reg, expected 1000000000000000, got 1000000000000000
# ERROR: Address adc0_reg, expected 0100000000000000, got 1100000000000000
# ERROR: Address adc0_reg, expected 0000000000000001, got 1000000000000001
# ERROR: Address adc0_reg, expected 0000000000010000, got 1000000000010000
# ////////////////////////////////
# ERROR: Address adc1_reg, expected 0000000000000000, got 0000000000000000
# ERROR: Address adc1_reg, expected 1111111111111111, got 1111111111111111
# ERROR: Address adc1_reg, expected 1000000000000000, got 0000000010000000
# ERROR: Address adc1_reg, expected 0100000000000000, got 0000000001000000
# ERROR: Address adc1_reg, expected 0000000000000001, got 0000000100000000
# ERROR: Address adc1_reg, expected 0000000000010000, got 0001000000000000
# ////////////////////////////////
# ERROR: Address temp_sensor0_reg, expected 0000000000000000, got 0000000000000000
# ERROR: Address temp_sensor0_reg, expected 1111111111111111, got 1111111111111110
# ERROR: Address temp_sensor0_reg, expected 1000000000000000, got 0000000000000000
# ERROR: Address temp_sensor0_reg, expected 0100000000000000, got 1000000000000000
# ERROR: Address temp_sensor0_reg, expected 0000000000000001, got 0000000000000010
# ERROR: Address temp_sensor0_reg, expected 0000000000010000, got 0000000000100000
# ////////////////////////////////
# ERROR: Address temp_sensor1_reg, expected 0000000000000000, got 0000000000000000
# ERROR: Address temp_sensor1_reg, expected 1111111111111111, got 1111111111111111
# ERROR: Address temp_sensor1_reg, expected 1000000000000000, got 1000000000000000
# ERROR: Address temp_sensor1_reg, expected 0100000000000000, got 0100000000000000
# ERROR: Address temp_sensor1_reg, expected 0000000000000001, got 0000000000000001
# ERROR: Address temp_sensor1_reg, expected 0000000000010000, got 0000000000010000
# ////////////////////////////////
```

```
# ERROR: Address analog_test, expected 0000000000000000, got 0000000000000000
# ERROR: Address analog_test, expected 1111111111111111, got 1111111111111111
# ERROR: Address analog_test, expected 1000000000000000, got 1000000000000000
# ERROR: Address analog_test, expected 0100000000000000, got 0100000000000000
# ERROR: Address analog_test, expected 0000000000000001, got 0000000000000001
# ERROR: Address analog_test, expected 0000000000010000, got 0000000000010000
# ////////////////////////////////
# ERROR: Address digital_test, expected 0000000000000000, got 0000000000000000
# ERROR: Address digital_test, expected 1111111111111111, got 0000000000000000
# ERROR: Address digital_test, expected 1000000000000000, got 0000000000000000
# ERROR: Address digital_test, expected 0100000000000000, got 0000000000000000
# ERROR: Address digital_test, expected 0000000000000001, got 0000000000000000
# ERROR: Address digital_test, expected 0000000000010000, got 0000000000000000
# ////////////////////////////////
# ERROR: Address amp_gain, expected 0000000000000000, got 0000000000010000
# ERROR: Address amp_gain, expected 1111111111111111, got 0000000000010000
# ERROR: Address amp_gain, expected 1000000000000000, got 0000000000010000
# ERROR: Address amp_gain, expected 0100000000000000, got 0000000000010000
# ERROR: Address amp_gain, expected 0000000000000001, got 0000000000010000
# ERROR: Address amp_gain, expected 0000000000010000, got 0000000000010000
# ////////////////////////////////
# ERROR: Address digital_config, expected 0000000000000000, got 0000000000000000
# ERROR: Address digital_config, expected 1111111111111111, got 0111111111111111
# ERROR: Address digital_config, expected 1000000000000000, got 0000000000000000
# ERROR: Address digital_config, expected 0100000000000000, got 0100000000000000
# ERROR: Address digital_config, expected 0000000000000001, got 0000000000000001
# ERROR: Address digital_config, expected 0000000000010000, got 0000000000010000
# ////////////////////////////////
# Correct:          30, Error:          27
# ** Note: $stop    : 4_buggy_reg_tb.sv(99)
#    Time: 130 ns  Iteration: 1  Instance: /config_reg_tb
# Break in Module config_reg_tb at 4_buggy_reg_tb.sv line 99
```