P1)

1. Testbench code

```
import ALSU_package::*;
module ALSU_tb ();
    reg clk_tb ;
    reg rst_tb ;
    reg signed [2:0] A_tb ;
    reg signed [2:0] B_tb ;
    reg signed [1:0] cin_tb ;
    reg serial_in_tb ;
    reg red_op_A_tb ;
    reg red_op_B_tb ;
    opcode_t opcode_tb;
    reg bypass_A_tb ;
    reg bypass_B_tb ;
    reg direction_tb;
    reg invalid ;

    reg signed [5:0] out_expected ;
    reg [16] leds_expected ;

    wire [15:0] leds_tb ;
    wire signed [5:0] out_tb ;

    ALSU_class obj1 = new ;


    ALSU #(.INPUT_PRIORITY("A"),.FULL_ADDER("ON")) DUT (A_tb, B_tb, cin_tb,
                                                        serial_in_tb,
                                                        red_op_A_tb, red_op_B_tb,
                                                        opcode_tb, bypass_A_tb,
                                                        bypass_B_tb, clk_tb, rst_tb,
                                                        direction_tb, leds_tb, out_tb);


    integer error_count , correct_count ;

    initial begin
        clk_tb = 0;
        forever begin
            #2 clk_tb    = ~clk_tb;
            obj1.clk_cv  = clk_tb ;
        end
    end
```

```
initial begin
        rst_tb = 0 ;
        error_count = 0 ;
        correct_count = 0 ;
        assert_reset () ;
        obj1.OPCODE_array.constraint_mode (0) ;
        repeat (20000) begin
            assert (obj1.randomize())
            rst_tb       = obj1.rst       ;
            A_tb         = obj1.A         ;
            B_tb         = obj1.B         ;
            cin_tb       = obj1.cin       ;
            serial_in_tb = obj1.serial_in ;
            red_op_A_tb  = obj1.red_op_A  ;
            red_op_B_tb  = obj1.red_op_B  ;
            bypass_A_tb  = obj1.bypass_A  ;
            bypass_B_tb  = obj1.bypass_B  ;
            direction_tb = obj1.direction ;
            opcode_tb    = obj1.opcode    ;
            obj1.sampling_order() ;
            $display ("current opcode_tb is %d   and time is %t"  , opcode_tb, $time ) ;
            obj1.printing() ;
            golden_model () ;
            check_result () ;
            golden_model () ;
        end
        obj1.constraint_mode (0) ;
        rst_tb = 0 ;
        bypass_A_tb = 0 ;
        bypass_B_tb = 0 ;
        red_op_A_tb = 0 ;
        red_op_B_tb = 0 ;
        obj1.OPCODE_array.constraint_mode (1) ;
        repeat (10000) begin
            assert (obj1.randomize())
            A_tb         = obj1.A         ;
            B_tb         = obj1.B         ;
            cin_tb       = obj1.cin       ;
            serial_in_tb = obj1.serial_in ;
            direction_tb = obj1.direction ;
            obj1.rst = 0 ;
            obj1.bypass_A = 0 ;
            obj1.bypass_B = 0 ;
            obj1.red_op_A = 0 ;
            obj1.red_op_B = 0 ;
            rst_tb = 0 ;
            bypass_A_tb = 0 ;
            bypass_B_tb = 0 ;
            red_op_A_tb = 0 ;
            red_op_B_tb = 0 ;
            for (int i = 0 ; i < 6 ; i = i + 1) begin
                opcode_tb = obj1.opcode_arr[5-i] ;
                obj1.sampling_order() ;
                golden_model () ;
                check_result () ;
                golden_model () ;
                $display ("ittiration number %d" , i );
                $display ("current opcode_tb is %d   and time is %t"  , opcode_tb, $time ) ;
                obj1.printing() ;
            end
        end
```

```
$display("=== Starting Directed Transition Coverage Test ===");

        // Set all control signals to safe values
        rst_tb = 0;
        bypass_A_tb = 0;
        bypass_B_tb = 0;
        red_op_A_tb = 0;
        red_op_B_tb = 0;
        obj1.rst = 0;
        obj1.bypass_A = 0;
        obj1.bypass_B = 0;
        obj1.red_op_A = 0;
        obj1.red_op_B = 0;
        assert (obj1.randomize())
            A_tb         = obj1.A         ;
            B_tb         = obj1.B         ;
            cin_tb       = obj1.cin       ;
            serial_in_tb = obj1.serial_in ;
            direction_tb = obj1.direction ;

        // Wait for stable clock
        @(posedge clk_tb);
        @(negedge clk_tb);

        // Test each transition explicitly
        $display("Testing OR -> XOR transition");
        opcode_tb = OR;
        obj1.opcode = OR;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        opcode_tb = XOR;
        obj1.opcode = XOR;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing XOR -> ADD transition");
        opcode_tb = ADD;
        obj1.opcode = ADD;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing ADD -> MULT transition");
        opcode_tb = MULT;
        obj1.opcode = MULT;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing MULT -> SHIFT transition");
        opcode_tb = SHIFT;
        obj1.opcode = SHIFT;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing SHIFT -> ROTATE transition");
        opcode_tb = ROTATE;
        obj1.opcode = ROTATE;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing ROTATE -> INVALID_6 transition");
        opcode_tb = INVALID_6;
        obj1.opcode = INVALID_6;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        $display("Testing INVALID_6 -> INVALID_7 transition");
        opcode_tb = INVALID_7;
        obj1.opcode = INVALID_7;
        @(posedge clk_tb);
        obj1.sampling_order();
        @(negedge clk_tb);

        assert_reset () ;

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

```
task golden_model ();
    leds_expected = 0 ;
    invalid = (opcode_tb == 3'b110) ||
                (opcode_tb == 3'b111) ||
                ((red_op_A_tb || red_op_B_tb) &&
                 (opcode_tb != 3'b000) && (opcode_tb != 3'b001)) ;
        if (rst_tb) begin
            out_expected  = 0 ;
        end
        else if (bypass_A_tb) out_expected = A_tb;
        else if (bypass_B_tb) out_expected = B_tb;
        else if (invalid) out_expected = 0;
        else if (opcode_tb == 0) begin
            if (red_op_A_tb) out_expected = |A_tb ;
            else if (red_op_B_tb) out_expected = |B_tb ;
            else out_expected = A_tb | B_tb ;
        end
        else if (opcode_tb == 1) begin
            if (red_op_A_tb) out_expected = ^A_tb ;
            else if (red_op_B_tb) out_expected = ^B_tb ;
            else out_expected = A_tb ^ B_tb ;
        end
        else if (opcode_tb == 2) out_expected = A_tb + B_tb + cin_tb ;
        else if (opcode_tb == 3) out_expected = A_tb * B_tb ;
        else if (opcode_tb == 4) begin
            if (direction_tb) out_expected = {out_expected[4:0] , serial_in_tb } ;
            else              out_expected = {serial_in_tb , out_expected[5:1] } ;
        end
        else if (opcode_tb == 5) begin
            if (direction_tb) out_expected = {out_expected[4:0] , out_expected[5] } ;
            else              out_expected = {out_expected[0] , out_expected[5:1] } ;
        end

        if (rst_tb) leds_expected = 0 ;
        else begin
            if (invalid)
                    leds_expected = ~leds_expected;
            // else leds_expected = 0;
        end

    endtask

    task check_result ();
        @(negedge clk_tb) ;
        @(negedge clk_tb) ;
        if (out_expected == out_tb && leds_expected == leds_tb) begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
    endtask

    task assert_reset ();
        rst_tb = 1 ;
        out_expected = 0 ;
        leds_expected = 0 ;
        @(negedge clk_tb) ;
        check_result () ;
        rst_tb = 0 ;
    endtask
endmodule
```

2. Package code

```
package ALSU_package;
    parameter MAXPOS =  3 ;
    parameter MAXNEG = -4 ;
    typedef enum logic [2:0] {
        OR, XOR, ADD, MULT, SHIFT, ROTATE, INVALID_6, INVALID_7
    } opcode_t ;
    class ALSU_class;
        rand logic rst ;
        rand logic signed [2:0] A ;
        rand logic signed [2:0] B ;
        rand logic signed [1:0] cin ;
        rand logic serial_in ;
        rand logic red_op_A ;
        rand logic red_op_B ;
        rand logic bypass_A ;
        rand logic bypass_B ;
        rand logic direction;
        rand opcode_t opcode ;
        bit [2:0] last_opcode;
        rand opcode_t [6] opcode_arr ;
        bit clk_cv ;

        constraint Arethmatic {
            if (opcode == ADD || opcode == MULT) {
                A dist {-4:/25, 0:/25, 3:/25, -3:/5, -2:/5, -1:/5, 1:/5, 2:/5};
                B dist {-4:/25, 0:/25, 3:/25, -3:/5, -2:/5, -1:/5, 1:/5, 2:/5};
            }
        }

        constraint reduction_A {
            if ( (opcode inside {XOR , OR}) && red_op_A ) {
                A dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
                        3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
                B dist {0:/100} ;
            }
        }

        constraint reduction_B {
            if ( (opcode inside {XOR , OR}) && red_op_B ) {
                B dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
                        3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
                A dist {0:/100} ;
            }
        }

        constraint OPCODE {
            opcode dist { [0:5] :/ 95,  [6:7] :/ 5 };
        }

        constraint BYPASS {
            bypass_A dist {0:/95 , 1:/5} ;
            bypass_B dist {0:/95 , 1:/5} ;
        }

        constraint RESET {
            rst dist {0:/99 , 1:/1} ;
        }

        constraint RED_OP {
            if (opcode == OR || opcode == XOR ) {
                red_op_A dist {0:/20 , 1:/80} ;
                red_op_B dist {0:/30 , 1:/70} ;
            }
            else {
                red_op_A dist {0:/95 , 1:/5} ;
                red_op_B dist {0:/95 , 1:/5} ;
            }
        }

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

            foreach (opcode_arr[i])
                foreach (opcode_arr[j])
                  if (i != j) opcode_arr[i] != opcode_arr[j];
        }
```

```
covergroup covcode @(posedge clk_cv);
            A_cp : coverpoint A {
                bins A_data_0             =    {  0  }    ;
                bins A_data_max           =    {MAXPOS}   ;
                bins A_data_min           =    {MAXNEG}   ;
                bins A_data_default       =    default    ;
                bins A_data_walkingones[] = {001,010,100} iff (red_op_A ) ;
            }
            B_cp : coverpoint B {
                bins B_data_0             =    {  0  }    ;
                bins B_data_max           =    {MAXPOS}   ;
                bins B_data_min           =    {MAXNEG}   ;
                bins B_data_default       =    default    ;
                bins B_data_walkingones[] = {001,010,100} iff (red_op_B) ;
            }
            ALU_CP : coverpoint opcode {
                bins Bins_shift[]    = {SHIFT , ROTATE} ;
                bins Bins_arith[]    = { ADD  ,  MULT } ;
                bins Bins_bitwise[]  = {OR , XOR} ;
                bins Bins_invalid    = {INVALID_6 , INVALID_7} ;
                bins trans  = (OR => XOR => ADD => MULT => SHIFT => ROTATE) ;
            }
            ALU_TRANS : coverpoint opcode {
                bins trans  = (OR => XOR => ADD => MULT => SHIFT => ROTATE => INVALID_6 => INVALID_7) ;
            }
            arith_A: cross A_cp  , ALU_CP {
                option.cross_auto_bin_max = 0 ;
                bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_0) ;
                bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_max) ;
                bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_min) ;
            }
            arith_B: cross B_cp  , ALU_CP {
                option.cross_auto_bin_max = 0 ;
                bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_0) ;
                bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_max) ;
                bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_min) ;
            }
            cin_cp: coverpoint cin iff (opcode == ADD) {
                bins zero_cin = {0} ;
                bins one_cin  = {1} ;
            }
            direction_cp : coverpoint direction iff (opcode inside {SHIFT , ROTATE}) {
                bins zero_direction = {0} ;
                bins one_direction  = {1} ;
            }
            serial_in_cp : coverpoint serial_in iff (opcode == SHIFT) {
                bins zero_serial_in  = {0} ;
                bins one__serial_in  = {1} ;
            }
            walking_one_A : cross ALU_CP , A_cp , B_cp {
                option.cross_auto_bin_max = 0 ;
                bins A_walkingones = binsof (ALU_CP.Bins_bitwise) && binsof (A_cp.A_data_walkingones)
&& binsof (B_cp.B_data_0) iff (red_op_A)  ;
            }
            walking_one_B : cross ALU_CP , A_cp , B_cp {
                option.cross_auto_bin_max = 0 ;
                bins B_walkingones = binsof (ALU_CP.Bins_bitwise) && binsof (B_cp.B_data_walkingones)
&& binsof (A_cp.A_data_0) iff (red_op_B)   ;
            }
            invalid_cp : coverpoint opcode iff (red_op_A || red_op_B) {
                bins invalid_opcode = {ADD , MULT , SHIFT , ROTATE , INVALID_6 , INVALID_7};
            }

        endgroup

        function new ();
            covcode = new () ;
        endfunction

        function  printing ();
            $display ("current opcode_arr is %d , %d , %d , %d , %d , %d  and time is %t" ,
opcode_arr[5] ,opcode_arr[4] , opcode_arr[3] , opcode_arr[2] , opcode_arr[1] , opcode_arr[0] , $time )
;
            $display ("current opcode is %d and time is %t" , opcode , $time ) ;
        endfunction

        function printing2 ();
            $display ("current opcode is %d and time is %t" , opcode , $time ) ;
        endfunction

        function sampling_order();
            if (!(rst || red_op_A || red_op_B))
                covcode.sample() ;
        endfunction

    endclass
endpackage
```

3. Design code

```
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction,
leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input signed [1:0] cin ;
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;

//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

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

//ALSU output processing
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
    if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else if (invalid)
       out <= 0;
    else begin
        case (opcode_reg)
          3'h0: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
            else if (red_op_A_reg)
              out <= |A_reg;
            else if (red_op_B_reg)
              out <= |B_reg;
            else
              out <= A_reg | B_reg;
          end
          3'h1: begin
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
            else if (red_op_A_reg)
              out <= ^A_reg;
            else if (red_op_B_reg)
              out <= ^B_reg;
            else
              out <= A_reg ^ B_reg;
          end
          3'h2: out <= (FULL_ADDER == "ON")? A_reg + B_reg + cin_reg : A_reg + B_reg;
          3'h3: out <= A_reg * B_reg  ;
          3'h4: begin
            if (direction_reg)
              out <= {out[4:0], serial_in_reg};
            else
              out <= {serial_in_reg, out[5:1]};
          end
          3'h5: begin
            if (direction_reg)
              out <= {out[4:0], out[5]};
            else
              out <= {out[0], out[5:1]};
          end
          default : out = 0 ;


        endcase
    end
  end
end

endmodule
```

4. Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
TEST_1	When the reset is asserted by task **assert_reset**, the output value should be low	Directed at the start of the simulation, then randomized with constraint that drive the reset to be most of the		A checker in the testbench to make sure the output is correct
TEST_2	when the **bypass_A** is high ,the output take the value of operand A ignoring opcode value, when the **bypass_A** is low & **bypass_A** is low , the output take the value of operand A ignoring opcode value when both are high the output take the value of A or B depending on **INPUT_PRIORITY** value	Randomized under constrains on the **bypass_A** & **bypass_B** to be off 95% of simulation	cover all values of **bypass_A** & **bypass_A** signals	A checker in the testbench to make sure the output is correct
TEST_3	when the **opcode** take the value 6 or 7 (INVALID_OPCODES) the out should be 0 and leds blinking	Randomized under constrains on the **opcode** to take the (value 6 or 7) 70% of simulation	Ensure that the opcode field is covered when it takes the values 6 and 7	A checker in the testbench to make sure the output is correct
TEST_4	when **red_op_A** or **red_op_B** is high and **opcode** value is not OR or XOR ,the out should be 0 and leds blinking	Randomized under constraint on **red_op_A** & **red_op_B** when the opcode is not **OR** or **XOR** to be off 95% of simulation	cover all values of **red_op_A** & **red_op_B** signals when the opcode is not **OR** or XOR	A checker in the testbench to make sure the output is correct
TEST_5	when **opcode** operation is **OR** if **red_op_A** is high the output should be **reduction OR of operand A** , if **red_op_A** is low & **red_op_B** is high the output should be **reduction OR of operand B** when both are high the output depend on **INPUT_PRIORITY** value If neither reduction control is asserted, the expected output is the **bitwise OR of operands A and B**	Randomized under constraint when the opcode is **OR** on **red_op_A** to be off 80% of simulation and **red_op_B** to be off 70% of simulation	cover all values of **red_op_A** & **red_op_B** signals when the **opcode** is **OR**	A checker in the testbench to make sure the output is correct
TEST_6	when **opcode** operation is **XOR** if **red_op_A** is high the output should be **reduction XOR of operand A** , if **red_op_A** is low & **red_op_B** is high the output should be **reduction XOR of operand B** when both are high the output depend on **INPUT_PRIORITY** value If neither reduction control is asserted, the expected output is the **bitwise XOR of operands A and B**	Randomized under constraint when the opcode is **XOR** on **red_op_A** to be off 80% of simulation and **red_op_B** to be off 70% of simulation	cover all values of **red_op_A** & **red_op_B** signals when the **opcode** is **OR**	A checker in the testbench to make sure the output is correct
TEST_7	when **opcode** operation is **ADD** , the **output** should be equal the **summition** of **operand A** & **operand B** , if **FULL_ADDER** is **ON** we take in consider the **CIN** value else we ignore it	Randomized under constraint when the opcode is **ADD** on **operand A & operand B** to take value (-4) 25% of simulation , take value (0) 25% of simulation , take value (3) 25% of simulation and take the rest of values 25% of simulation	cover all values of **operand A** & **operand B** inputs when the **opcode** is **ADD** cover the values of **c_in** to be **zero** OR **one**	A checker in the testbench to make sure the output is correct
TEST_8	when **opcode** operation is **MULT** , the **output** should be equal the **multiplication** of **operand A** & **operand B**	Randomized under constraint when the opcode is **MULT** on **operand A & operand B** to take value (-4) 25% of simulation , take value (0) 25% of simulation , take value (3) 25% of simulation and take the rest of values 25% of simulation	cover all values of **operand A** & **operand B** inputs when the **opcode** is **MULT**	A checker in the testbench to make sure the output is correct
TEST_9	when **opcode** operation is **SHIFT** , the **output** should be shifted one bit by **serial_in** input to **left** if **direction** signal is **high** and one bit by **serial_in** input to **right** if **direction** signal is **low**	Randomized the **direction** signal and **serial_in** input without any constraints	cover all values of **direction** signal and **serial_in** input when the **opcode** is **SHIFT**	A checker in the testbench to make sure the output is correct
TEST_10	when **opcode** operation is **ROTATE** , the **output** should be rotated one bit **left** if **direction** signal is **high** and one bit to **right** if **direction** signal is **low**	Randomized the **direction** signal without any constraints	cover all values of **direction** signal input when the **opcode** is **ROTATE**	A checker in the testbench to make sure the output is correct
TEST_11	When the reset is asserted, the output value should be low	Directed at the end of the simulation		A checker in the testbench to make sure the output is correct

5. Do file

```
vlib work
vlog ALSU.v Package.svh TESTBENCH.svh +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage exclude -src ALSU.v -line 112 -code s
coverage save ALSU.ucdb -onexit -du ALSU
run -all
```

6. Code Coverage report snippets

```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------------
Covergroup                                                Metric       Goal       Bins    Status
----------------------------------------------------------------------------------------------------------------
 TYPE /ALSU_package/ALSU_class/covcode                   100.00%        100          -    Covered
    covered/total bins:                                       32         32          -
    missing/total bins:                                        0         32          -
    % Hit:                                                100.00%        100          -
    Coverpoint A_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
    Coverpoint B_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
    Coverpoint ALU_CP                                     100.00%        100          -    Covered
        covered/total bins:                                    8          8          -
        missing/total bins:                                    0          8          -
        % Hit:                                            100.00%        100          -
    Coverpoint ALU_TRANS                                  100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
    Coverpoint cin_cp                                     100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint direction_cp                               100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint serial_in_cp                               100.00%        100          -    Covered
        covered/total bins:                                    2          2          -
        missing/total bins:                                    0          2          -
        % Hit:                                            100.00%        100          -
    Coverpoint invalid_cp                                 100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
    Cross arith_A                                         100.00%        100          -    Covered
        covered/total bins:                                    3          3          -
        missing/total bins:                                    0          3          -
        % Hit:                                            100.00%        100          -
```

```
    Cross arith_B                                         100.00%        100          -    Covered
        covered/total bins:                                    3          3          -
        missing/total bins:                                    0          3          -
        % Hit:                                            100.00%        100          -
    Cross walking_one_A                                   100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
    Cross walking_one_B                                   100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
Covergroup instance \/ALSU_package::ALSU_class::covcode
                                                          100.00%        100          -    Covered
    covered/total bins:                                       32         32          -
    missing/total bins:                                        0         32          -
    % Hit:                                                100.00%        100          -
    Coverpoint A_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
        bin A_data_0                                       18496          1          -    Covered
        bin A_data_max                                     16684          1          -    Covered
        bin A_data_min                                     16580          1          -    Covered
        bin A_data_walkingones[1]                            762          1          -    Covered
        default bin A_data_default                         60519                     -    Occurred
    Coverpoint B_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
        bin B_data_0                                       19542          1          -    Covered
        bin B_data_max                                     17054          1          -    Covered
        bin B_data_min                                     16954          1          -    Covered
        bin B_data_walkingones[1]                            470          1          -    Covered
        default bin B_data_default                         59767                     -    Occurred
    Coverpoint ALU_CP                                     100.00%        100          -    Covered
        covered/total bins:                                    8          8          -
        missing/total bins:                                    0          8          -
        % Hit:                                            100.00%        100          -
        bin Bins_shift[SHIFT]                              17035          1          -    Covered
        bin Bins_shift[ROTATE]                             16435          1          -    Covered
        bin Bins_arith[ADD]                                16965          1          -    Covered
        bin Bins_arith[MULT]                               16989          1          -    Covered
        bin Bins_bitwise[OR]                               16273          1          -    Covered
        bin Bins_bitwise[XOR]                              16252          1          -    Covered
        bin Bins_invalid                                   28063          1          -    Covered
        bin trans                                              1          1          -    Covered
```

```
    Coverpoint ALU_TRANS                          100.00%        100          -    Covered
        covered/total bins:                             1          1          -
        missing/total bins:                             0          1          -
        % Hit:                                    100.00%        100          -
        bin trans                                       1          1          -    Covered
    Coverpoint cin_cp                             100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                    100.00%        100          -
        bin zero_cin                                 4198          1          -    Covered
        bin one_cin                                  4126          1          -    Covered
    Coverpoint direction_cp                       100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                    100.00%        100          -
        bin zero_direction                          16746          1          -    Covered
        bin one_direction                           16724          1          -    Covered
    Coverpoint serial_in_cp                       100.00%        100          -    Covered
        covered/total bins:                             2          2          -
        missing/total bins:                             0          2          -
        % Hit:                                    100.00%        100          -
        bin zero_serial_in                           8576          1          -    Covered
        bin one__serial_in                           8459          1          -    Covered
    Coverpoint invalid_cp                         100.00%        100          -    Covered
        covered/total bins:                             1          1          -
        missing/total bins:                             0          1          -
        % Hit:                                    100.00%        100          -
        bin invalid_opcode                           1471          1          -    Covered
    Cross arith_A                                 100.00%        100          -    Covered
        covered/total bins:                             3          3          -
        missing/total bins:                             0          3          -
        % Hit:                                    100.00%        100          -
        Auto, Default and User Defined Bins:
            bin arith_zero                           4844          1          -    Covered
            bin arith_max                            5146          1          -    Covered
            bin arith_min                            4536          1          -    Covered
    Cross arith_B                                 100.00%        100          -    Covered
        covered/total bins:                             3          3          -
        missing/total bins:                             0          3          -
        % Hit:                                    100.00%        100          -
        Auto, Default and User Defined Bins:
            bin arith_zero                           5008          1          -    Covered
            bin arith_max                            5122          1          -    Covered
            bin arith_min                            5194          1          -    Covered
```

```
    Cross walking_one_A                           100.00%        100          -    Covered
        covered/total bins:                             1          1          -
        missing/total bins:                             0          1          -
        % Hit:                                    100.00%        100          -
        Auto, Default and User Defined Bins:
            bin A_walkingones                         680          1          -    Covered
    Cross walking_one_B                           100.00%        100          -    Covered
        covered/total bins:                             1          1          -
        missing/total bins:                             0          1          -
        % Hit:                                    100.00%        100          -
        Auto, Default and User Defined Bins:
            bin B_walkingones                         398          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%
```

7. Functional Coverage report snippets

```
# ================================================================================
# === Instance: /\ALSU_tb#DUT
# === Design Unit: work.ALSU
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                        32        31         1    96.87%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\ALSU_tb#DUT
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File ALSU.v
# ------------------------------------IF Branch------------------------------------
#     25                                     159990     Count coming in to IF
#     25              1                         362       if(rst) begin
#
#     36              1                      159628       end else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     52                                     160202     Count coming in to IF
#     52              1                         546       if(rst) begin
#
#     54              1                      159656       end else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     55                                     159656     Count coming in to IF
#     55              1                        4432            if (invalid)
#
#     57              1                      155224            else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
```

```
# ------------------------------------IF Branch------------------------------------
#     68                                     147204     Count coming in to IF
#     68              1                         80         if (bypass_A_reg && bypass_B_reg)
#
#     70              1                       1850         else if (bypass_A_reg)
#
#     72              1                       1786         else if (bypass_B_reg)
#
#     74              1                       3369         else if (invalid)
#
#     76              1                     140119         else begin
#
# Branch totals: 5 hits of 5 branches = 100.00%
#
# ------------------------------------CASE Branch------------------------------------
#     77                                     140119     Count coming in to CASE
#     78              1                      23658              3'h0: begin
#
#     88              1                      23920              3'h1: begin
#
#     98              1                      23869              3'h2: out <= (FULL_ADDER == "ON")? A_reg + B_reg + cin_reg : A_reg + B_reg;
#
#     99              1                      24020              3'h3: out <= A_reg * B_reg  ;
#
#     100             1                      23256              3'h4: begin
#
#     106             1                      21396              3'h5: begin
#
#     112             1                     ***0***             default : out = 0 ;
#
# Branch totals: 6 hits of 7 branches = 85.71%
#
# ------------------------------------IF Branch------------------------------------
#     79                                     23658     Count coming in to IF
#     79              1                         61               if (red_op_A_reg && red_op_B_reg)
#
#     81              1                        849               else if (red_op_A_reg)
#
#     83              1                       1320               else if (red_op_B_reg)
#
#     85              1                      21428               else
#
# Branch totals: 4 hits of 4 branches = 100.00%
```

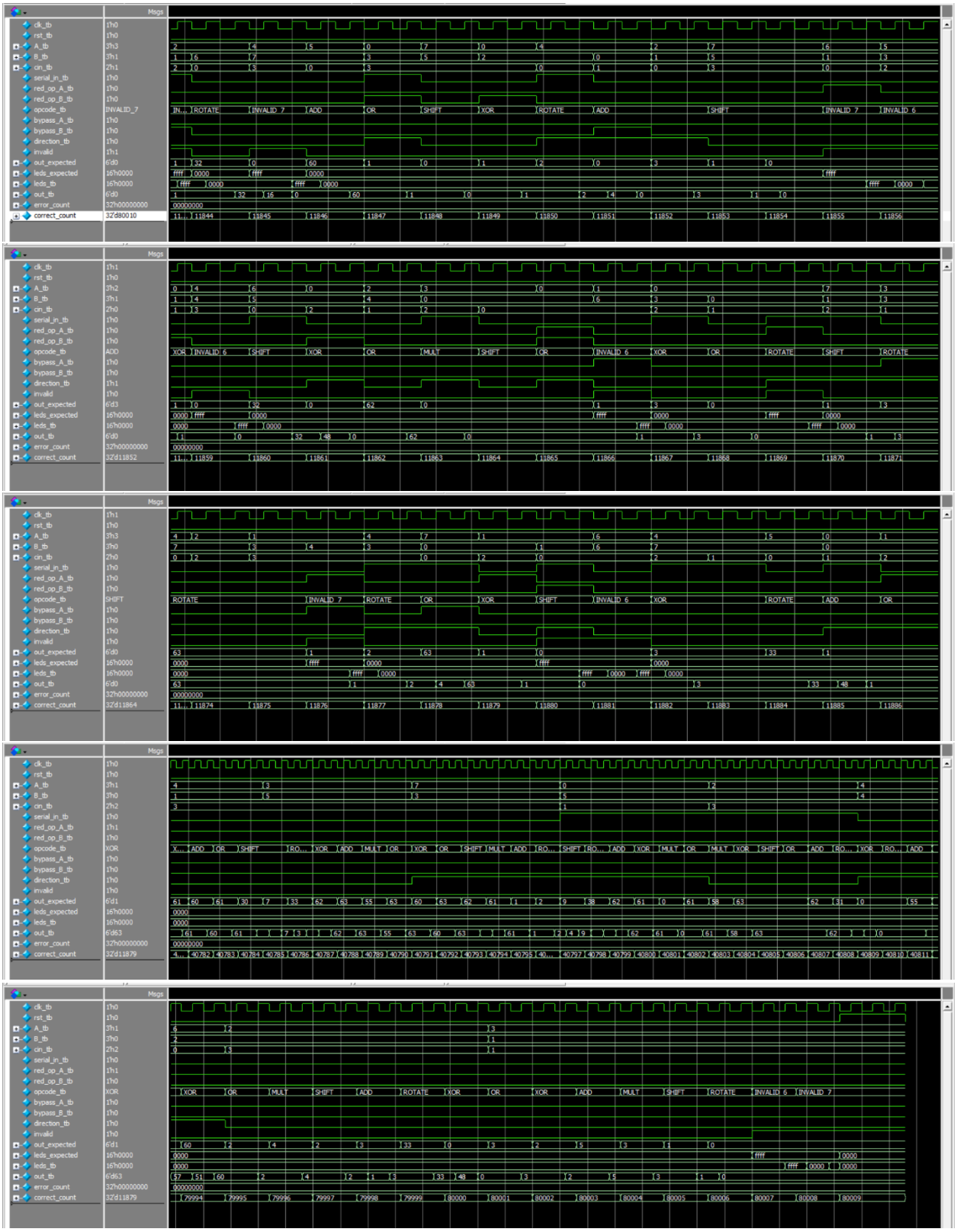
```
#
# ------------------------------------IF Branch------------------------------------
#     89                                    23920     Count coming in to IF
#     89              1                        52                 if (red_op_A_reg && red_op_B_reg)
#
#     91              1                       864                 else if (red_op_A_reg)
#
#     93              1                      1308                 else if (red_op_B_reg)
#
#     95              1                     21696                 else
#
# Branch totals: 4 hits of 4 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     101                                   23256     Count coming in to IF
#     101             1                     11726                 if (direction_reg)
#
#     103             1                     11530                 else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     107                                   21396     Count coming in to IF
#     107             1                     10949                 if (direction_reg)
#
#     109             1                     10447                 else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
#
# Condition Coverage:
#     Enabled Coverage              Bins   Covered    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Conditions                       6         6         0   100.00%
#
# ================================Condition Details================================
#
# Condition Coverage for instance /\ALSU_tb#DUT  --
#
#   File ALSU.v
# ----------------Focused Condition View-------------------
# Line       68 Item    1  (bypass_A_reg && bypass_B_reg)
# Condition totals: 2 of 2 input terms covered = 100.00%
#
#     Input Term   Covered  Reason for no coverage   Hint
#    -----------  --------  -----------------------  --------------
#   bypass_A_reg         Y
#   bypass_B_reg         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  bypass_A_reg_0        -
#   Row   2:          1  bypass_A_reg_1        bypass_B_reg
#   Row   3:          1  bypass_B_reg_0        bypass_A_reg
#   Row   4:          1  bypass_B_reg_1        bypass_A_reg
#
# ----------------Focused Condition View-------------------
# Line       79 Item    1  (red_op_A_reg && red_op_B_reg)
# Condition totals: 2 of 2 input terms covered = 100.00%
#
#     Input Term   Covered  Reason for no coverage   Hint
#    -----------  --------  -----------------------  --------------
#   red_op_A_reg         Y
#   red_op_B_reg         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  red_op_A_reg_0        -
#   Row   2:          1  red_op_A_reg_1        red_op_B_reg
#   Row   3:          1  red_op_B_reg_0        red_op_A_reg
#   Row   4:          1  red_op_B_reg_1        red_op_A_reg
```

```
#
# ----------------Focused Condition View-------------------
# Line       89 Item    1  (red_op_A_reg && red_op_B_reg)
# Condition totals: 2 of 2 input terms covered = 100.00%
#
#     Input Term   Covered  Reason for no coverage   Hint
#    -----------  --------  ----------------------  --------------
#   red_op_A_reg         Y
#   red_op_B_reg         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  red_op_A_reg_0        -
#   Row   2:          1  red_op_A_reg_1        red_op_B_reg
#   Row   3:          1  red_op_B_reg_0        red_op_A_reg
#   Row   4:          1  red_op_B_reg_1        red_op_A_reg
#
#
# Expression Coverage:
#     Enabled Coverage              Bins   Covered    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Expressions                      8         8         0   100.00%
#
# ================================Expression Details================================
#
# Expression Coverage for instance /\ALSU_tb#DUT  --
#
#   File ALSU.v
# ----------------Focused Expression View-----------------
# Line       19 Item    1  ((red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]))
# Expression totals: 4 of 4 input terms covered = 100.00%
#
#      Input Term   Covered  Reason for no coverage   Hint
#     -----------  --------  ----------------------  --------------
#    red_op_A_reg         Y
#    red_op_B_reg         Y
#   opcode_reg[1]         Y
#   opcode_reg[2]         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  red_op_A_reg_0        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
#   Row   2:          1  red_op_A_reg_1        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_B_reg)
#   Row   3:          1  red_op_B_reg_0        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
#   Row   4:          1  red_op_B_reg_1        ((opcode_reg[1] | opcode_reg[2]) && ~red_op_A_reg)
#   Row   5:          1  opcode_reg[1]_0       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
#   Row   6:          1  opcode_reg[1]_1       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[2])
#   Row   7:          1  opcode_reg[2]_0       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])
#   Row   8:          1  opcode_reg[2]_1       ((red_op_A_reg | red_op_B_reg) && ~opcode_reg[1])
#
# ----------------Focused Expression View-----------------
# Line       20 Item    1  (opcode_reg[1] & opcode_reg[2])
# Expression totals: 2 of 2 input terms covered = 100.00%
#
#      Input Term   Covered  Reason for no coverage   Hint
#     -----------  --------  ----------------------  --------------
#   opcode_reg[1]         Y
#   opcode_reg[2]         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  opcode_reg[1]_0       opcode_reg[2]
#   Row   2:          1  opcode_reg[1]_1       opcode_reg[2]
#   Row   3:          1  opcode_reg[2]_0       opcode_reg[1]
#   Row   4:          1  opcode_reg[2]_1       opcode_reg[1]
#
# ----------------Focused Expression View-----------------
# Line       21 Item    1  (invalid_red_op | invalid_opcode)
# Expression totals: 2 of 2 input terms covered = 100.00%
#
#       Input Term   Covered  Reason for no coverage   Hint
#      -----------  --------  ----------------------  --------------
#   invalid_red_op         Y
#   invalid_opcode         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  invalid_red_op_0      ~invalid_opcode
#   Row   2:          1  invalid_red_op_1      ~invalid_opcode
#   Row   3:          1  invalid_opcode_0      ~invalid_red_op
#   Row   4:          1  invalid_opcode_1      ~invalid_red_op
#
```

```
# 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 ALSU.v
#     1                                               module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, led
#
#     2                                               parameter INPUT_PRIORITY = "A";
#
#     3                                               parameter FULL_ADDER = "ON";
#
#     4                                               input clk, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
#
#     5                                               input signed [1:0] cin ;
#
#     6                                               input [2:0] opcode;
#
#     7                                               input signed [2:0] A, B;
#
#     8                                               output reg [15:0] leds;
#
#     9                                               output reg signed [5:0] out;
#
#     10
#
#     11                                              reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
#
#     12                                              reg signed [1:0] cin_reg;
#
#     13                                              reg [2:0] opcode_reg;
#
#     14                                              reg signed [2:0] A_reg, B_reg;
#
#     15
#
#     16                                              wire invalid_red_op , invalid;
```

```
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Toggles                        122       122         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[0-1]           1           1                              100.00
#                                   cin_reg[1-0]           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     =         61
# Toggled Node Count   =         61
# Untoggled Node Count =          0
#
# Toggle Coverage      =     100.00% (122 of 122 bins)
#
```

8. Clear and neat QuestaSim waveform snippets showing the functionality of the design



```
# test passes
# correct counter =        80010 , error counter =           0
# ** Note: $stop    : TESTBENCH.svh(159)
#    Time: 640088 ns  Iteration: 1  Instance: /ALSU_tb
```

P2) Module

```
1  module assertion_t ();
2      bit a ;
3      bit b ;
4      bit c ;
5      bit clk;
6      logic [7:0] Y ;
7      bit enable ;
8      logic [3:0] D ;
9      bit valid ;
10
11     property assertion1;
12         @(posedge clk) a |-> ##2 (b == 1) ;
13     endproperty
14
15     property assertion2;
16         @(posedge clk) (a&&b) |-> ##[1:3] c ;
17     endproperty
18
19     property assertion3;
20         @(posedge clk) $rose(b) |-> ##2 !b;
21     endproperty
22
23     property assertion4;
24         @(posedge clk) enable |=> $onehot(Y) ;
25     endproperty
26
27     property assertion5;
28         @(posedge clk) (!D) |=> (!valid) ;
29     endproperty
30
31     assert property (assertion1) ;
32     assert property (assertion2) ;
33     assert property (assertion3) ;
34     assert property (assertion4) ;
35     assert property (assertion5) ;
36 endmodule
```

Transcript

```
# reading C:/questasim64_2021.1/win64/../modelsim.ini
# Loading project f
# Compile of module.sv was successful.

QuestaSim>
```

P3)

1. Testbench code

```
import counter_package::*;
module counter_tb (counter_if.TEST c_test);
    counter_class obj1 = new;

    always_comb begin
        obj1.clk_cv = c_test.clk;
    end

    always @(posedge c_test.clk) begin
        obj1.count_out = c_test.count_out;
        obj1.max_count = c_test.max_count;
        obj1.ZERO = c_test.zero;
    end

    initial begin
        c_test.rst_n = 1 ;
        // test_1
        c_test.rst_n = 0 ;
        // test_2
        repeat (10000) begin
            @(posedge c_test.clk);
            assert(obj1.randomize()) ;
            c_test.rst_n = obj1.rst_n ;
            c_test.load_n = obj1.load_n;
            c_test.ce = obj1.ce ;
            c_test.up_down = obj1.up_down ;
            c_test.data_load = obj1.data_load ;
        end
        // test_3
        c_test.rst_n = 0 ;

        $stop ;
    end
endmodule
```

2. Top module code

```
import counter_package::*;
module counter_top();
    bit clk ;
    initial begin
        clk = 0;
        forever begin
            #2 clk = ~clk;
        end
    end

    counter_if c_if (clk);
    counter inst_dut(c_if);
    counter_tb inst_test(c_if);
    bind counter counter_sva assertion_inst(.c_if(c_DUT));
endmodule
```

3. SVA module code

```
module counter_sva(counter_if.DUT c_if);
     property load_on;
        @(posedge c_if.clk) (!c_if.load_n && c_if.rst_n) |=> (c_if.count_out == $past(c_if.data_load));
    endproperty

    property load_enable_off;
        @(posedge c_if.clk) (c_if.load_n && !c_if.ce && c_if.rst_n) |=> (c_if.count_out == $past(c_if.count_out)) ;
    endproperty

    property load_off_enable_on_up;
        @(posedge c_if.clk) (c_if.load_n && c_if.ce && c_if.up_down && c_if.rst_n) |=> (c_if.count_out == $past(c_if.count_out) + 1'b1) ;
    endproperty

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

    property assert_reset;
        @(posedge c_if.clk) (!c_if.rst_n) |=> (c_if.count_out == 0) ;
    endproperty

    property max;
        @(posedge c_if.clk) (c_if.count_out == 4'b1111) |-> (c_if.max_count == 1'b1) ;
    endproperty

    property zero;
        @(posedge c_if.clk) (c_if.count_out == 1'b0) |-> (c_if.zero == 1'b1) ;
    endproperty

    assert property (load_on) ;
    assert property (load_enable_off) ;
    assert property (load_off_enable_on_up) ;
    assert property (load_off_enable_on_down) ;
    assert property (assert_reset) ;
    assert property (max) ;
    assert property (zero) ;

    cover property (load_on) ;
    cover property (load_enable_off) ;
    cover property (load_off_enable_on_up) ;
    cover property (load_off_enable_on_down) ;
    cover property (assert_reset) ;
    cover property (max) ;
    cover property (zero) ;
endmodule
```

4. Package code

```
package counter_package;
    parameter WIDTH = 4 ;
    class counter_class;
        rand logic load_n;
        rand logic rst_n ;
        rand logic up_down;
        rand logic ce ;
        rand logic [3:0] data_load ;
        bit clk_cv ;
        bit [WIDTH] count_out ;
        bit max_count ;
        bit ZERO ;

        constraint rst_n_dist {
            rst_n dist {1:/95 , 0:/5} ;
        }

        constraint load_n_dist {
            load_n dist {1:/70 , 0 :/30} ;
        }

        constraint ce_dist {
            ce dist {1:/70 , 0 :/30} ;
        }

        constraint up_down_dist {
            up_down dist {1:/70 , 0 :/30} ;
        }

        covergroup covcode @(posedge clk_cv);
            load_data_cv : coverpoint data_load iff (!load_n && rst_n) ;

            count_out_cv_1 : coverpoint count_out iff (rst_n && ce && up_down) ;

            count_out_cv_2 : coverpoint count_out iff (rst_n && ce && up_down) {
                bins overflow1 = ((1 << WIDTH) - 1 => 0 ) ;
            }

            count_out_cv_3 : coverpoint count_out iff (rst_n && ce && !up_down) ;

            count_out_cv_4 : coverpoint count_out iff (rst_n && ce && !up_down) {
                bins overflow2 = ( 0 => (1 << WIDTH) - 1 ) ;
            }

            max_count_cv    : coverpoint max_count ;

            zero_cv         : coverpoint ZERO ;

        endgroup

        function new ();
            covcode = new () ;
        endfunction
    endclass
endpackage
```

5. Interface code

```
interface counter_if(input bit clk);
    parameter WIDTH = 4;
    logic rst_n;
    logic load_n;
    logic up_down;
    logic ce;
    logic [WIDTH-1:0] data_load;
    logic [WIDTH-1:0] count_out;
    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 TEST (output rst_n , load_n , up_down , ce , data_load ,
                  input count_out , max_count , zero , clk) ;
endinterface
```

6. Design code

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: Counter Design
//
////////////////////////////////////////////////////////////////////////////////
module counter (counter_if.DUT c_DUT);
always @(posedge c_DUT.clk) begin
    if (!c_DUT.rst_n)
        c_DUT.count_out <= 0;
    else if (!c_DUT.load_n)
        c_DUT.count_out <= c_DUT.data_load;
    else if (c_DUT.ce)
        if (c_DUT.up_down)
            c_DUT.count_out <= c_DUT.count_out + 1;
        else
            c_DUT.count_out <= c_DUT.count_out - 1;
end

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

endmodule
```

7. DO file

```
vlib work
vlog *v  +cover
vsim -voptargs=+acc counter_top -cover -sv_seed random -l sim.log
add wave /counter_top/inst_dut/assertion_inst/assert__load_on /counter_top/inst_dut/assertion_inst/
assert__load_enable_off /counter_top/inst_dut/assertion_inst/assert__load_off_enable_on_up /counter_top/inst_dut
/assertion_inst/assert__load_off_enable_on_down /counter_top/inst_dut
/assertion_inst/assert__assert_reset /counter_top/inst_dut
/assertion_inst/assert__max /counter_top/inst_dut
/assertion_inst/assert__zero /counter_top/inst_test/#ublk#95084642#28/#ublk#95084642#40/immed__42
add wave -position insertpoint  \
sim:/counter_top/c_if/WIDTH \
sim:/counter_top/c_if/clk \
sim:/counter_top/c_if/rst_n \
sim:/counter_top/c_if/load_n \
sim:/counter_top/c_if/up_down \
sim:/counter_top/c_if/ce \
sim:/counter_top/c_if/data_load \
sim:/counter_top/c_if/count_out \
sim:/counter_top/c_if/max_count \
sim:/counter_top/c_if/zero
coverage save counter.ucdb -onexit
run -all
```

8. Verification plan

A	B	C	D	E
Label	**Design Requirement Description**	**Stimulus Generation**	**Functional Coverage**	**Functionality Check**
TEST_1	When the reset is asserted , the output value should be low	Directed at the start of the simulation, then randomized with constraint that drive the reset to be most of the simulation off		A concurrent assertion **non_overlapped** to check the reset functionality
TEST_2	when the **load_n** is low and reset is deasserted , the output **count_out** should take the value of **data_load** input	Randomized under constrains on the **load** signal to be off 70% of simulation	cover all values of **load_n** input signal.	A concurrent assertion **non_overlapped** to check the load data
TEST_3	when the **load_n** is high , **ce** is high , & **up_down** is high the output **count_out** should increment	Randomized under constrains on the **ce** signal to be on 70% of simulation & **up_down** signal to be on 70% of simulation	cover all values of **ce** input signal. cover all values of **up_down** input signal.	A concurrent assertion **non_overlapped** to check the **increment** output of previous clock
TEST_4	when the **load_n** is high , **ce** is high , & **up_down** is high the output **count_out** should decrement	Randomized under constrains on the **ce** signal to be on 70% of simulation & **up_down** signal to be on 70% of simulation	cover all values of **ce** input signal. cover all values of **up_down** input signal.	A concurrent assertion **non_overlapped** to check the **decrement** output of previous clock
TEST_5	when the output **count_out** is max (15) , **max_count** flag should be asserted			A concurrent assertion **overlapped** to check the flag assertion once **count_out** equal 15
TEST_6	when the output **count_out** is zero , zero flag should be asserted			A concurrent assertion **overlapped** to check the flag assertion once **count_out** equal 0
TEST_7	When the reset is asserted, the output value should be low	Directed at the end of the simulation		A concurrent assertion non_overlapped to check the reset functionality

9. Functional coverage

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

----------------------------------------------------------------------------------------------------------
 TYPE /counter_package/counter_class/covcode       100.00%        100          -    Covered
    covered/total bins:                                 54         54          -
    missing/total bins:                                  0         54          -
    % Hit:                                          100.00%        100          -
    Coverpoint load_data_cv                         100.00%        100          -    Covered
        covered/total bins:                             16         16          -
        missing/total bins:                              0         16          -
        % Hit:                                      100.00%        100          -
    Coverpoint count_out_cv_1                       100.00%        100          -    Covered
        covered/total bins:                             16         16          -
        missing/total bins:                              0         16          -
        % Hit:                                      100.00%        100          -
    Coverpoint count_out_cv_2                       100.00%        100          -    Covered
        covered/total bins:                              1          1          -
        missing/total bins:                              0          1          -
        % Hit:                                      100.00%        100          -
    Coverpoint count_out_cv_3                       100.00%        100          -    Covered
        covered/total bins:                             16         16          -
        missing/total bins:                              0         16          -
        % Hit:                                      100.00%        100          -
    Coverpoint count_out_cv_4                       100.00%        100          -    Covered
        covered/total bins:                              1          1          -
        missing/total bins:                              0          1          -
        % Hit:                                      100.00%        100          -
    Coverpoint max_count_cv                         100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                      100.00%        100          -
    Coverpoint zero_cv                              100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                      100.00%        100          -
```

```
Covergroup instance \/counter_package::counter_class::covcode
                                                 100.00%        100          -    Covered
    covered/total bins:                               54         54          -
    missing/total bins:                                0         54          -
    % Hit:                                       100.00%        100          -
    Coverpoint load_data_cv                      100.00%        100          -    Covered
        covered/total bins:                           16         16          -
        missing/total bins:                            0         16          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  184          1          -    Covered
        bin auto[1]                                  157          1          -    Covered
        bin auto[2]                                  162          1          -    Covered
        bin auto[3]                                  167          1          -    Covered
        bin auto[4]                                  170          1          -    Covered
        bin auto[5]                                  167          1          -    Covered
        bin auto[6]                                  182          1          -    Covered
        bin auto[7]                                  185          1          -    Covered
        bin auto[8]                                  210          1          -    Covered
        bin auto[9]                                  182          1          -    Covered
        bin auto[10]                                 162          1          -    Covered
        bin auto[11]                                 152          1          -    Covered
        bin auto[12]                                 194          1          -    Covered
        bin auto[13]                                 152          1          -    Covered
        bin auto[14]                                 204          1          -    Covered
        bin auto[15]                                 163          1          -    Covered
    Coverpoint count_out_cv_1                    100.00%        100          -    Covered
        covered/total bins:                           16         16          -
        missing/total bins:                            0         16          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  596          1          -    Covered
        bin auto[1]                                  378          1          -    Covered
        bin auto[2]                                  280          1          -    Covered
        bin auto[3]                                  263          1          -    Covered
        bin auto[4]                                  263          1          -    Covered
        bin auto[5]                                  240          1          -    Covered
        bin auto[6]                                  251          1          -    Covered
        bin auto[7]                                  258          1          -    Covered
        bin auto[8]                                  286          1          -    Covered
        bin auto[9]                                  283          1          -    Covered
        bin auto[10]                                 257          1          -    Covered
        bin auto[11]                                 221          1          -    Covered
        bin auto[12]                                 262          1          -    Covered
        bin auto[13]                                 258          1          -    Covered
        bin auto[14]                                 281          1          -    Covered
        bin auto[15]                                 297          1          -    Covered

    Coverpoint count_out_cv_2                    100.00%        100          -    Covered
        covered/total bins:                            1          1          -
        missing/total bins:                            0          1          -
        % Hit:                                   100.00%        100          -
        bin overflow1                                151          1          -    Covered
    Coverpoint count_out_cv_3                    100.00%        100          -    Covered
        covered/total bins:                           16         16          -
        missing/total bins:                            0         16          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                  231          1          -    Covered
        bin auto[1]                                  175          1          -    Covered
        bin auto[2]                                  124          1          -    Covered
        bin auto[3]                                  107          1          -    Covered
        bin auto[4]                                  109          1          -    Covered
        bin auto[5]                                  120          1          -    Covered
        bin auto[6]                                  100          1          -    Covered
        bin auto[7]                                  119          1          -    Covered
        bin auto[8]                                  115          1          -    Covered
        bin auto[9]                                  124          1          -    Covered
        bin auto[10]                                 118          1          -    Covered
        bin auto[11]                                 101          1          -    Covered
        bin auto[12]                                  94          1          -    Covered
        bin auto[13]                                 104          1          -    Covered
        bin auto[14]                                 111          1          -    Covered
        bin auto[15]                                 131          1          -    Covered
    Coverpoint count_out_cv_4                    100.00%        100          -    Covered
        covered/total bins:                            1          1          -
        missing/total bins:                            0          1          -
        % Hit:                                   100.00%        100          -
        bin overflow2                                 41          1          -    Covered
    Coverpoint max_count_cv                      100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                 9332          1          -    Covered
        bin auto[1]                                  668          1          -    Covered
    Coverpoint zero_cv                           100.00%        100          -    Covered
        covered/total bins:                            2          2          -
        missing/total bins:                            0          2          -
        % Hit:                                   100.00%        100          -
        bin auto[0]                                 8720          1          -    Covered
        bin auto[1]                                 1280          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

10. Assertion coverage

```
TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
--------------------------------------------------------------------------------------------
/counter_top/inst_dut/assertion_inst/cover__zero
                                        counter_sva Verilog  SVA  counter_sva.sv(44)
                                                                        1280 Covered
/counter_top/inst_dut/assertion_inst/cover__max
                                        counter_sva Verilog  SVA  counter_sva.sv(43)
                                                                         668 Covered
/counter_top/inst_dut/assertion_inst/cover__assert_reset
                                        counter_sva Verilog  SVA  counter_sva.sv(42)
                                                                         467 Covered
/counter_top/inst_dut/assertion_inst/cover__load_off_enable_on_down
                                        counter_sva Verilog  SVA  counter_sva.sv(41)
                                                                        1397 Covered
/counter_top/inst_dut/assertion_inst/cover__load_off_enable_on_up
                                        counter_sva Verilog  SVA  counter_sva.sv(40)
                                                                        3321 Covered
/counter_top/inst_dut/assertion_inst/cover__load_enable_off
                                        counter_sva Verilog  SVA  counter_sva.sv(39)
                                                                        2021 Covered
/counter_top/inst_dut/assertion_inst/cover__load_on
                                        counter_sva Verilog  SVA  counter_sva.sv(38)
                                                                        2792 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 7

Total Coverage By Instance (filtered view): 100.00%
```

11. Code coverage

```
# ================================================================================
# === Instance: /counter_top/c_if
# === Design Unit: work.counter_if
# ================================================================================
# Toggle Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Toggles                         30        30         0   100.00%
#
# ================================Toggle Details================================
#
# Toggle Coverage for instance /counter_top/c_if --
#
#                                             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[3-0]            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)
#
```

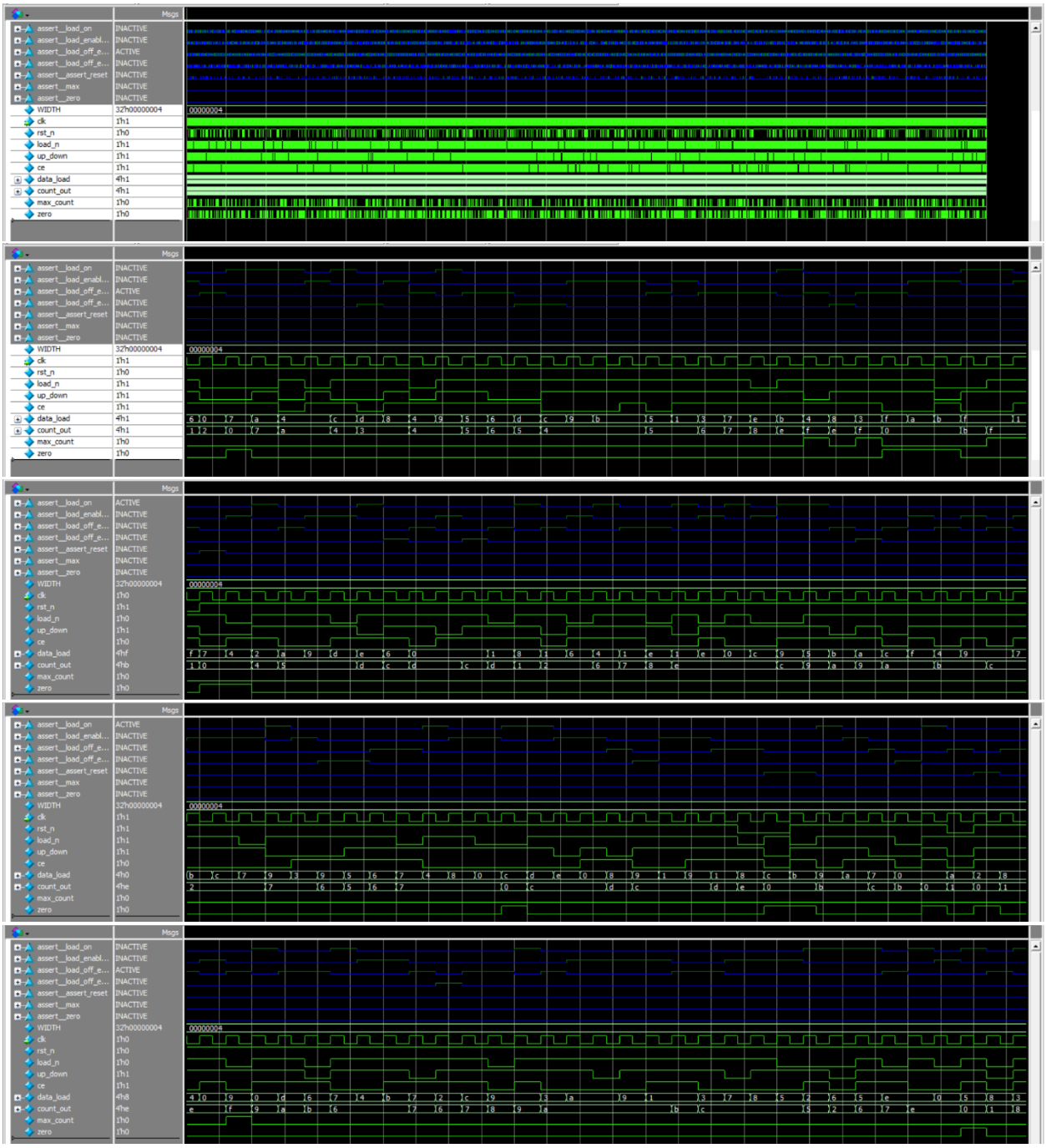
```
#
# Assertion Coverage:
#     Assertions                        7         7         0   100.00%
# --------------------------------------------------------------------
# Name                  File(Line)                    Failure      Pass
#                                                     Count        Count
# --------------------------------------------------------------------
# /counter_top/inst_dut/assertion_inst/assert__zero
#                       counter_sva.sv(36)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__max
#                       counter_sva.sv(35)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__assert_reset
#                       counter_sva.sv(34)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__load_off_enable_on_down
#                       counter_sva.sv(33)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__load_off_enable_on_up
#                       counter_sva.sv(32)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__load_enable_off
#                       counter_sva.sv(31)                  0          1
# /counter_top/inst_dut/assertion_inst/assert__load_on
#                       counter_sva.sv(30)                  0          1
#
# Directive Coverage:
#     Directives                        7         7         0   100.00%
#
```

```
# Directive Coverage:
#     Directives                        7         7         0   100.00%
#
# DIRECTIVE COVERAGE:
# ---------------------------------------------------------------------------------------------
# Name                                  Design Design   Lang File(Line)      Hits Status
#                                       Unit   UnitType
# ---------------------------------------------------------------------------------------------
# /counter_top/inst_dut/assertion_inst/cover__zero
#                                       counter_sva Verilog  SVA  counter_sva.sv(44)
#                                                                       1239 Covered
# /counter_top/inst_dut/assertion_inst/cover__max
#                                       counter_sva Verilog  SVA  counter_sva.sv(43)
#                                                                        636 Covered
# /counter_top/inst_dut/assertion_inst/cover__assert_reset
#                                       counter_sva Verilog  SVA  counter_sva.sv(42)
#                                                                        492 Covered
# /counter_top/inst_dut/assertion_inst/cover__load_off_enable_on_down
#                                       counter_sva Verilog  SVA  counter_sva.sv(41)
#                                                                       1366 Covered
# /counter_top/inst_dut/assertion_inst/cover__load_off_enable_on_up
#                                       counter_sva Verilog  SVA  counter_sva.sv(40)
#                                                                       3332 Covered
# /counter_top/inst_dut/assertion_inst/cover__load_enable_off
#                                       counter_sva Verilog  SVA  counter_sva.sv(39)
#                                                                       1983 Covered
# /counter_top/inst_dut/assertion_inst/cover__load_on
#                                       counter_sva Verilog  SVA  counter_sva.sv(38)
#                                                                       2825 Covered
#
```

```
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                        10        10         0   100.00%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /counter_top/inst_dut
# NOTE: The modification timestamp for source file 'counter.sv' has been altered since compilation.
#
#     Line         Item                     Count     Source
#     ----         ----                     -----     ------
#   File counter.sv
# ------------------------------------IF Branch------------------------------------
#     10                                   10000     Count coming in to IF
#     10             1                       492         if (!c_DUT.rst_n)
#
#     12             1                      2825         else if (!c_DUT.load_n)
#
#     14             1                      4699         else if (c_DUT.ce)
#
#                                           1984     All False Count
# Branch totals: 4 hits of 4 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     15                                    4699     Count coming in to IF
#     15             1                      3333              if (c_DUT.up_down)
#
#     17             1                      1366              else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     21                                    7780     Count coming in to IF
#     21             1                       499     assign c_DUT.max_count = (c_DUT.count_out == {c_DUT.WIDTH{1'b1}})? 1:0;
#
#     21             2                      7281     assign c_DUT.max_count = (c_DUT.count_out == {c_DUT.WIDTH{1'b1}})? 1:0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     22                                    7780     Count coming in to IF
#     22             1                       881     assign c_DUT.zero = (c_DUT.count_out == 0)? 1:0;
#
#     22             2                      6899     assign c_DUT.zero = (c_DUT.count_out == 0)? 1:0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
#
# Condition Coverage:
#     Enabled Coverage              Bins   Covered    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Conditions                       2         2         0   100.00%
#
# ===============================Condition Details================================
#
# Condition Coverage for instance /counter_top/inst_dut --
# NOTE: The modification timestamp for source file 'counter.sv' has been altered since compilation.
#
#   File counter.sv
# ----------------Focused Condition View-------------------
# Line       21 Item    1  (c_DUT.count_out == {c_DUT.WIDTH{1}})
# Condition totals: 1 of 1 input term covered = 100.00%
#
#                            Input Term   Covered  Reason for no coverage   Hint
#                           -----------  --------  ----------------------  --------------
#   (c_DUT.count_out == {c_DUT.WIDTH{1}})         Y
#
#      Rows:       Hits  FEC Target                                Non-masking condition(s)
#  ---------  ---------  --------------------                      -------------------------
#   Row   1:          1  (c_DUT.count_out == {c_DUT.WIDTH{1}})_0  -
#   Row   2:          1  (c_DUT.count_out == {c_DUT.WIDTH{1}})_1  -
#
# ----------------Focused Condition View-------------------
# Line       22 Item    1  (c_DUT.count_out == 0)
# Condition totals: 1 of 1 input term covered = 100.00%
#
#              Input Term   Covered  Reason for no coverage   Hint
#             -----------  --------  ----------------------  --------------
#   (c_DUT.count_out == 0)         Y
#
#      Rows:       Hits  FEC Target               Non-masking condition(s)
#  ---------  ---------  --------------------     -------------------------
#   Row   1:          1  (c_DUT.count_out == 0)_0  -
#   Row   2:          1  (c_DUT.count_out == 0)_1  -
#
#
# Statement Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Statements                       7         7         0   100.00%
#
```

```
# ================================Statement Details================================
#
# Statement Coverage for instance /counter_top/inst_dut --
# NOTE: The modification timestamp for source file 'counter.sv' has been altered since compilation.
#
#    Line         Item                    Count     Source
#    ----         ----                    -----     ------
#  File counter.sv
#    8                                              module counter_d (counter_if.DUT c_DUT);
#
#    9              1                    10000     always @(posedge c_DUT.clk) begin
#
#    10                                                 if (!c_DUT.rst_n)
#
#    11             1                      492             c_DUT.count_out <= 0;
#
#    12                                                 else if (!c_DUT.load_n)
#
#    13             1                     2825             c_DUT.count_out <= c_DUT.data_load;
#
#    14                                                 else if (c_DUT.ce)
#
#    15                                                     if (c_DUT.up_down)
#
#    16             1                     3333                 c_DUT.count_out <= c_DUT.count_out + 1;
#
#    17                                                     else
#
#    18             1                     1366                 c_DUT.count_out <= c_DUT.count_out - 1;
#
#    19                                             end
#
#    20
#
#    21             1                     7781     assign c_DUT.max_count = (c_DUT.count_out == {c_DUT.WIDTH{1'b1}})? 1:0;
#
#    22             1                     7781     assign c_DUT.zero = (c_DUT.count_out == 0)? 1:0;
#
#
#
#
#
# ================================================================================
# === Instance: /counter_top/inst_test
# === Design Unit: work.counter_tb
# ================================================================================
#
# Assertion Coverage:
#     Assertions                       1         1         0   100.00%
# --------------------------------------------------------------------
# Name                 File(Line)                    Failure      Pass
#                                                    Count        Count
# --------------------------------------------------------------------
# /counter_top/inst_test/#ublk#95084642#20/immed__22
#                      counter_tb.sv(22)                   0          1
# Statement Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Statements                      18        18         0   100.00%
#
```

12. Questasim snippets



P4)

1. Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
TEST_1	When the **reset** is asserted, the registers should take the default values 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	Directed at the start of the simulation		A checker in the testbench to make sure each register take the default value
TEST_2	When **write** signal is high the input **data_in** is stored in the register of input **address**	Randomized input **data_in** and **address** to take value from (0) to (7)		A checker in the testbench to make sure the data stored correctly
TEST_3	When **read** signal is high the output **data_out** should be equal to data stored in **data_assoc** array	Randomized input **address** to take value from (0) to (7)		A checker in the testbench to make sure the data stored amd readed correctly
TEST_4	When the **reset** is asserted, the registers should take the default values 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	Directed at the end of the simulation		A checker in the testbench to make sure each register take the default value

2. Package code

```
package reg_buggy_pkg;
    typedef enum logic [3:0] { adc0_reg ,adc1_reg ,temp_sensor0_reg ,temp_sensor1_reg ,
                        analog_test ,digital_test ,amp_gain ,digital_config } address_t;
    class reg_buggy_class;
        rand logic reset ;
        rand logic [7:0] [15:0] data_in ;
        rand address_t address_pkg ;
        rand address_t [8] address_arr;

        constraint reset_t {
            reset dist {0:/99 , 1:/1} ;
        }

        constraint address_datain  {
            foreach (address_arr[i])
                address_arr[i] inside {adc0_reg ,adc1_reg ,temp_sensor0_reg ,temp_sensor1_reg ,
                        analog_test ,digital_test ,amp_gain ,digital_config};

            foreach (address_arr[i])
                foreach (address_arr[j])
                  if (i != j) address_arr[i] != address_arr[j];

            foreach (data_in[j])
                data_in [j] dist {[16'h0 : 16'hffff] } ;
        }
    endclass
endpackage
```

3. Testbench code

```
import reg_buggy_pkg::*;
module reg_buggy_tb();
    reg clk ;
    reg reset;
    reg write;
    reg [15:0] data_in;
    wire [15:0] data_out;
    reg [15:0] data_out_expected;

    address_t address_tb ;
    logic [15:0] reset_assoc[string] ;
    logic [15:0] data_assoc[int] ;

    config_reg DUT (clk,reset,write,data_in,address_tb,data_out) ;

    integer correct_count = 0;
    integer error_count   = 0;

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

    initial begin
        reg_buggy_class obj = new ;
        reset = 0 ;
        $display("==============testing reset 1=================");
        assert_reset() ;
        repeat (3) begin
            write = 1 ;
            assert (obj.randomize())
            for (int i = 0 ; i < 8 ; i++) begin
                @(negedge clk);
                address_tb = obj.address_arr[i] ;
                data_in = obj.data_in[i] ;
                golden_model();
            end
            $display("==============checking reading from meomery==============");
            write = 0 ;
            assert (obj.randomize())
            for (int i = 0 ; i < 8 ; i++) begin
                @(negedge clk);
                address_tb = obj.address_arr[i];
                golden_model();
                check_result();
            end
        end

        $display("==================testing reset 2==================");
        assert_reset();

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

    task golden_model ();
        if (reset) begin
            reset_assoc [adc0_reg] = 16'hffff;
            reset_assoc [adc1_reg] = 16'h0;
            reset_assoc [temp_sensor0_reg] = 16'h0;
            reset_assoc [temp_sensor1_reg] = 16'h0;
            reset_assoc [analog_test] = 16'habcd;
            reset_assoc [digital_test] = 16'h0;
            reset_assoc [amp_gain] = 16'h0;
            reset_assoc [digital_config] = 16'h1;
        end
        else begin
            if (write) begin
                data_assoc [address_tb] = data_in ;
            end
            else begin
                data_out_expected = data_assoc [address_tb] ;
            end
        end
    endtask

    task assert_reset();
        reset = 1 ;
        golden_model() ;
        check_reset() ;
        reset = 0 ;
    endtask

    task check_result();
        @(negedge clk) ;
        if (data_out_expected == data_out) begin
            correct_count = correct_count + 1 ;
            $display("test passes");
        end
        else begin
           error_count = error_count + 1 ;
            $display("test %d fails expected %h but output is %h " ,address_tb ,data_out_expected , data_out) ;
        end
    endtask

    task check_reset();
        for (address_tb = address_tb.first(); address_tb != address_tb.last();address_tb = address_tb.next()) begin
            @(negedge clk) ;
            if (reset_assoc[address_tb] == data_out) begin
                correct_count = correct_count + 1 ;
                $display("test %d passes " , address_tb);
            end
            else begin
                error_count = error_count + 1 ;
                $display("test %d fails expected %h bit output is %h " , address_tb ,reset_assoc[address_tb] , data_out);
            end
        end
        @(negedge clk) ;
        if (reset_assoc[address_tb] == data_out) begin
            correct_count = correct_count + 1 ;
            $display("test %d passes " , address_tb);
        end
        else begin
            error_count = error_count + 1 ;
            $display("test %d fails expected %h bit output is %h " , address_tb ,reset_assoc[address_tb] , data_out);
        end
    endtask

endmodule
```

4. Complete bug report

Bug 1 :

a) **Design Input for Bug to Appear:**
When reset is high in the begin of simulation → write **16'hABCD** in register **analog_test**

b) **Expected Behavior:**
When check the value of register **analog_test** after reset to be **16'hABCD**

c) **Observed Behavior:**
After reset register **analog_test** has value **16'hABCC**

Bug 2 :

a) **Design Input for Bug to Appear:**
When write is high → input to register **adc1_reg** is **16'hDC9E**

b) **Expected Behavior:**
When read is high →the output from register **adc1_reg** is expected to be **16'hDC9E**

c) **Observed Behavior:**
The observed output from register **adc1_reg** is **16'h9EDC**

Bug 3 :

a) **Design Input for Bug to Appear:**
When write is high → input to register **digital_test** is **16'h9E4C**

b) **Expected Behavior:**
When read is high →the output from register **digital_test** is expected to be **16'h9E4C**

c) **Observed Behavior:**
The observed output from register **digital_test** is **16'h703B**

Bug 4 :

a) **Design Input for Bug to Appear:**
When write is high → input to register **amp_gain** is **16'h703B**

b) **Expected Behavior:**
When read is high →the output from register **amp_gain** is expected to be **16'h703B**

c) **Observed Behavior:**
The observed output from register **amp_gain** is **16'hE94C**

Bug 5 :

a) **Design Input for Bug to Appear:**
When write is high → input to register **adc0_reg** is **16'hE32F**

b) **Expected Behavior:**
When read is high →the output from register **adc0_reg** is expected to be **16'hE32F**

c) **Observed Behavior:**
The observed output from register **adc0_reg** is **16'hFFFF**

Bug 6 :

a) **Design Input for Bug to Appear:**

When write is high → input to register **temp_sensor0** is **16'h763D**

b) **Expected Behavior:**

When read is high →the output from register **temp_sensor0** is expected to be **16'h763D**

c) **Observed Behavior:**

The observed output from register **temp_sensor0** is **16'hFFFF**

Bug 7 :

a) **Design Input for Bug to Appear:**

When reset is high again → write **16'h0** in register **temp_sensor1_reg**

b) **Expected Behavior:**

When check the value of register **temp_sensor1_reg** after reset to be **16'h0**

c) **Observed Behavior:**

After reset register **temp_sensor1_reg** has value **16'h3EF0**

Bug 8 :

a) **Design Input for Bug to Appear:**

When reset is high in the begin of simulation → write **16'hABCD** in register **analog_test**

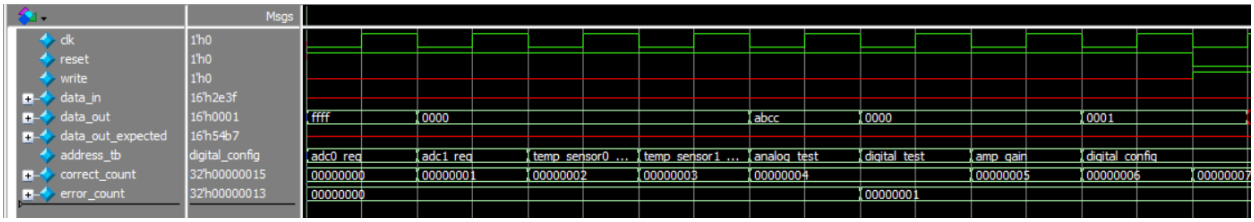
b) **Expected Behavior:**

When check the value of register **analog_test** after reset to be **16'hABCD**

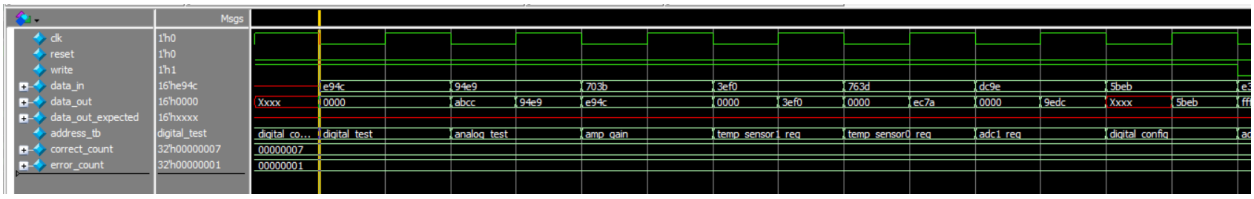
c) **Observed Behavior:**

After reset register **analog_test** has value **16'hABCC**

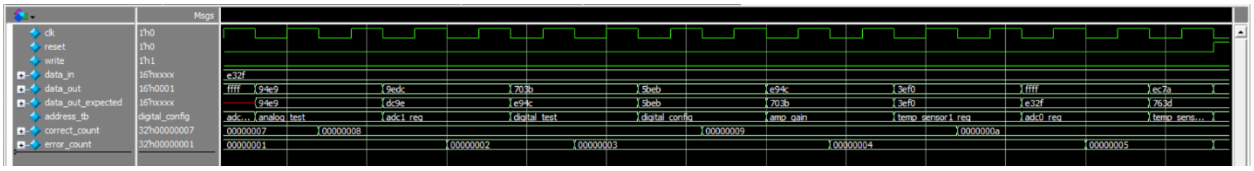
5. Separate snippets taken from QuestaSim for each bugs detected



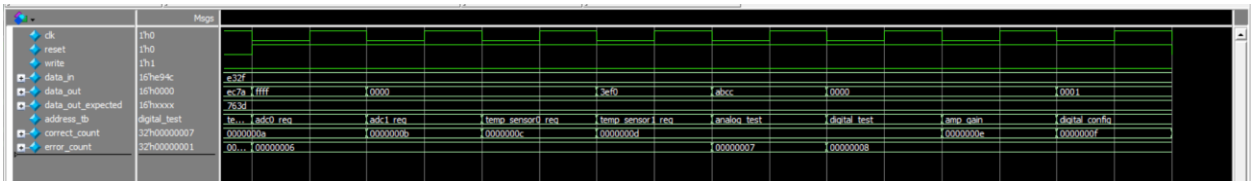
Bugs in first reset



Entered data in 8 registers



Bugs found in reading data



Bugs found in second reset