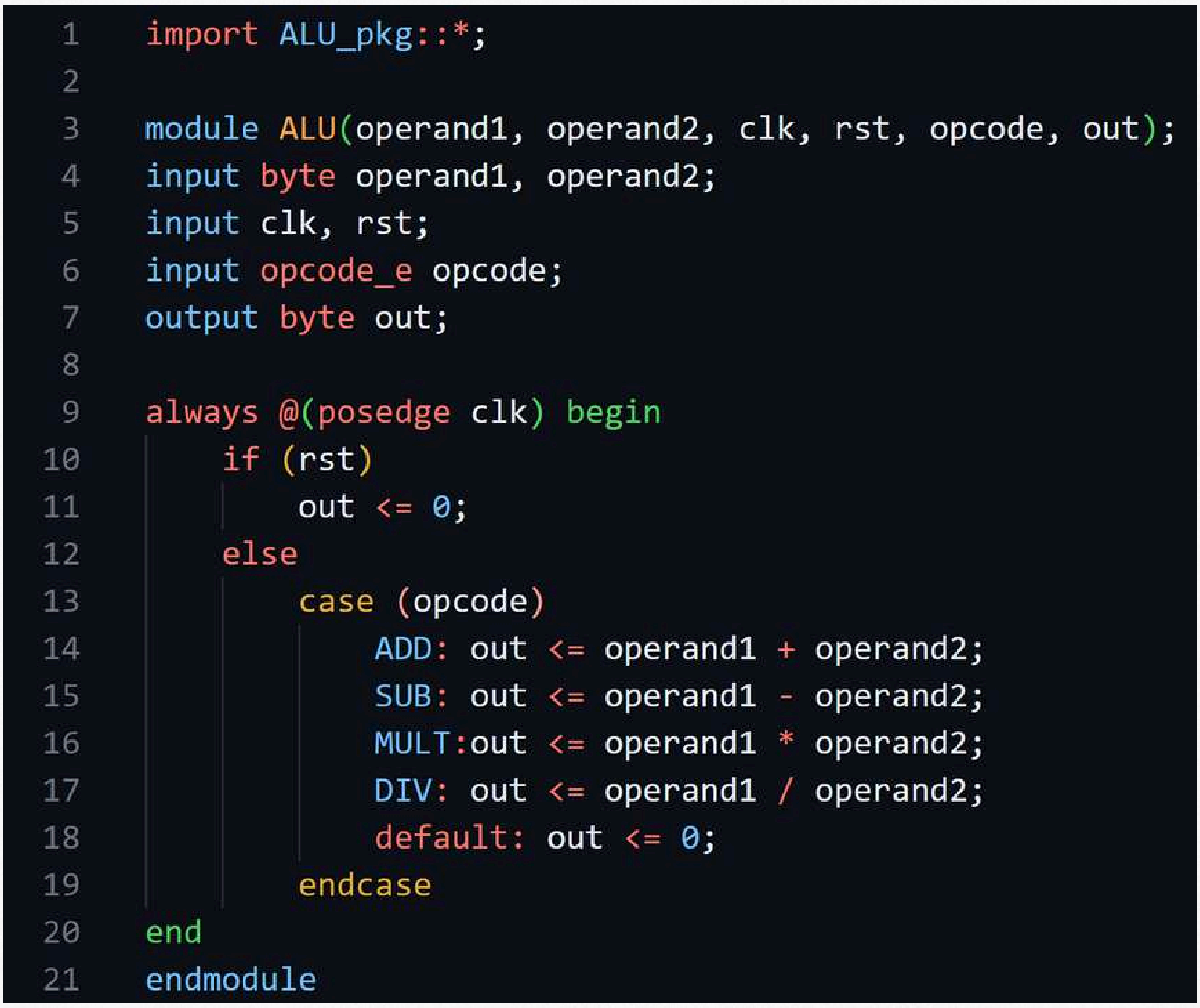
يوسف أحمد محمد ابراهيم

Assignment 3

Question 1:

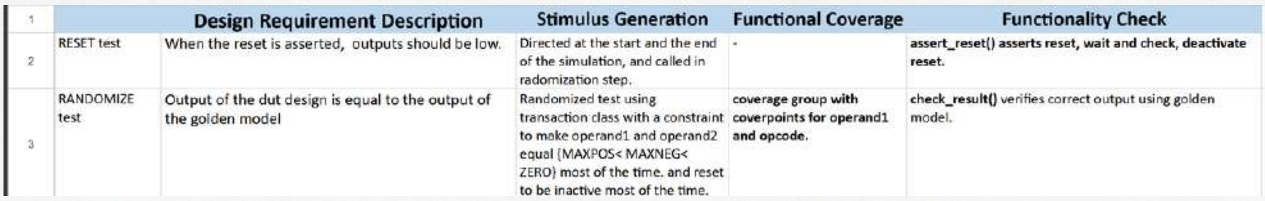
1. RTL design:



```
import ALU_pkg::*;

module ALU(operand1, operand2, clk, rst, opcode, out);
input byte operand1, operand2;
input clk, rst;
input opcode_e opcode;
output byte out;

always @(posedge clk) begin
    if (rst)
        out <= 0;
    else
        case (opcode)
            ADD: out <= operand1 + operand2;
            SUB: out <= operand1 - operand2;
            MULT:out <= operand1 * operand2;
            DIV: out <= operand1 / operand2;
            default: out <= 0;
        endcase
end
endmodule
```

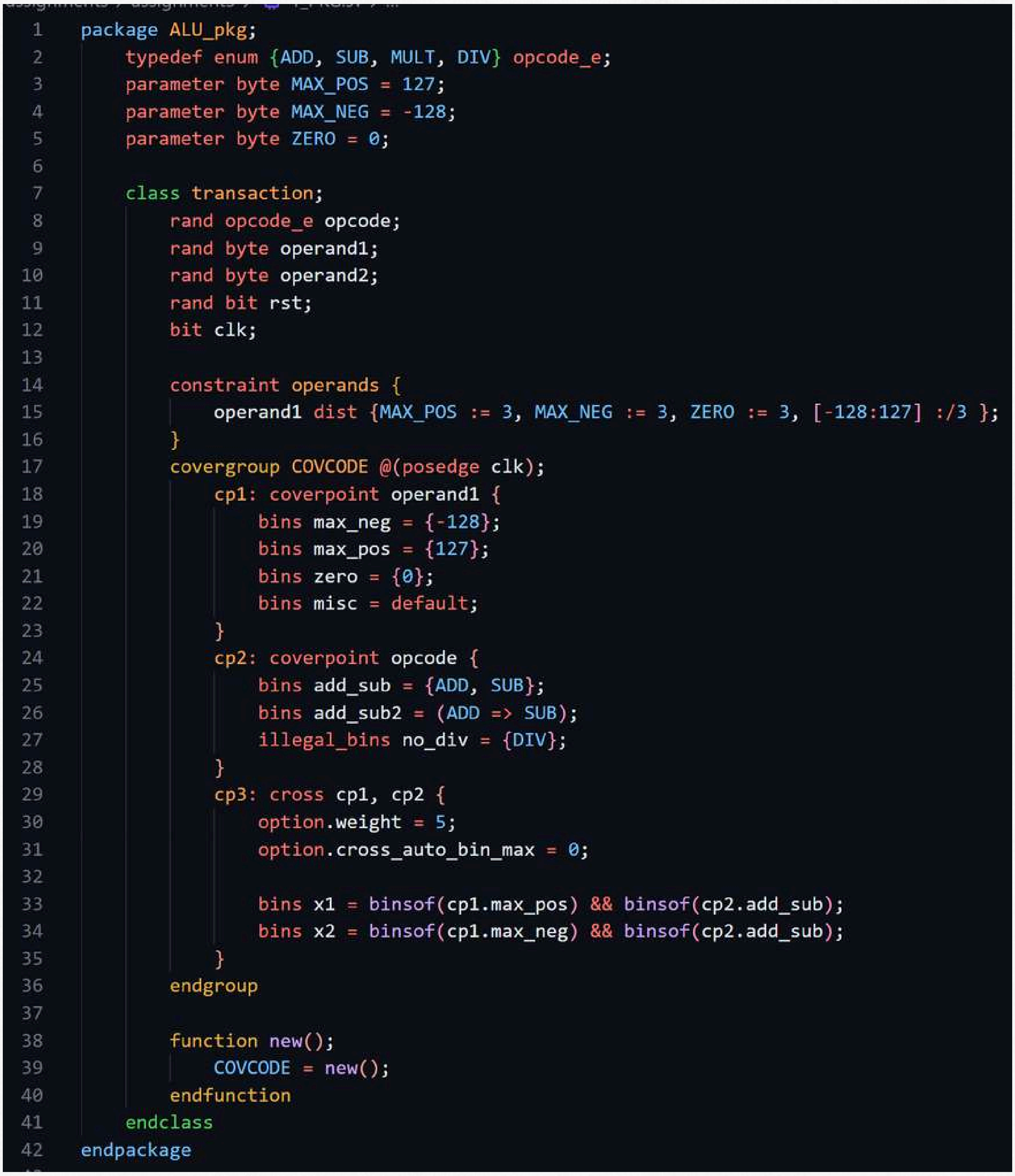
The design is working properly.

2. Verification plan:



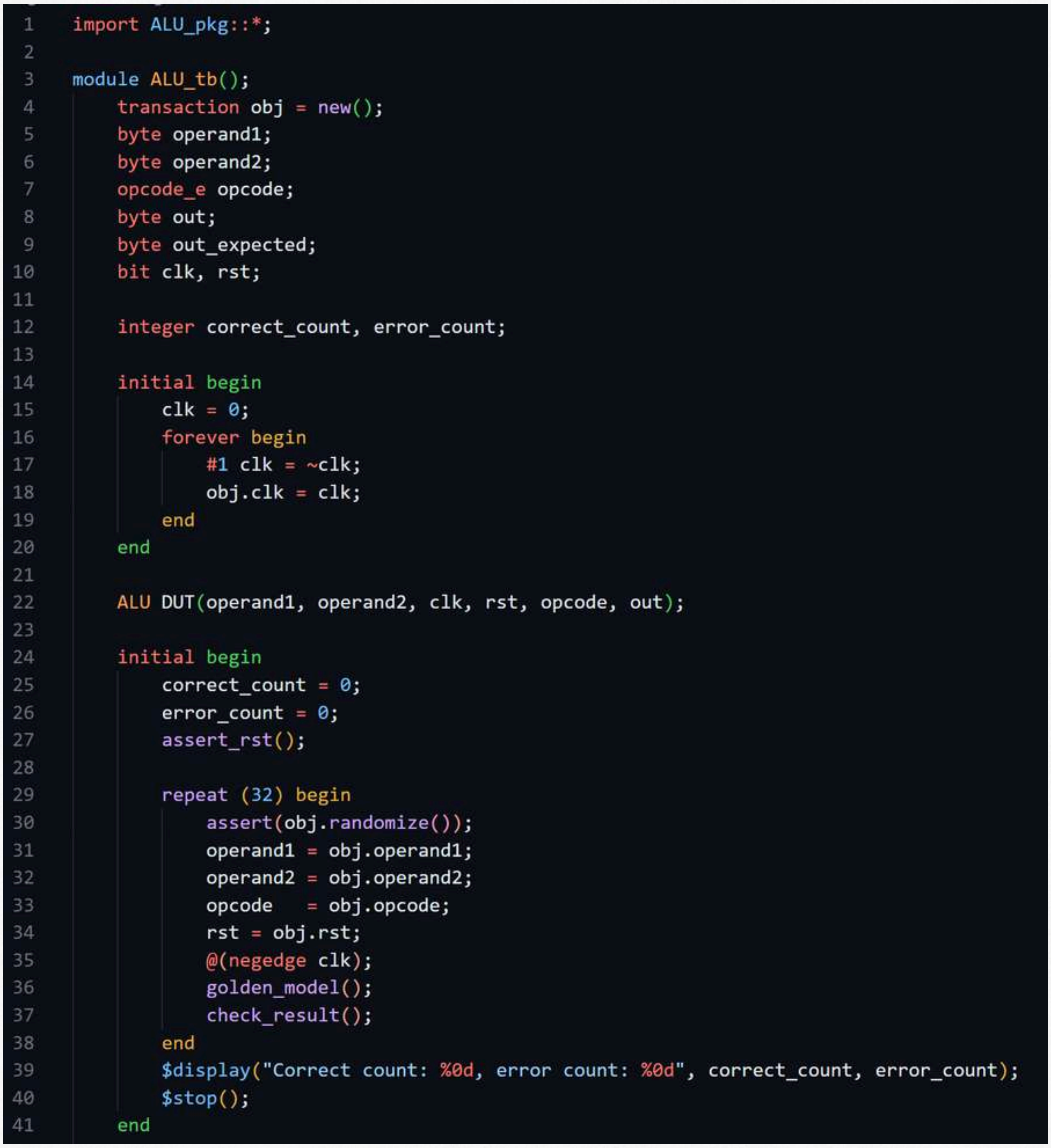
	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET test	When the reset is asserted, outputs should be low.	Directed at the start and the end of the simulation, and called in radomization step.	-	**assert_reset() asserts reset, wait and check, deactivate reset.**
RANDOMIZE test	Output of the dut design is equal to the output of the golden model	Randomized test using transaction class with a constraint to make operand1 and operand2 equal {MAXPOS< MAXNEG< ZERO} most of the time. and reset to be inactive most of the time.	**coverage group with coverpoints for operand1 and opcode.**	**check_result()** verifies correct output using golden model.

3. package code:



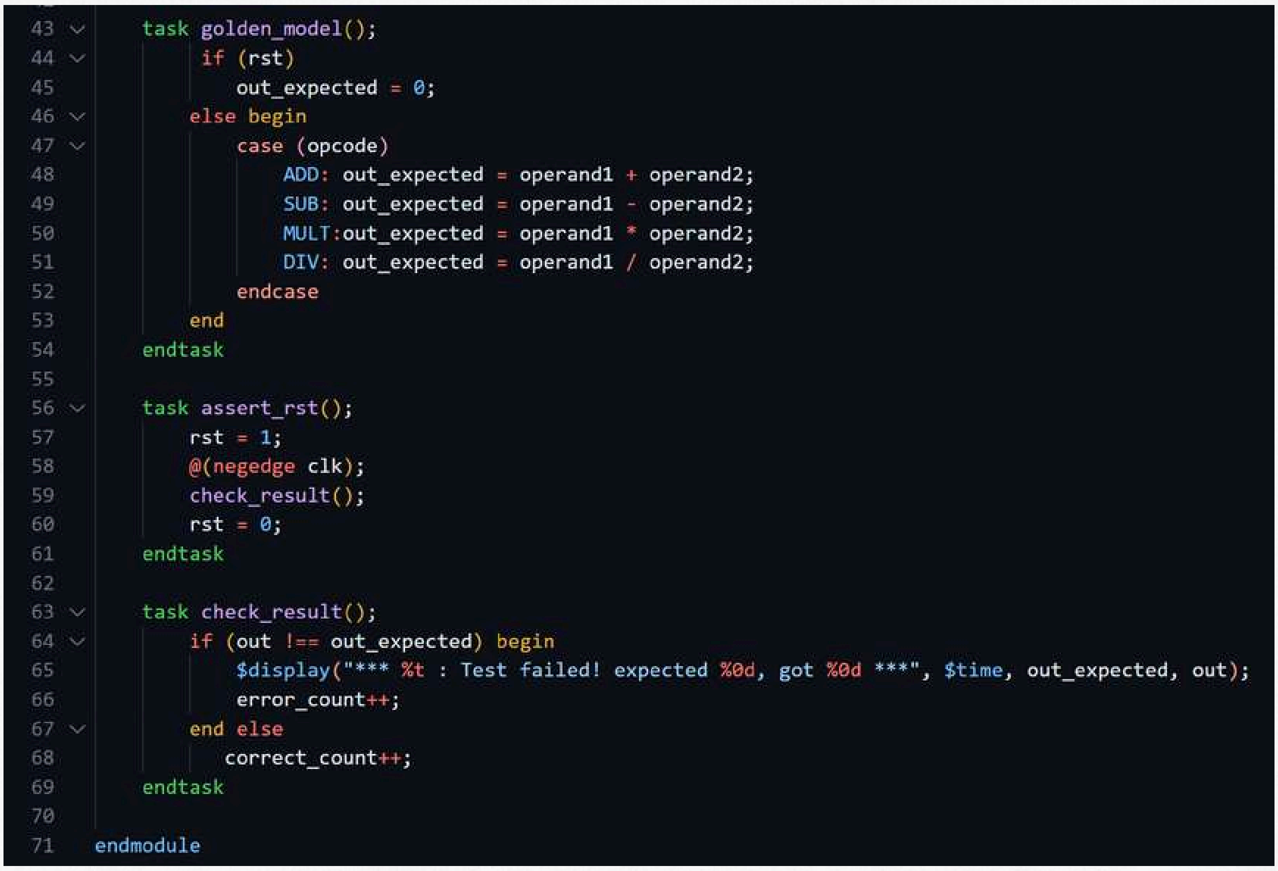
```systemverilog
package ALU_pkg;
    typedef enum {ADD, SUB, MULT, DIV} opcode_e;
    parameter byte MAX_POS = 127;
    parameter byte MAX_NEG = -128;
    parameter byte ZERO = 0;

    class transaction;
        rand opcode_e opcode;
        rand byte operand1;
        rand byte operand2;
        rand bit rst;
        bit clk;

        constraint operands {
            operand1 dist {MAX_POS := 3, MAX_NEG := 3, ZERO := 3, [-128:127] :/3 };
        }
        covergroup COVCODE @(posedge clk);
            cp1: coverpoint operand1 {
                bins max_neg = {-128};
                bins max_pos = {127};
                bins zero = {0};
                bins misc = default;
            }
            cp2: coverpoint opcode {
                bins add_sub = {ADD, SUB};
                bins add_sub2 = (ADD => SUB);
                illegal_bins no_div = {DIV};
            }
            cp3: cross cp1, cp2 {
                option.weight = 5;
                option.cross_auto_bin_max = 0;

                bins x1 = binsof(cp1.max_pos) && binsof(cp2.add_sub);
                bins x2 = binsof(cp1.max_neg) && binsof(cp2.add_sub);
            }
        endgroup

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

4. Testbench code:



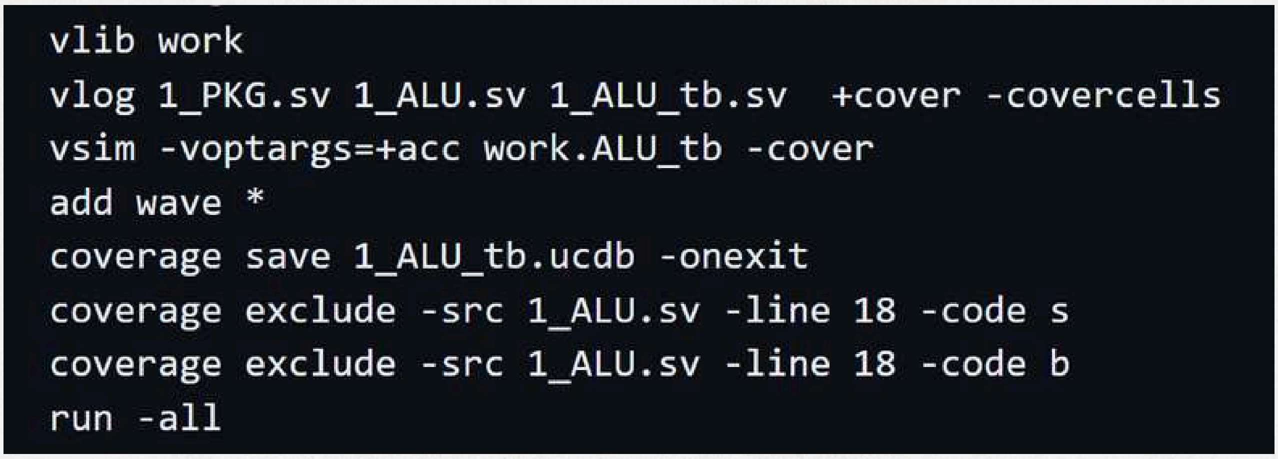
```
import ALU_pkg::*;

module ALU_tb();
    transaction obj = new();
    byte operand1;
    byte operand2;
    opcode_e opcode;
    byte out;
    byte out_expected;
    bit clk, rst;

    integer correct_count, error_count;

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

    ALU DUT(operand1, operand2, clk, rst, opcode, out);

    initial begin
        correct_count = 0;
        error_count = 0;
        assert_rst();

        repeat (32) begin
            assert(obj.randomize());
            operand1 = obj.operand1;
            operand2 = obj.operand2;
            opcode   = obj.opcode;
            rst = obj.rst;
            @(negedge clk);
            golden_model();
            check_result();
        end
        $display("Correct count: %0d, error count: %0d", correct_count, error_count);
        $stop();
    end
```

4. Testbench code:



```
    task golden_model();
         if (rst)
            out_expected = 0;
        else begin
            case (opcode)
                ADD: out_expected = operand1 + operand2;
                SUB: out_expected = operand1 - operand2;
                MULT:out_expected = operand1 * operand2;
                DIV: out_expected = operand1 / operand2;
            endcase
        end
    endtask

    task assert_rst();
        rst = 1;
        @(negedge clk);
        check_result();
        rst = 0;
    endtask

    task check_result();
        if (out !== out_expected) begin
            $display("*** %t : Test failed! expected %0d, got %0d ***", $time, out_expected, out);
            error_count++;
        end else
           correct_count++;
    endtask

endmodule
```

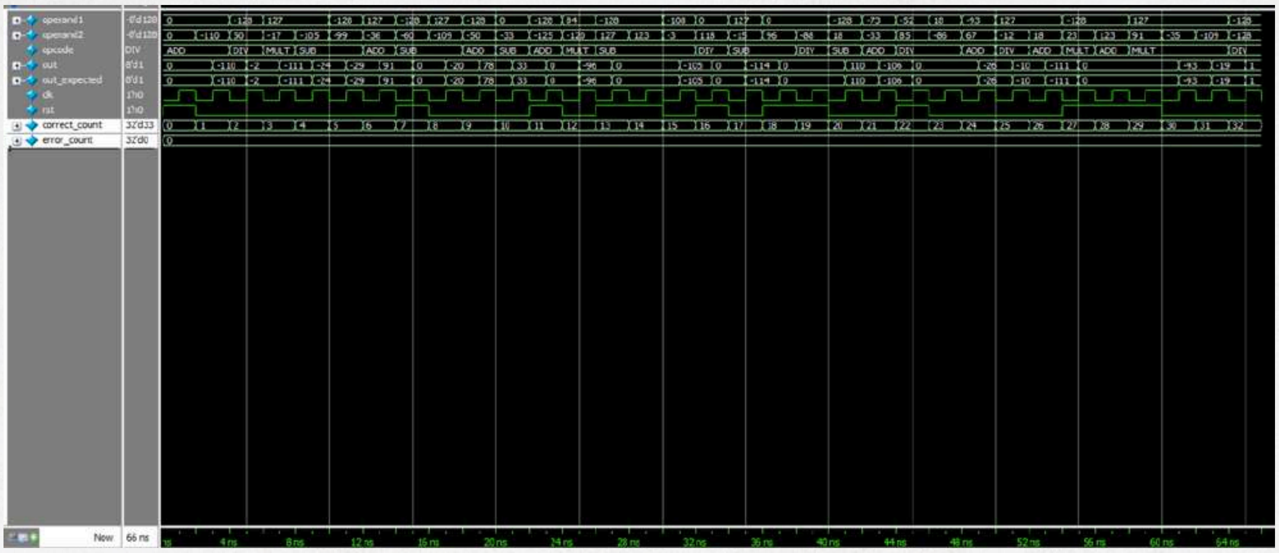
5. Do file :

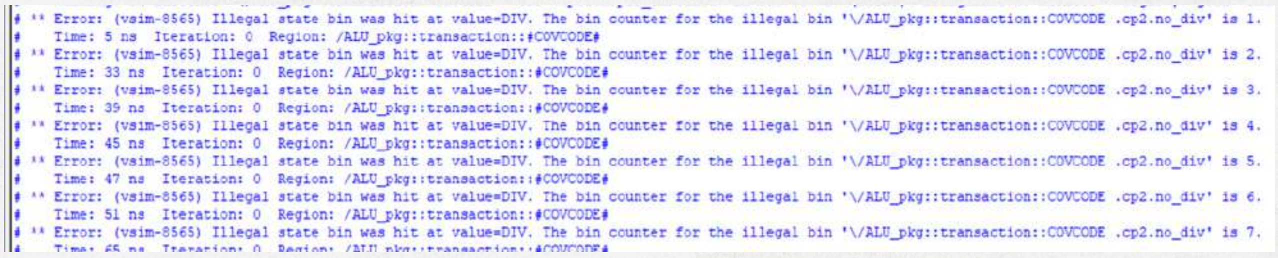


```
vlib work
vlog 1_PKG.sv 1_ALU.sv 1_ALU_tb.sv  +cover -covercells
vsim -voptargs=+acc work.ALU_tb -cover
add wave *
coverage save 1_ALU_tb.ucdb -onexit
coverage exclude -src 1_ALU.sv -line 18 -code s
coverage exclude -src 1_ALU.sv -line 18 -code b
run -all
```

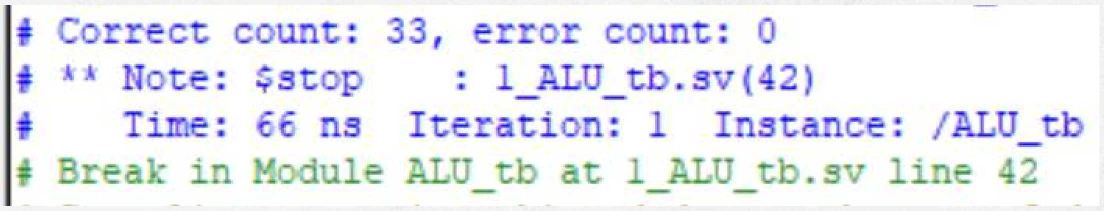
Excluded default case as it will never occur.

6. Qesta sim wave snippets (decimal radix)



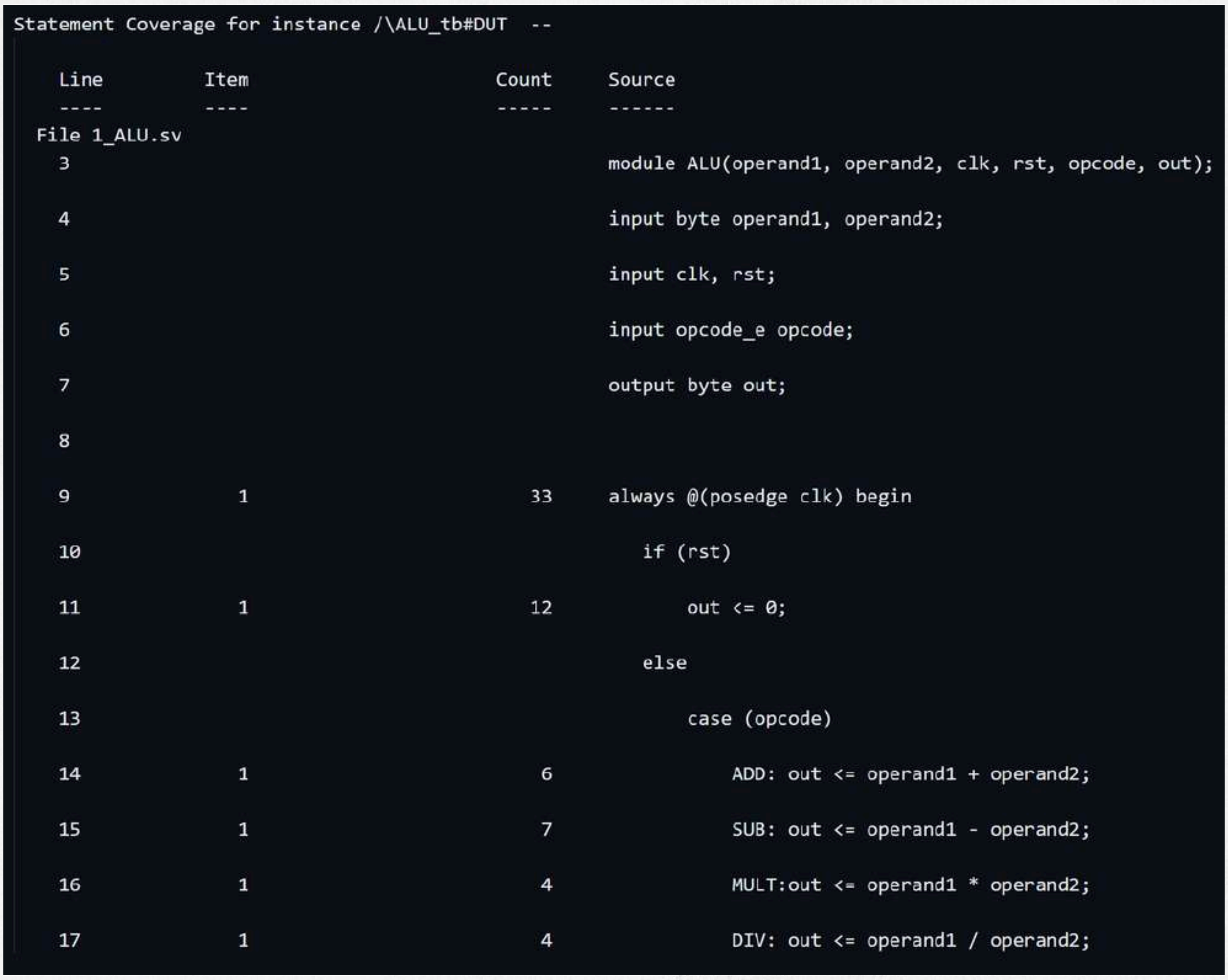


```
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 1.
#    Time: 5 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 2.
#    Time: 33 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 3.
#    Time: 39 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 4.
#    Time: 45 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 5.
#    Time: 47 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 6.
#    Time: 51 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
# ** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/ALU_pkg::transaction::COVCODE .cp2.no_div' is 7.
#    Time: 65 ns  Iteration: 0  Region: /ALU_pkg::transaction::#COVCODE#
```



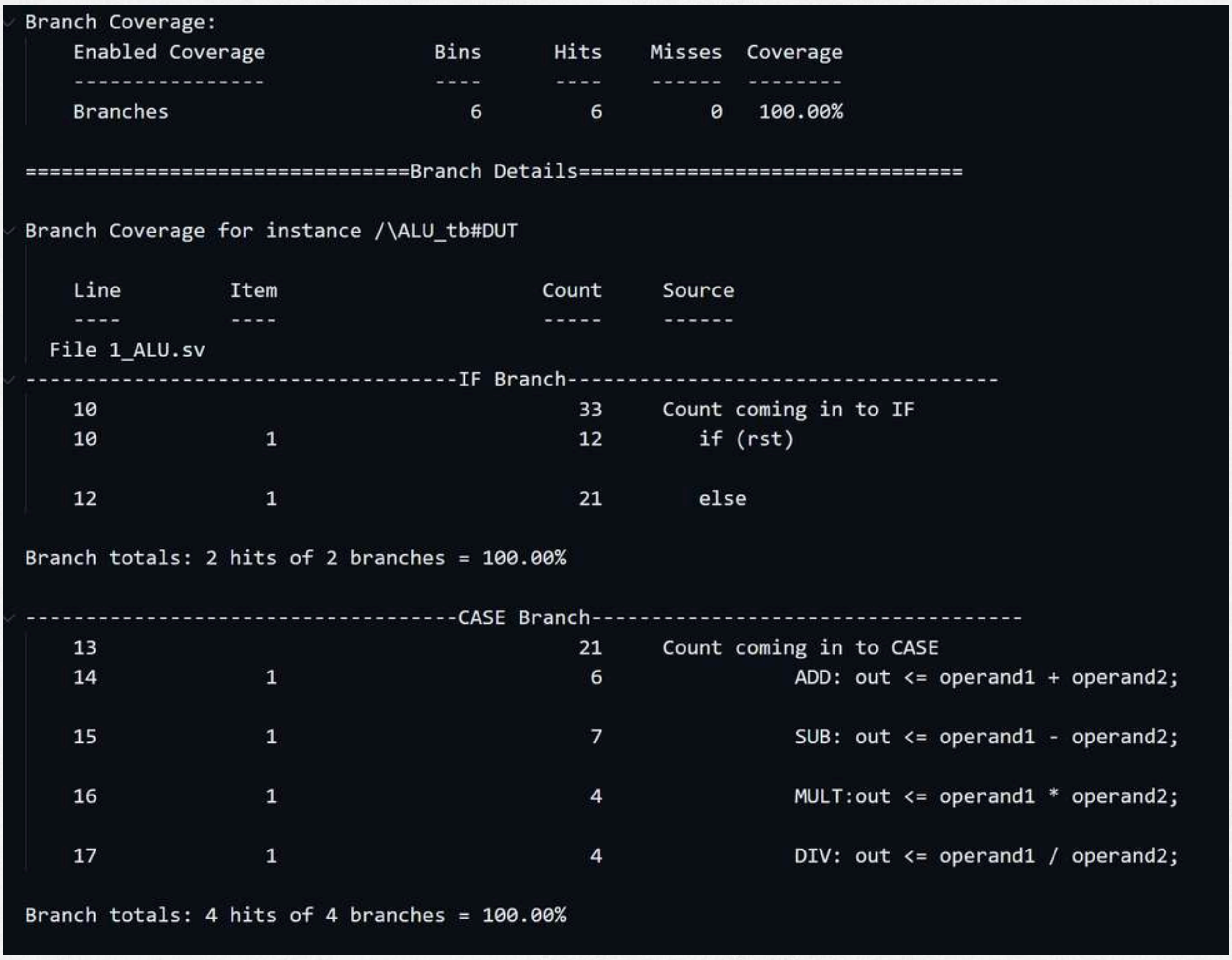
```
# Correct count: 33, error count: 0
# ** Note: $stop    : 1_ALU_tb.sv(42)
#    Time: 66 ns  Iteration: 1  Instance: /ALU_tb
# Break in Module ALU_tb at 1_ALU_tb.sv line 42
```

7. Statement coverage report :



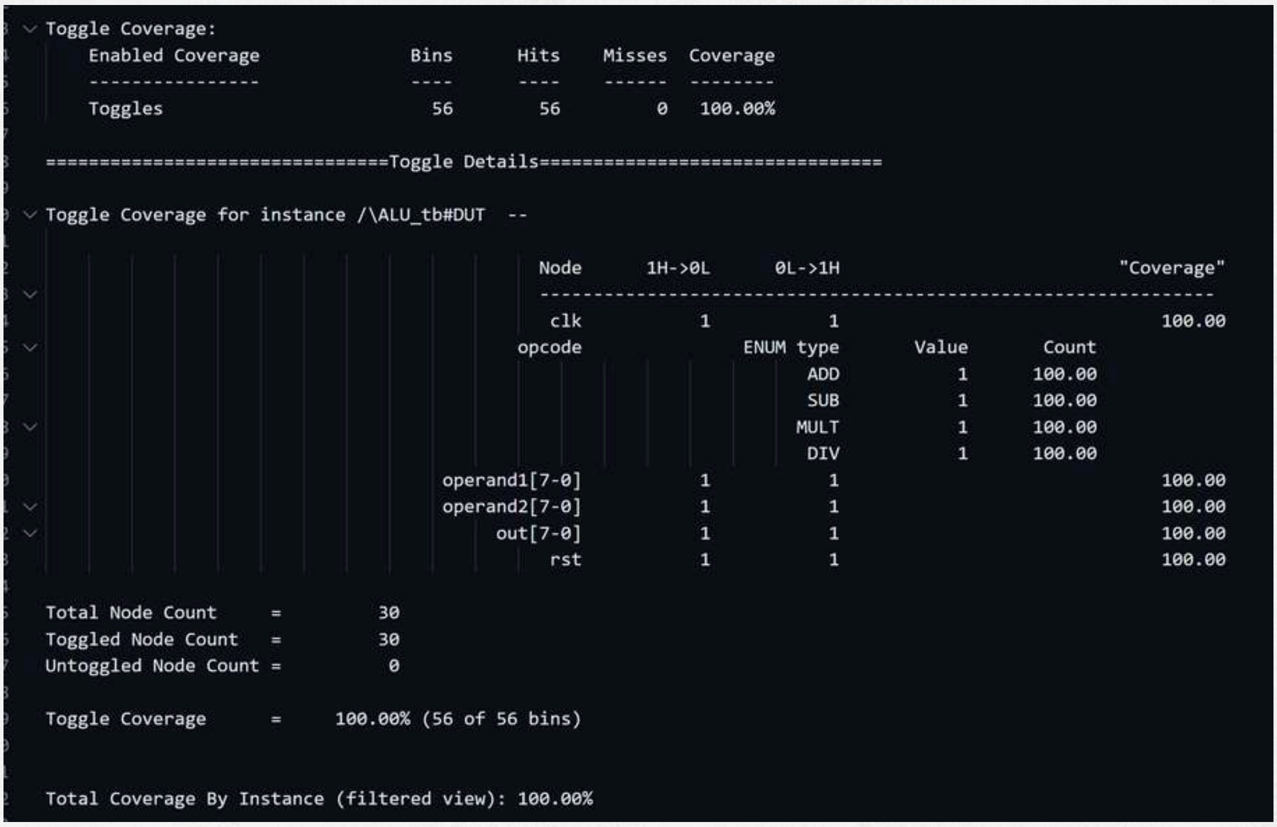
```
Statement Coverage for instance /\ALU_tb#DUT  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 1_ALU.sv
    3                                                module ALU(operand1, operand2, clk, rst, opcode, out);

    4                                                input byte operand1, operand2;

    5                                                input clk, rst;

    6                                                input opcode_e opcode;

    7                                                output byte out;

    8

    9               1                         33     always @(posedge clk) begin

    10                                                  if (rst)

    11              1                         12            out <= 0;

    12                                                  else

    13                                                      case (opcode)

    14              1                          6                ADD: out <= operand1 + operand2;

    15              1                          7                SUB: out <= operand1 - operand2;

    16              1                          4                MULT:out <= operand1 * operand2;

    17              1                          4                DIV: out <= operand1 / operand2;
```

8. Branch coverage report :



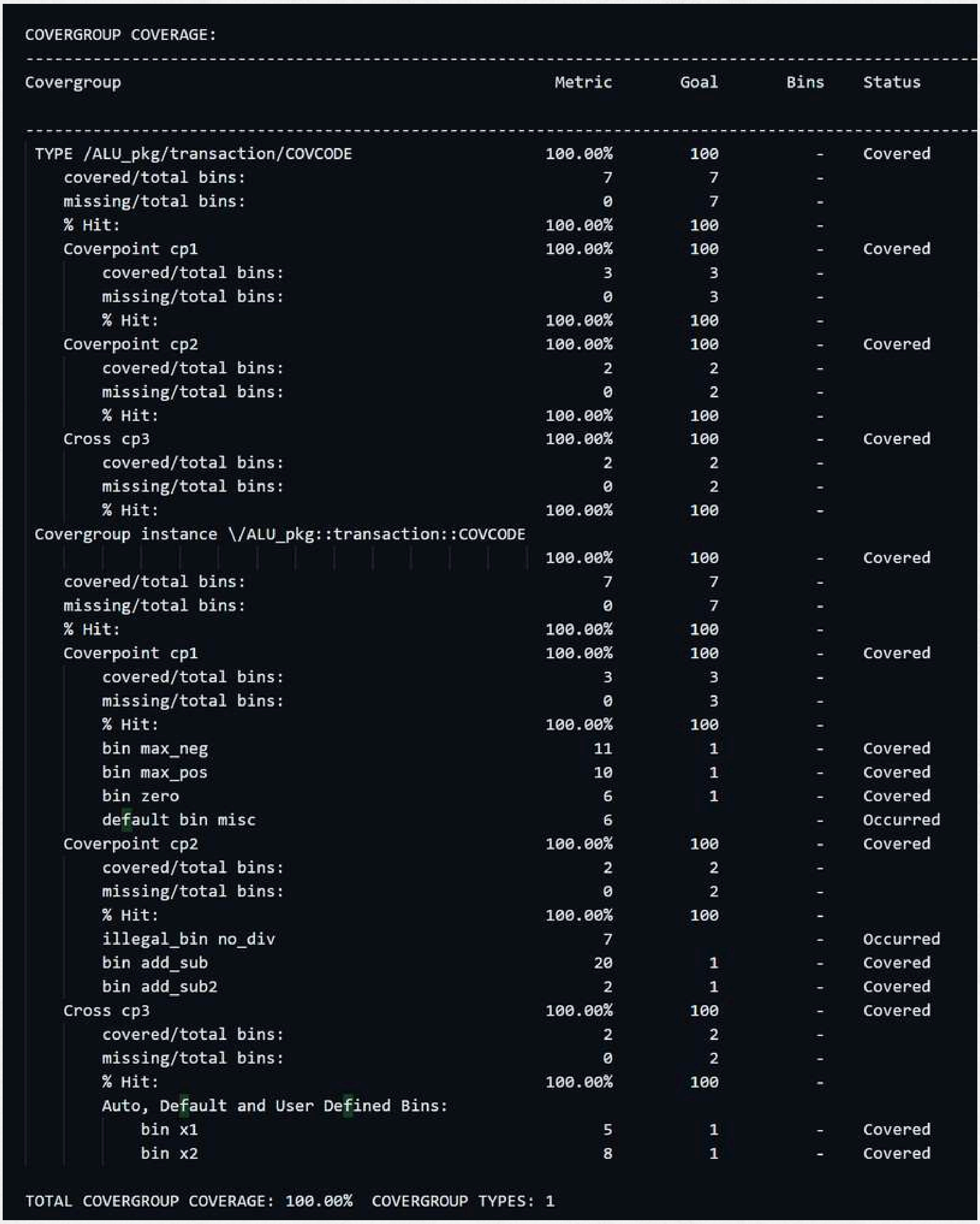
```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         6         6         0   100.00%

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

Branch Coverage for instance /\ALU_tb#DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 1_ALU.sv
------------------------------------IF Branch------------------------------------
    10                                        33     Count coming in to IF
    10              1                         12        if (rst)

    12              1                         21        else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    13                                        21     Count coming in to CASE
    14              1                          6               ADD: out <= operand1 + operand2;

    15              1                          7               SUB: out <= operand1 - operand2;

    16              1                          4               MULT:out <= operand1 * operand2;

    17              1                          4               DIV: out <= operand1 / operand2;

Branch totals: 4 hits of 4 branches = 100.00%
```

9. Toggle coverage report :



```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         56        56         0   100.00%

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

Toggle Coverage for instance /\ALU_tb#DUT  --

                                              Node      1H->0L      0L->1H                          "Coverage"
                                              ---------------------------------------------------------------
                                               clk           1           1                              100.00
                                            opcode                ENUM type       Value       Count
                                                                        ADD           1      100.00
                                                                        SUB           1      100.00
                                                                       MULT           1      100.00
                                                                        DIV           1      100.00
                                     operand1[7-0]           1           1                              100.00
                                     operand2[7-0]           1           1                              100.00
                                          out[7-0]           1           1                              100.00
                                               rst           1           1                              100.00

Total Node Count     =         30
Toggled Node Count   =         30
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (56 of 56 bins)


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

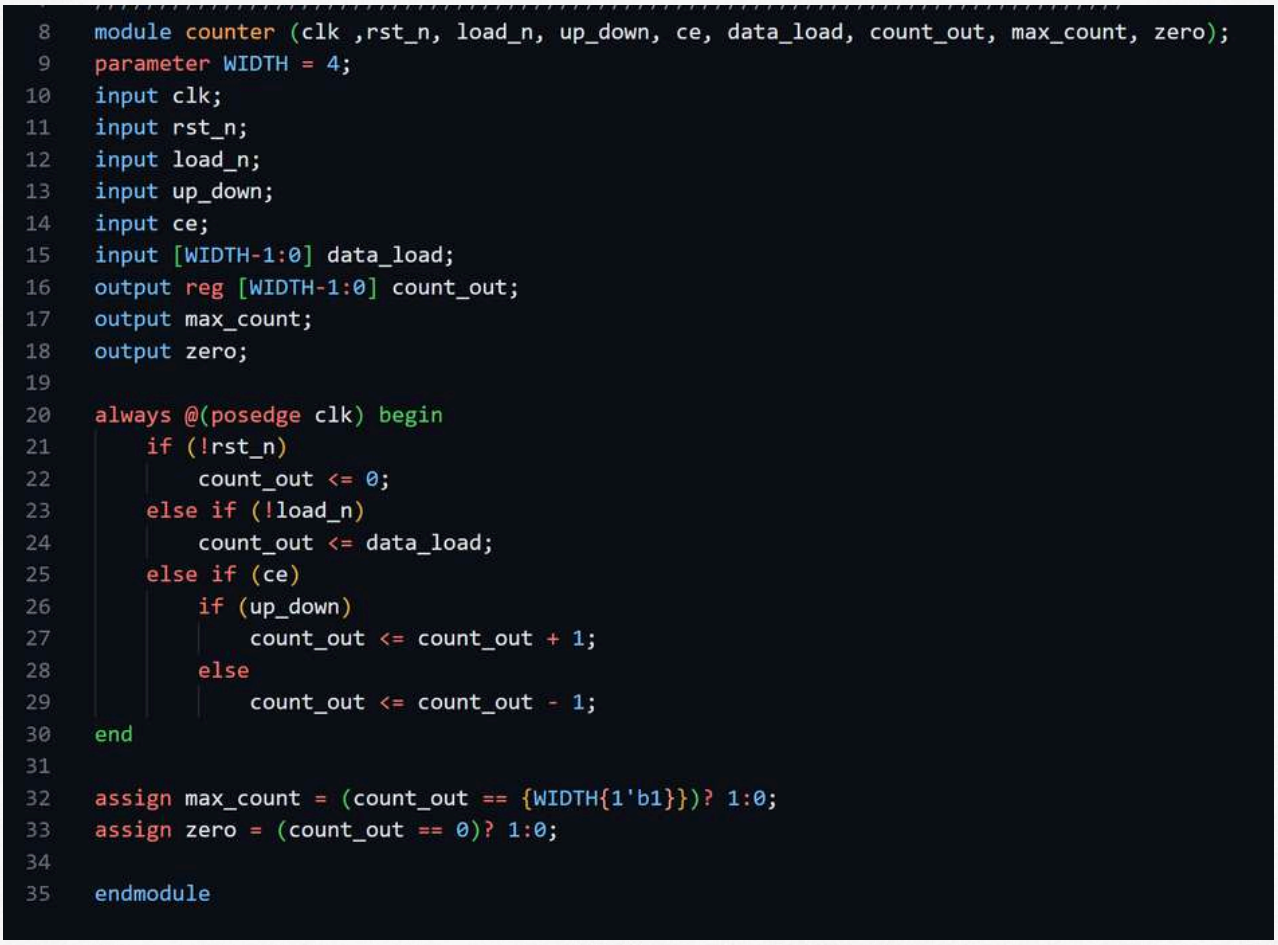
10. Functionalcoverage report :



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


---------------------------------------------------------------------------------------------------------
 TYPE /ALU_pkg/transaction/COVCODE                  100.00%        100          -    Covered
    covered/total bins:                                   7          7          -
    missing/total bins:                                   0          7          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp1                                  100.00%        100          -    Covered
        covered/total bins:                               3          3          -
        missing/total bins:                               0          3          -
        % Hit:                                      100.00%        100          -
    Coverpoint cp2                                  100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
    Cross cp3                                       100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
 Covergroup instance \/ALU_pkg::transaction::COVCODE
                                                    100.00%        100          -    Covered
    covered/total bins:                                   7          7          -
    missing/total bins:                                   0          7          -
    % Hit:                                          100.00%        100          -
    Coverpoint cp1                                  100.00%        100          -    Covered
        covered/total bins:                               3          3          -
        missing/total bins:                               0          3          -
        % Hit:                                      100.00%        100          -
        bin max_neg                                      11          1          -    Covered
        bin max_pos                                      10          1          -    Covered
        bin zero                                          6          1          -    Covered
        default bin misc                                  6                     -    Occurred
    Coverpoint cp2                                  100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        illegal_bin no_div                                7                     -    Occurred
        bin add_sub                                      20          1          -    Covered
        bin add_sub2                                      2          1          -    Covered
    Cross cp3                                       100.00%        100          -    Covered
        covered/total bins:                               2          2          -
        missing/total bins:                               0          2          -
        % Hit:                                      100.00%        100          -
        Auto, Default and User Defined Bins:
            bin x1                                        5          1          -    Covered
            bin x2                                        8          1          -    Covered

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

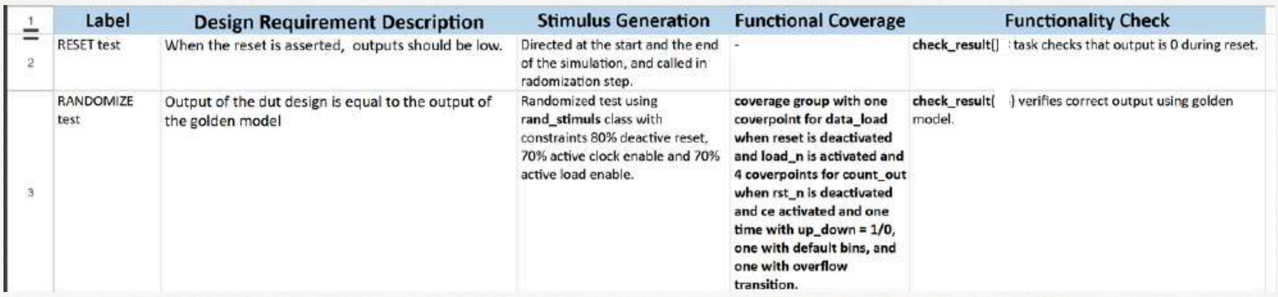
Question 2:

1. RTL design:



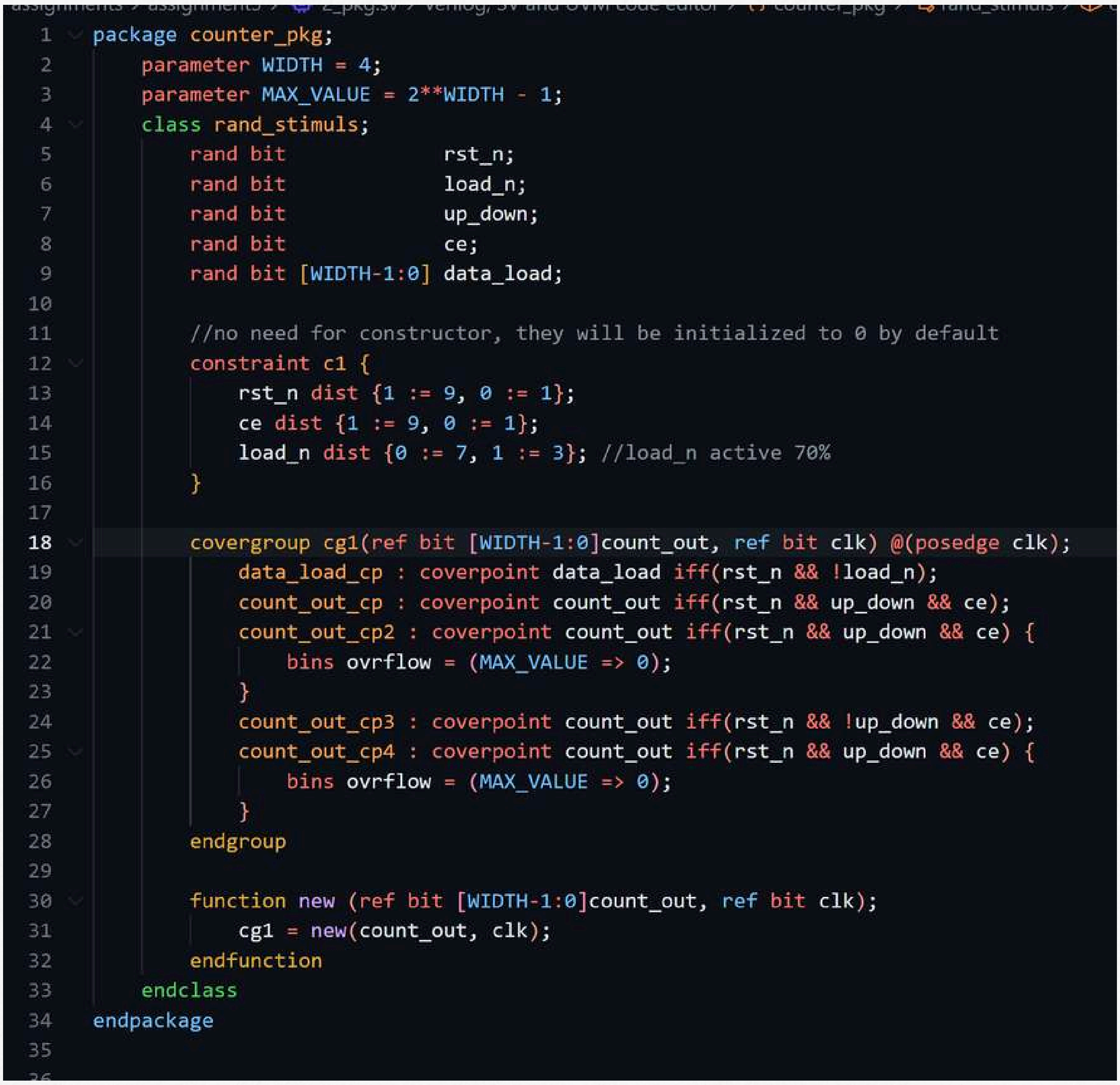
```verilog
module counter (clk ,rst_n, load_n, up_down, ce, data_load, count_out, max_count, zero);
parameter WIDTH = 4;
input clk;
input rst_n;
input load_n;
input up_down;
input ce;
input [WIDTH-1:0] data_load;
output reg [WIDTH-1:0] count_out;
output max_count;
output zero;

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

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

endmodule
```

2. Verification plan:



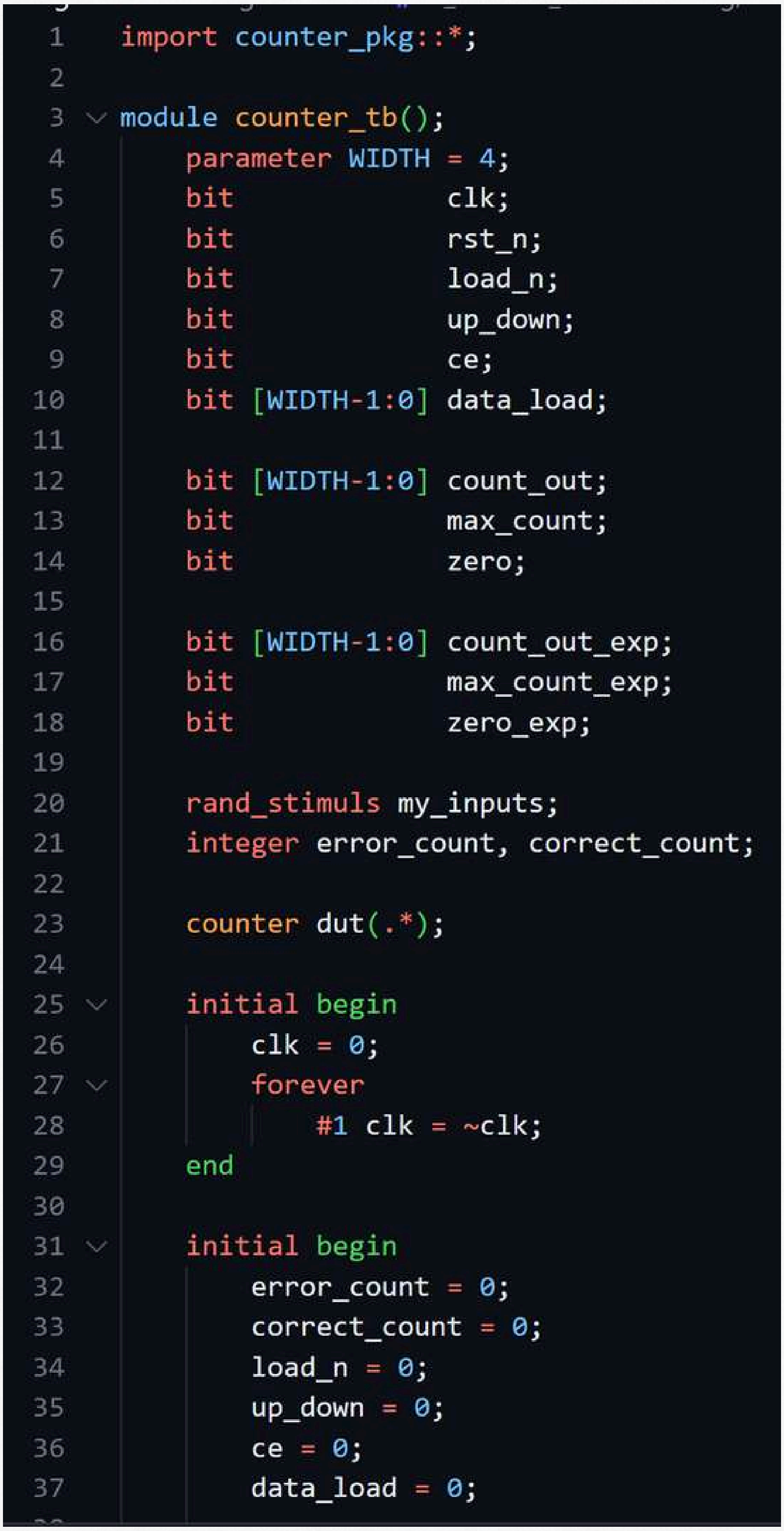
Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET test	When the reset is asserted, outputs should be low.	Directed at the start and the end of the simulation, and called in radomization step.	-	**check_result()** task checks that output is 0 during reset.
RANDOMIZE test	Output of the dut design is equal to the output of the golden model	Randomized test using **rand_stimuls** class with constraints 80% deactive reset, 70% active clock enable and 70% active load enable.	**coverage group with one coverpoint for data_load when reset is deactivated and load_n is activated and 4 coverpoints for count_out when rst_n is deactivated and ce activated and one time with up_down = 1/0, one with default bins, and one with overflow transition.**	**check_result( )** verifies correct output using golden model.

3. packege code:



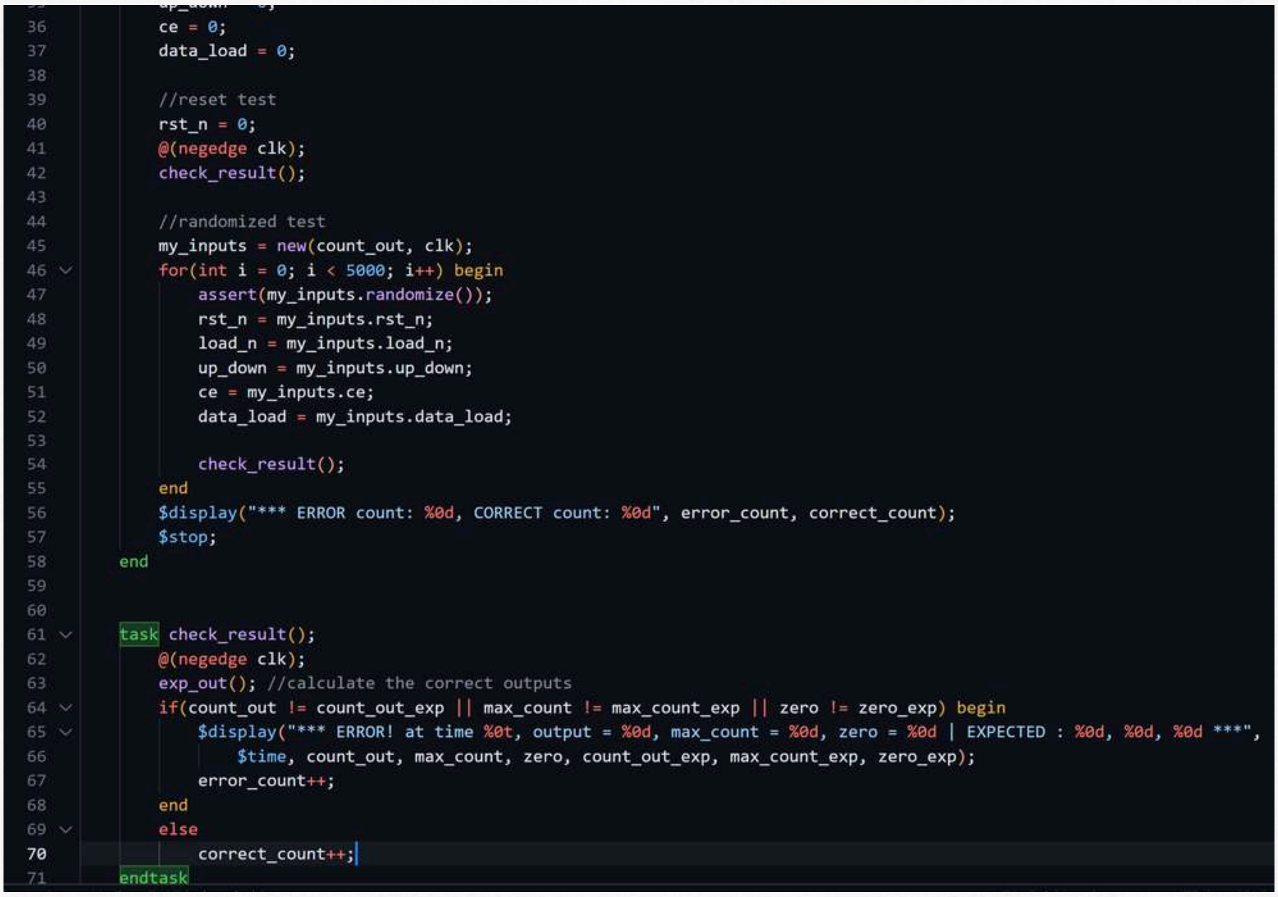
```systemverilog
package counter_pkg;
    parameter WIDTH = 4;
    parameter MAX_VALUE = 2**WIDTH - 1;
    class rand_stimuls;
        rand bit             rst_n;
        rand bit             load_n;
        rand bit             up_down;
        rand bit             ce;
        rand bit [WIDTH-1:0] data_load;

        //no need for constructor, they will be initialized to 0 by default
        constraint c1 {
            rst_n dist {1 := 9, 0 := 1};
            ce dist {1 := 9, 0 := 1};
            load_n dist {0 := 7, 1 := 3}; //load_n active 70%
        }

        covergroup cg1(ref bit [WIDTH-1:0]count_out, ref bit clk) @(posedge clk);
            data_load_cp : coverpoint data_load iff(rst_n && !load_n);
            count_out_cp : coverpoint count_out iff(rst_n && up_down && ce);
            count_out_cp2 : coverpoint count_out iff(rst_n && up_down && ce) {
                bins ovrflow = (MAX_VALUE => 0);
            }
            count_out_cp3 : coverpoint count_out iff(rst_n && !up_down && ce);
            count_out_cp4 : coverpoint count_out iff(rst_n && up_down && ce) {
                bins ovrflow = (MAX_VALUE => 0);
            }
        endgroup

        function new (ref bit [WIDTH-1:0]count_out, ref bit clk);
            cg1 = new(count_out, clk);
        endfunction
    endclass
endpackage
```

4. Testbench code:



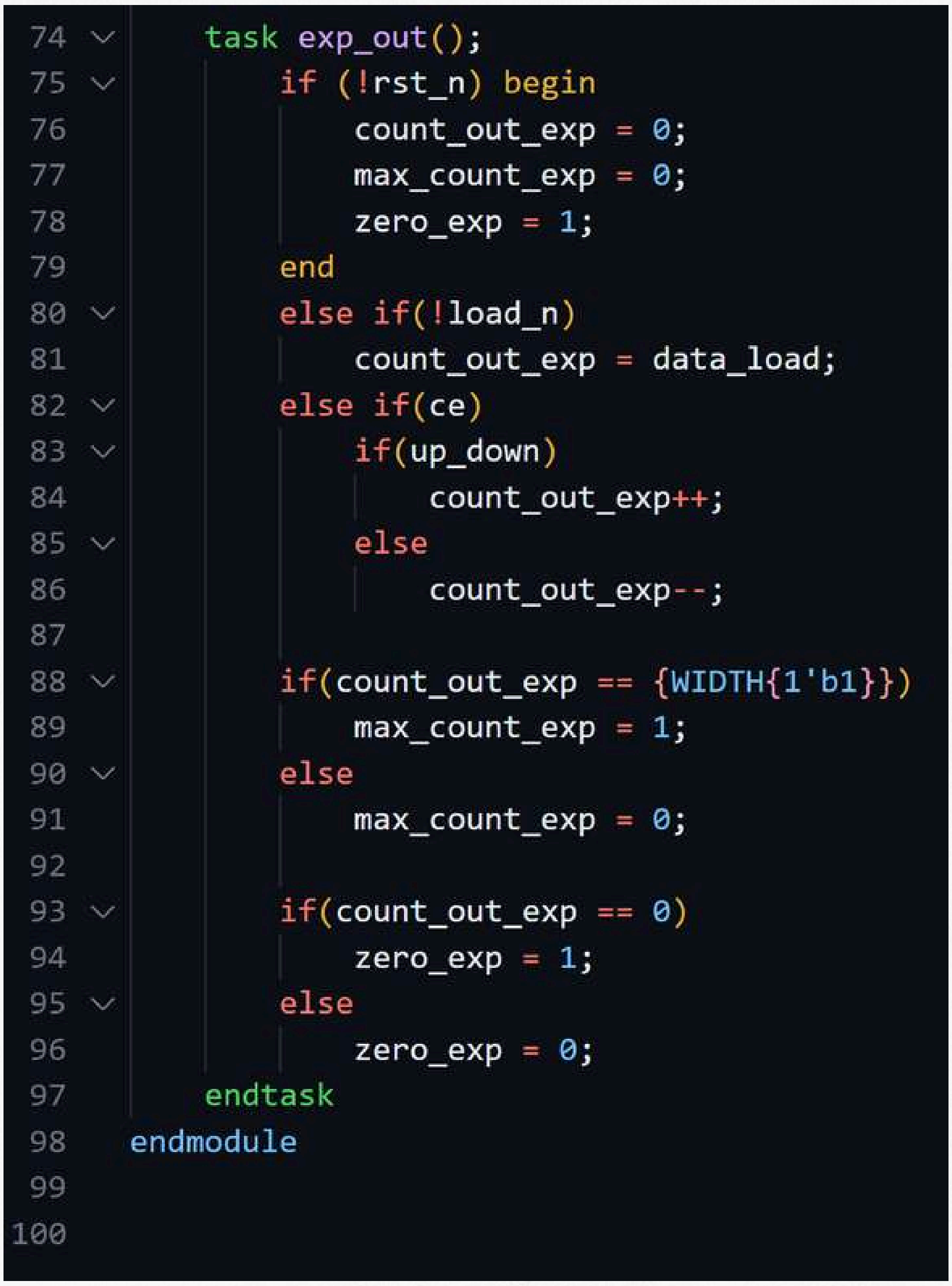
```
import counter_pkg::*;

module counter_tb();
    parameter WIDTH = 4;
    bit             clk;
    bit             rst_n;
    bit             load_n;
    bit             up_down;
    bit             ce;
    bit [WIDTH-1:0] data_load;

    bit [WIDTH-1:0] count_out;
    bit             max_count;
    bit             zero;

    bit [WIDTH-1:0] count_out_exp;
    bit             max_count_exp;
    bit             zero_exp;

    rand_stimuls my_inputs;
    integer error_count, correct_count;

    counter dut(.*);

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

    initial begin
        error_count = 0;
        correct_count = 0;
        load_n = 0;
        up_down = 0;
        ce = 0;
        data_load = 0;
```

4. Testbench code:



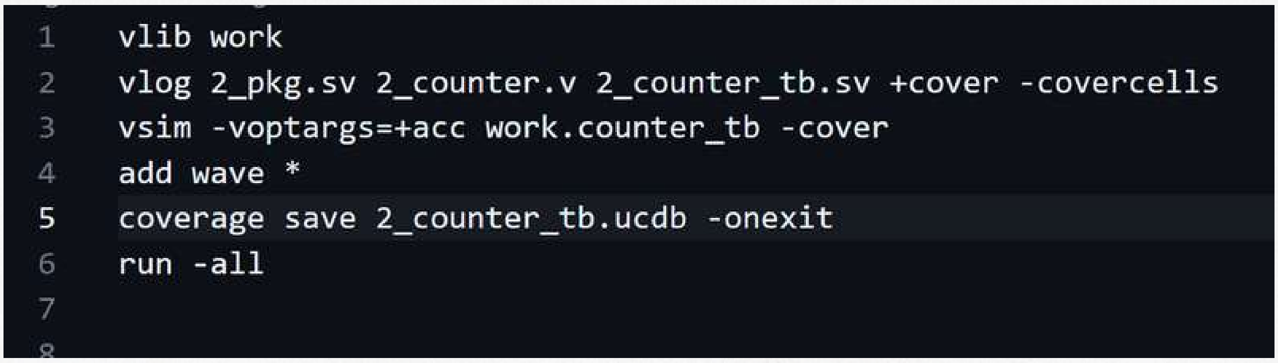
```
        ce = 0;
        data_load = 0;

        //reset test
        rst_n = 0;
        @(negedge clk);
        check_result();

        //randomized test
        my_inputs = new(count_out, clk);
        for(int i = 0; i < 5000; i++) begin
            assert(my_inputs.randomize());
            rst_n = my_inputs.rst_n;
            load_n = my_inputs.load_n;
            up_down = my_inputs.up_down;
            ce = my_inputs.ce;
            data_load = my_inputs.data_load;

            check_result();
        end
        $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
        $stop;
    end


    task check_result();
        @(negedge clk);
        exp_out(); //calculate the correct outputs
        if(count_out != count_out_exp || max_count != max_count_exp || zero != zero_exp) begin
            $display("*** ERROR! at time %0t, output = %0d, max_count = %0d, zero = %0d | EXPECTED : %0d, %0d, %0d ***",
                $time, count_out, max_count, zero, count_out_exp, max_count_exp, zero_exp);
            error_count++;
        end
        else
            correct_count++;
    endtask
```

4. Testbench code:



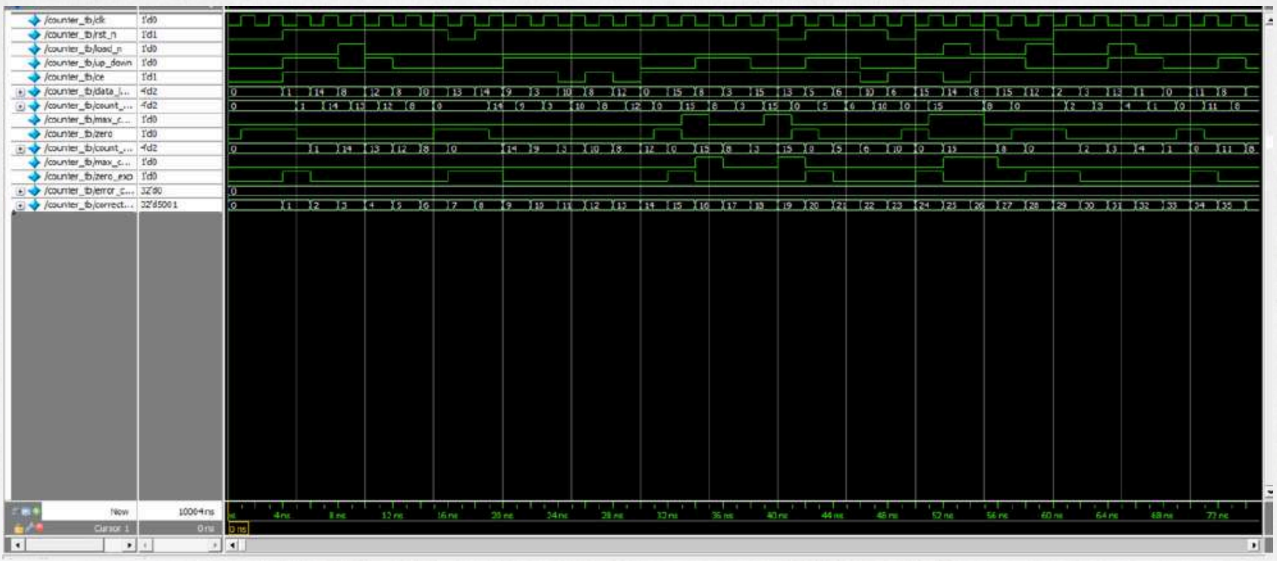
```
    task exp_out();
        if (!rst_n) begin
            count_out_exp = 0;
            max_count_exp = 0;
            zero_exp = 1;
        end
        else if(!load_n)
            count_out_exp = data_load;
        else if(ce)
            if(up_down)
                count_out_exp++;
            else
                count_out_exp--;

        if(count_out_exp == {WIDTH{1'b1}})
            max_count_exp = 1;
        else
            max_count_exp = 0;

        if(count_out_exp == 0)
            zero_exp = 1;
        else
            zero_exp = 0;
    endtask
endmodule


```

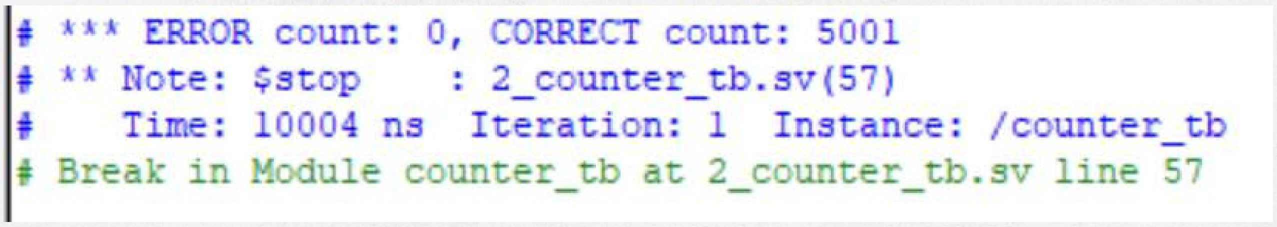
5. Do file :



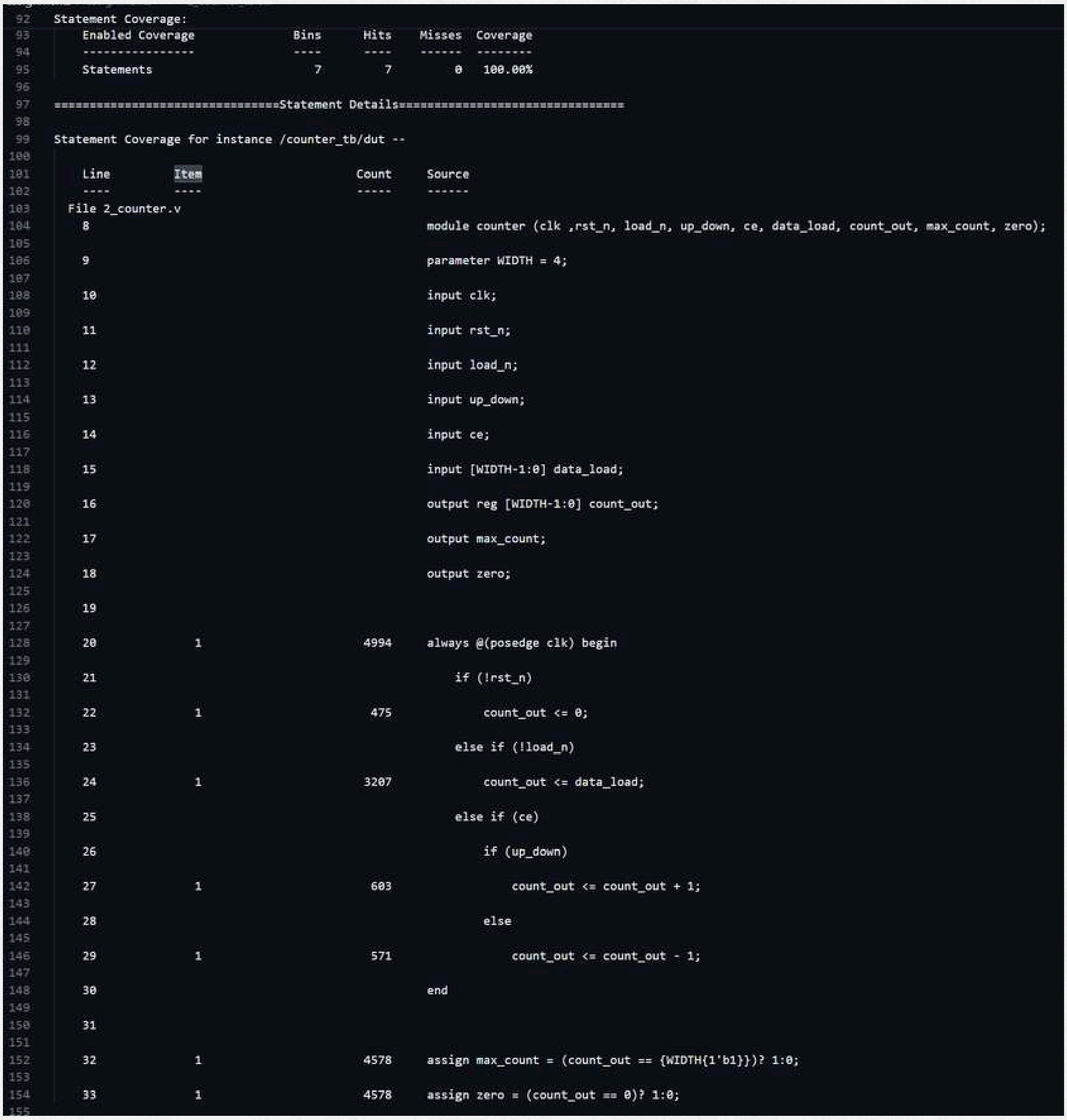
```
vlib work
vlog 2_pkg.sv 2_counter.v 2_counter_tb.sv +cover -covercells
vsim -voptargs=+acc work.counter_tb -cover
add wave *
coverage save 2_counter_tb.ucdb -onexit
run -all
```

6. Qesta sim wave snippets (Unsigned):



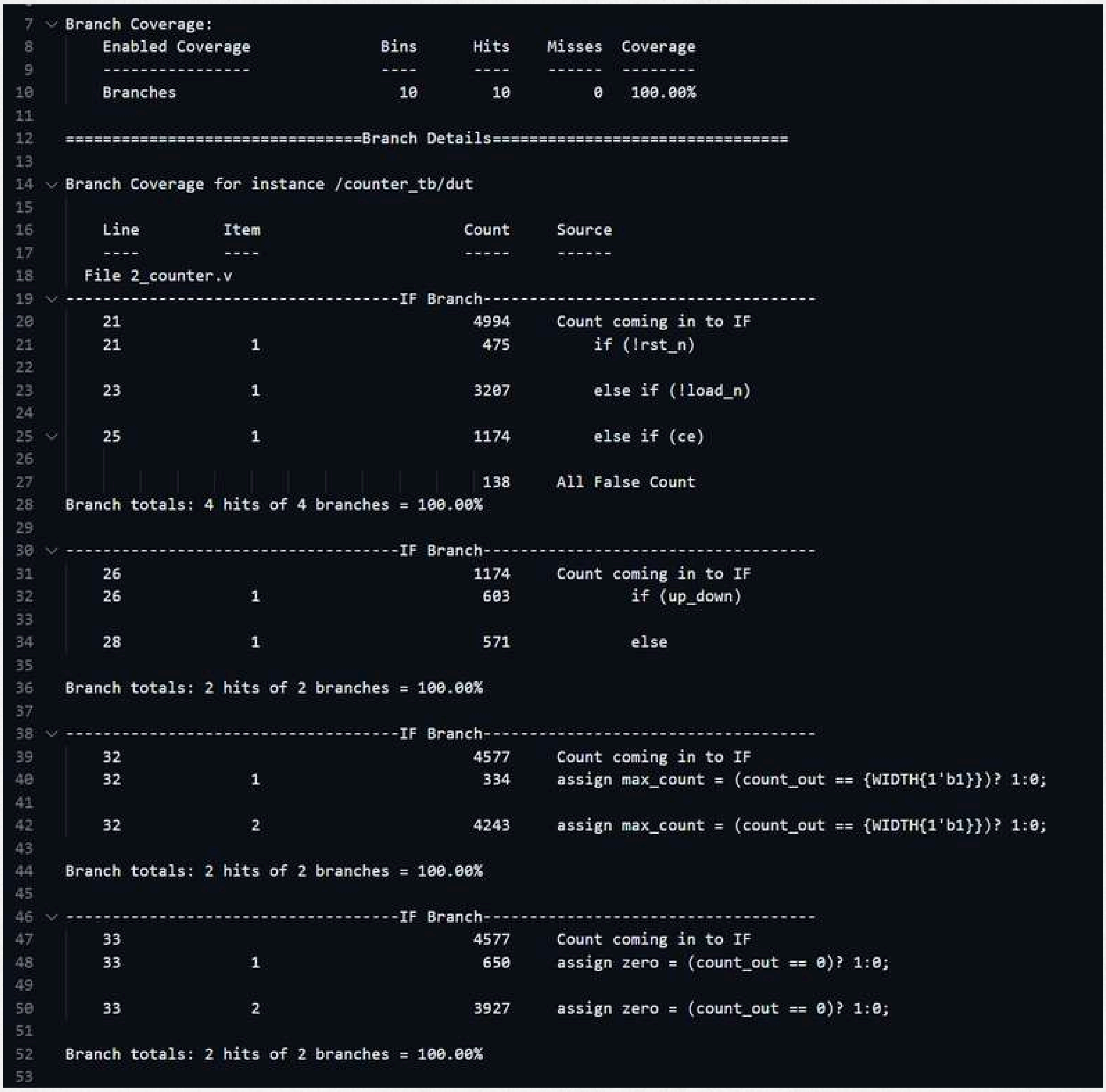

```
# *** ERROR count: 0, CORRECT count: 5001
# ** Note: $stop    : 2_counter_tb.sv(57)
#    Time: 10004 ns  Iteration: 1  Instance: /counter_tb
# Break in Module counter_tb at 2_counter_tb.sv line 57
```

7. Statement coverage report :



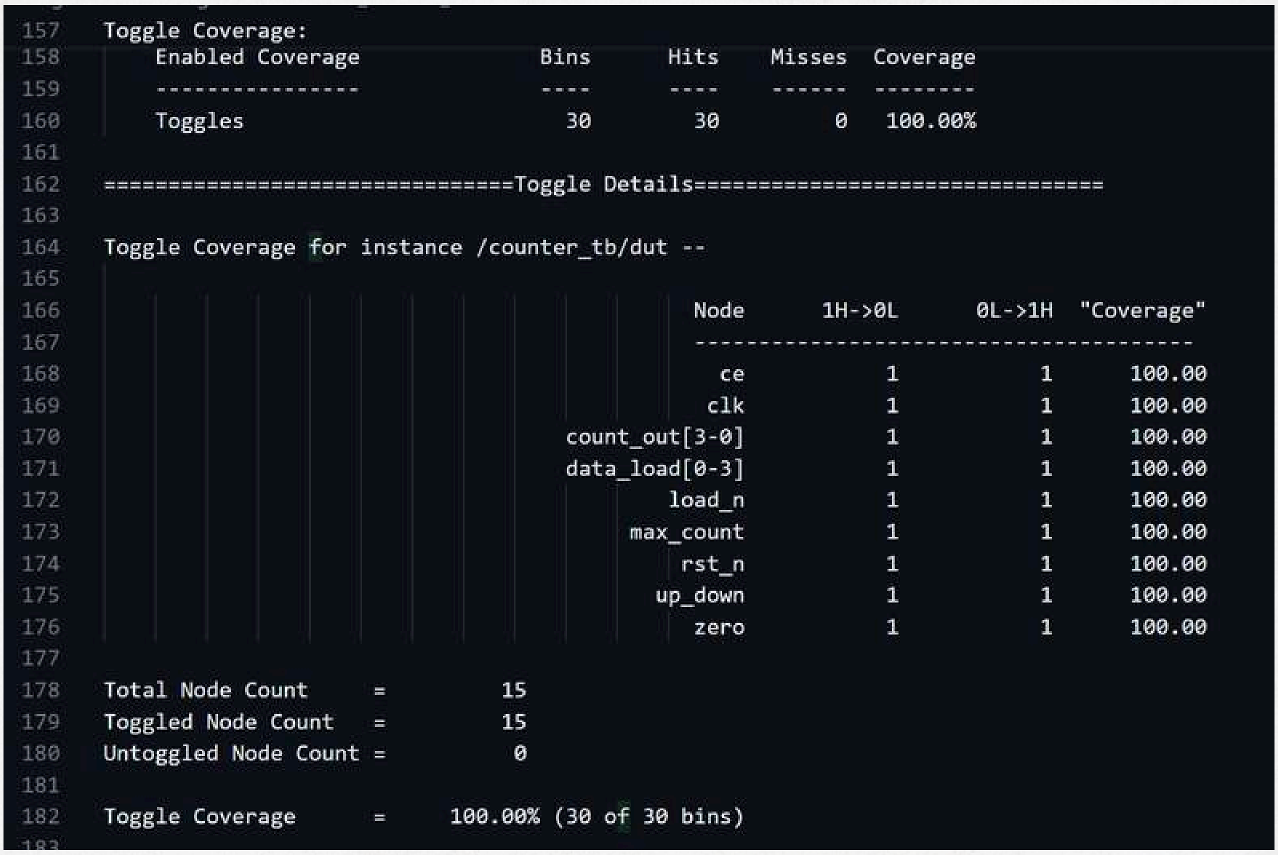
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       7         7         0   100.00%

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

Statement Coverage for instance /counter_tb/dut --

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

    9                                                parameter WIDTH = 4;

    10                                               input clk;

    11                                               input rst_n;

    12                                               input load_n;

    13                                               input up_down;

    14                                               input ce;

    15                                               input [WIDTH-1:0] data_load;

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

    17                                               output max_count;

    18                                               output zero;

    19

    20              1                       4994     always @(posedge clk) begin

    21                                                   if (!rst_n)

    22              1                        475             count_out <= 0;

    23                                                   else if (!load_n)

    24              1                       3207             count_out <= data_load;

    25                                                   else if (ce)

    26                                                       if (up_down)

    27              1                        603                 count_out <= count_out + 1;

    28                                                       else

    29              1                        571                 count_out <= count_out - 1;

    30                                               end

    31

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

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

8. Branch coverage report:



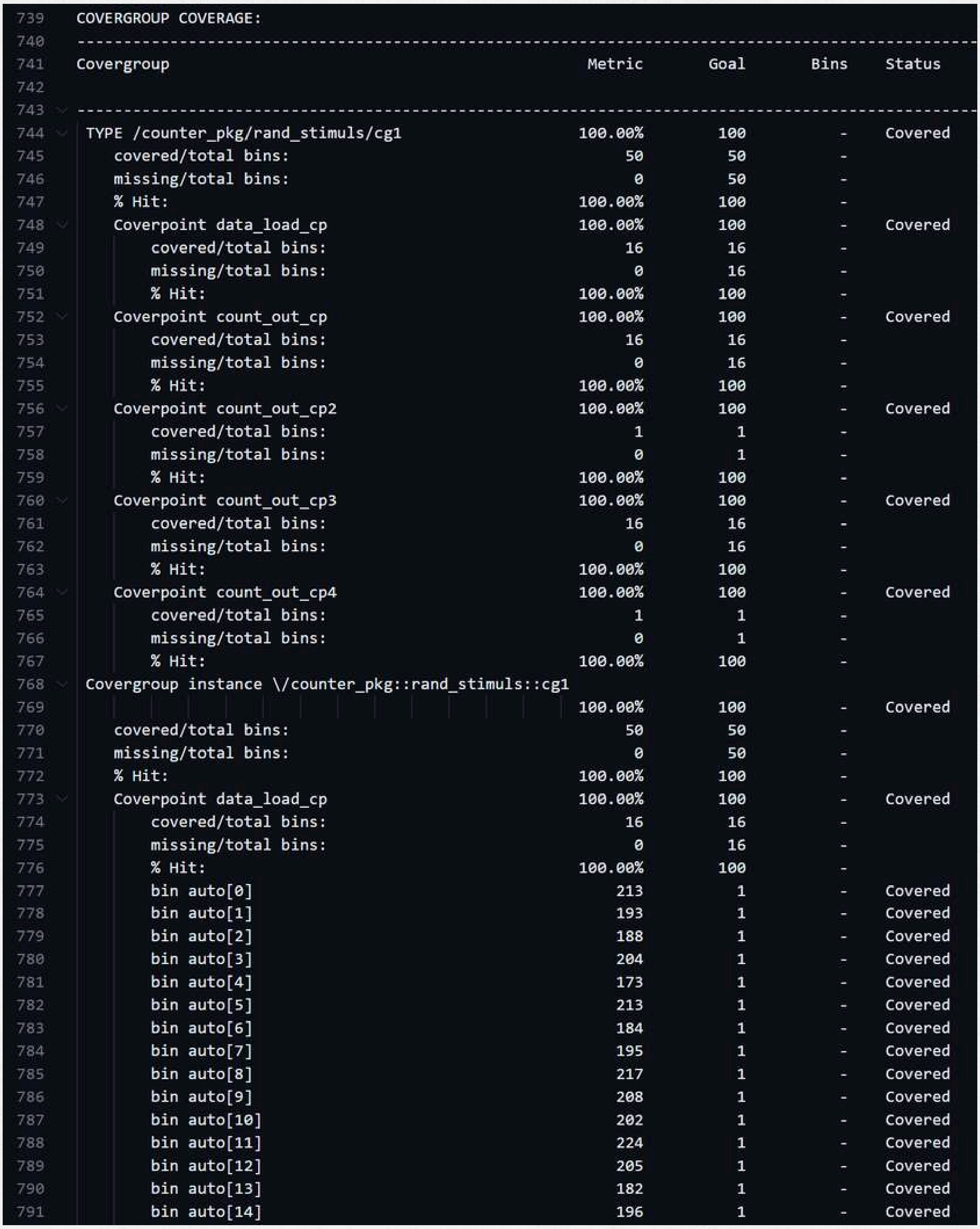
```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for instance /counter_tb/dut

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

    23              1                       3207         else if (!load_n)

    25              1                       1174         else if (ce)

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

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

    28              1                        571             else

Branch totals: 2 hits of 2 branches = 100.00%

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

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

Branch totals: 2 hits of 2 branches = 100.00%

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

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

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

10. Toggle coverage:



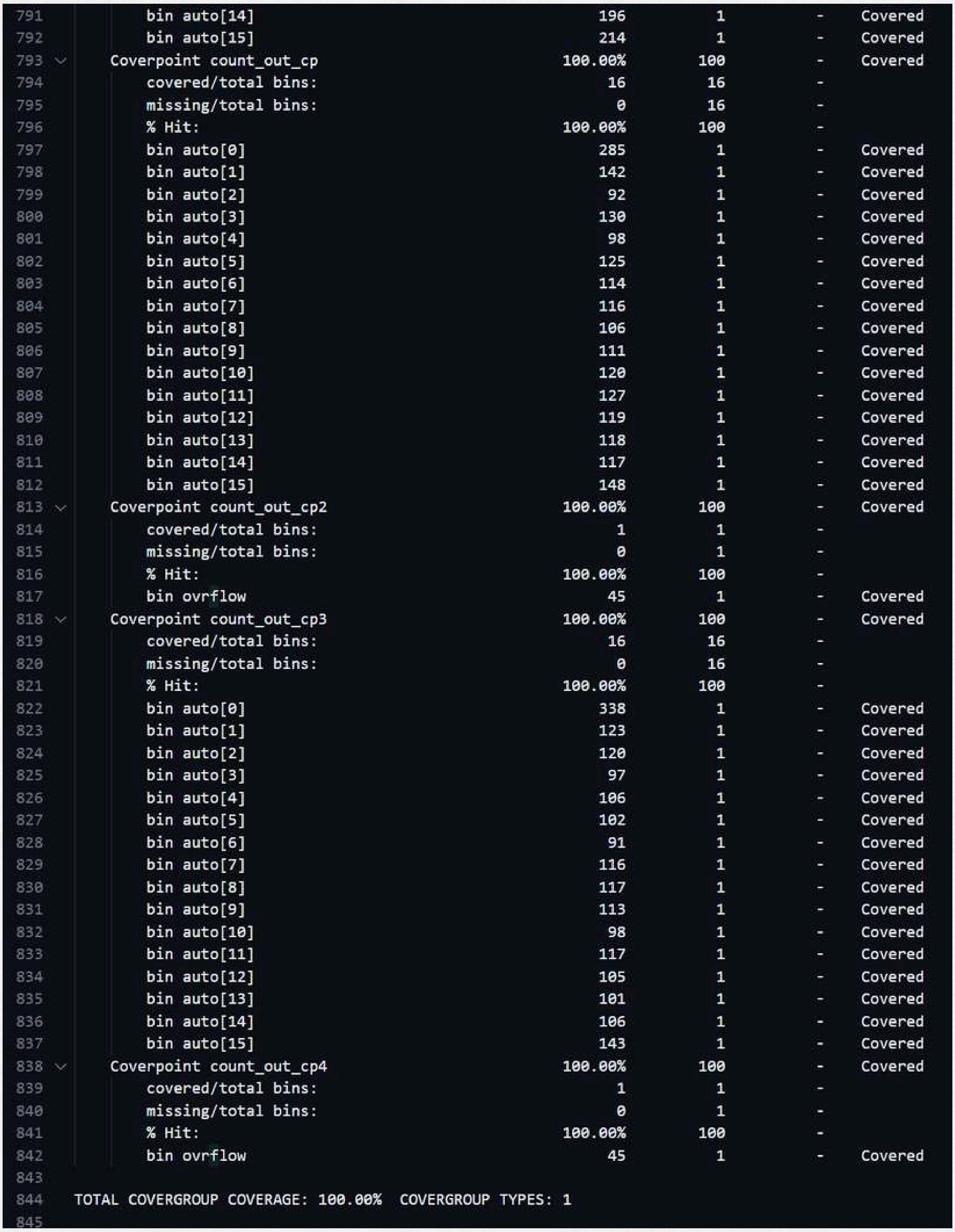
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        30         0   100.00%

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

Toggle Coverage for instance /counter_tb/dut --

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

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

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

10. Functional coverage:



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

----------------------------------------------------------------------------------------------------
 TYPE /counter_pkg/rand_stimuls/cg1                  100.00%        100          -    Covered
    covered/total bins:                                   50         50          -
    missing/total bins:                                    0         50          -
    % Hit:                                           100.00%        100          -
    Coverpoint data_load_cp                          100.00%        100          -    Covered
        covered/total bins:                               16         16          -
        missing/total bins:                                0         16          -
        % Hit:                                       100.00%        100          -
    Coverpoint count_out_cp                          100.00%        100          -    Covered
        covered/total bins:                               16         16          -
        missing/total bins:                                0         16          -
        % Hit:                                       100.00%        100          -
    Coverpoint count_out_cp2                         100.00%        100          -    Covered
        covered/total bins:                                1          1          -
        missing/total bins:                                0          1          -
        % Hit:                                       100.00%        100          -
    Coverpoint count_out_cp3                         100.00%        100          -    Covered
        covered/total bins:                               16         16          -
        missing/total bins:                                0         16          -
        % Hit:                                       100.00%        100          -
    Coverpoint count_out_cp4                         100.00%        100          -    Covered
        covered/total bins:                                1          1          -
        missing/total bins:                                0          1          -
        % Hit:                                       100.00%        100          -
 Covergroup instance \/counter_pkg::rand_stimuls::cg1
                                                     100.00%        100          -    Covered
    covered/total bins:                                   50         50          -
    missing/total bins:                                    0         50          -
    % Hit:                                           100.00%        100          -
    Coverpoint data_load_cp                          100.00%        100          -    Covered
        covered/total bins:                               16         16          -
        missing/total bins:                                0         16          -
        % Hit:                                       100.00%        100          -
        bin auto[0]                                      213          1          -    Covered
        bin auto[1]                                      193          1          -    Covered
        bin auto[2]                                      188          1          -    Covered
        bin auto[3]                                      204          1          -    Covered
        bin auto[4]                                      173          1          -    Covered
        bin auto[5]                                      213          1          -    Covered
        bin auto[6]                                      184          1          -    Covered
        bin auto[7]                                      195          1          -    Covered
        bin auto[8]                                      217          1          -    Covered
        bin auto[9]                                      208          1          -    Covered
        bin auto[10]                                     202          1          -    Covered
        bin auto[11]                                     224          1          -    Covered
        bin auto[12]                                     205          1          -    Covered
        bin auto[13]                                     182          1          -    Covered
        bin auto[14]                                     196          1          -    Covered
```

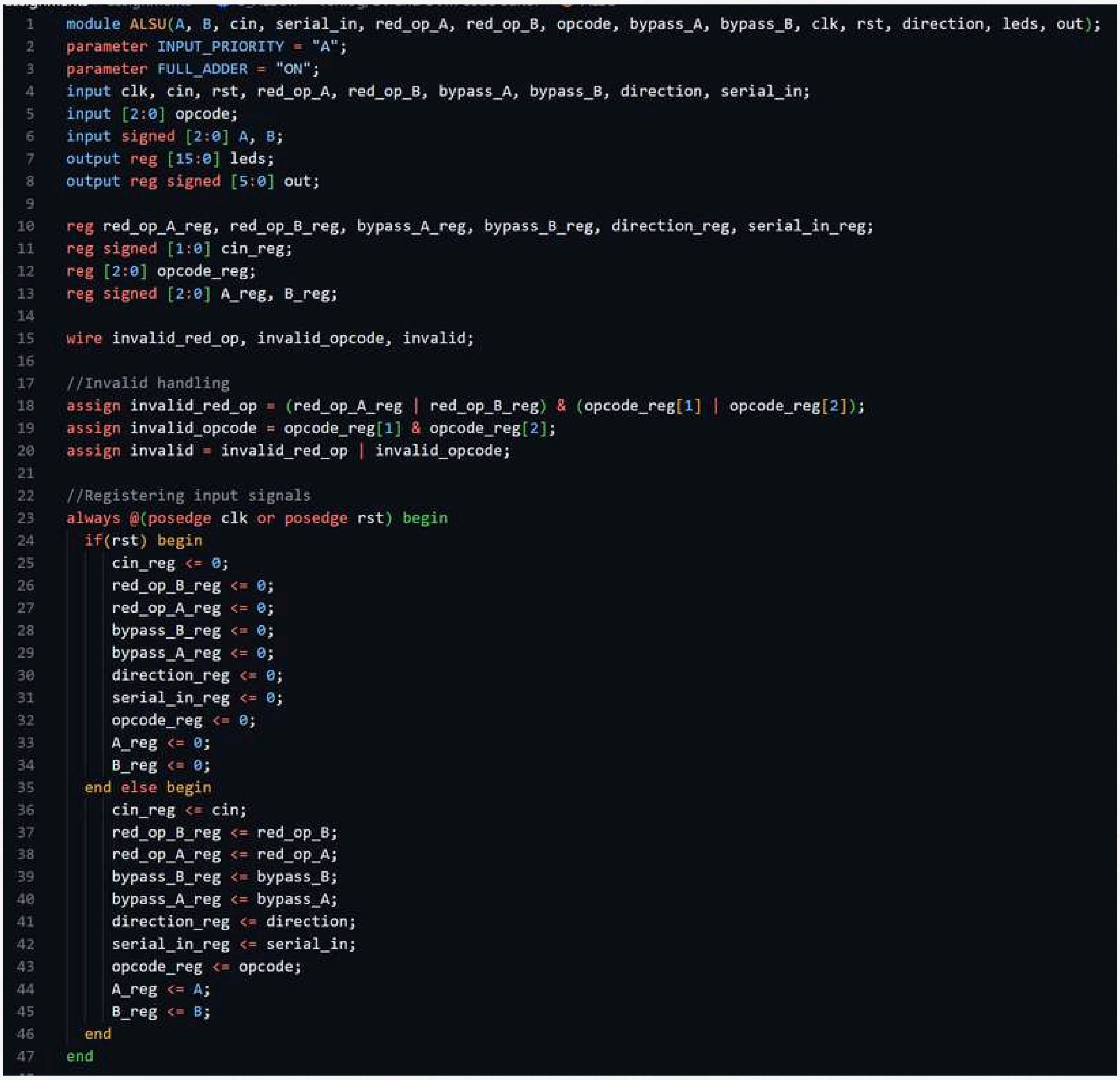
10. Functional coverage:



```
        bin auto[14]                                   196          1           -      Covered
        bin auto[15]                                   214          1           -      Covered
    Coverpoint count_out_cp                        100.00%        100           -      Covered
        covered/total bins:                             16         16           -
        missing/total bins:                              0         16           -
        % Hit:                                     100.00%        100           -
        bin auto[0]                                    285          1           -      Covered
        bin auto[1]                                    142          1           -      Covered
        bin auto[2]                                     92          1           -      Covered
        bin auto[3]                                    130          1           -      Covered
        bin auto[4]                                     98          1           -      Covered
        bin auto[5]                                    125          1           -      Covered
        bin auto[6]                                    114          1           -      Covered
        bin auto[7]                                    116          1           -      Covered
        bin auto[8]                                    106          1           -      Covered
        bin auto[9]                                    111          1           -      Covered
        bin auto[10]                                   120          1           -      Covered
        bin auto[11]                                   127          1           -      Covered
        bin auto[12]                                   119          1           -      Covered
        bin auto[13]                                   118          1           -      Covered
        bin auto[14]                                   117          1           -      Covered
        bin auto[15]                                   148          1           -      Covered
    Coverpoint count_out_cp2                       100.00%        100           -      Covered
        covered/total bins:                              1          1           -
        missing/total bins:                              0          1           -
        % Hit:                                     100.00%        100           -
        bin ovrflow                                     45          1           -      Covered
    Coverpoint count_out_cp3                       100.00%        100           -      Covered
        covered/total bins:                             16         16           -
        missing/total bins:                              0         16           -
        % Hit:                                     100.00%        100           -
        bin auto[0]                                    338          1           -      Covered
        bin auto[1]                                    123          1           -      Covered
        bin auto[2]                                    120          1           -      Covered
        bin auto[3]                                     97          1           -      Covered
        bin auto[4]                                    106          1           -      Covered
        bin auto[5]                                    102          1           -      Covered
        bin auto[6]                                     91          1           -      Covered
        bin auto[7]                                    116          1           -      Covered
        bin auto[8]                                    117          1           -      Covered
        bin auto[9]                                    113          1           -      Covered
        bin auto[10]                                    98          1           -      Covered
        bin auto[11]                                   117          1           -      Covered
        bin auto[12]                                   105          1           -      Covered
        bin auto[13]                                   101          1           -      Covered
        bin auto[14]                                   106          1           -      Covered
        bin auto[15]                                   143          1           -      Covered
    Coverpoint count_out_cp4                       100.00%        100           -      Covered
        covered/total bins:                              1          1           -
        missing/total bins:                              0          1           -
        % Hit:                                     100.00%        100           -
        bin ovrflow                                     45          1           -      Covered

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

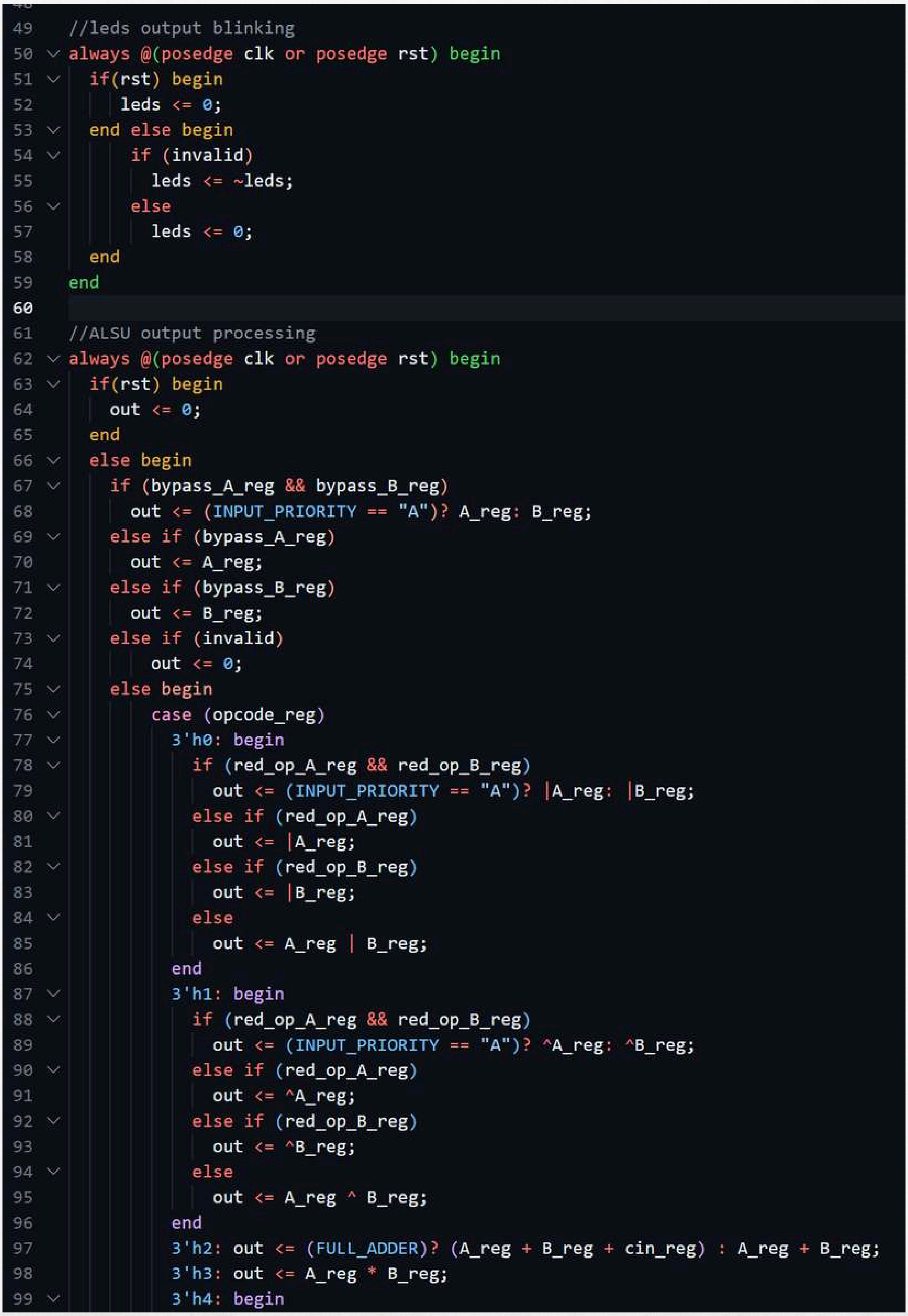
Question 3:

1. RTL design:



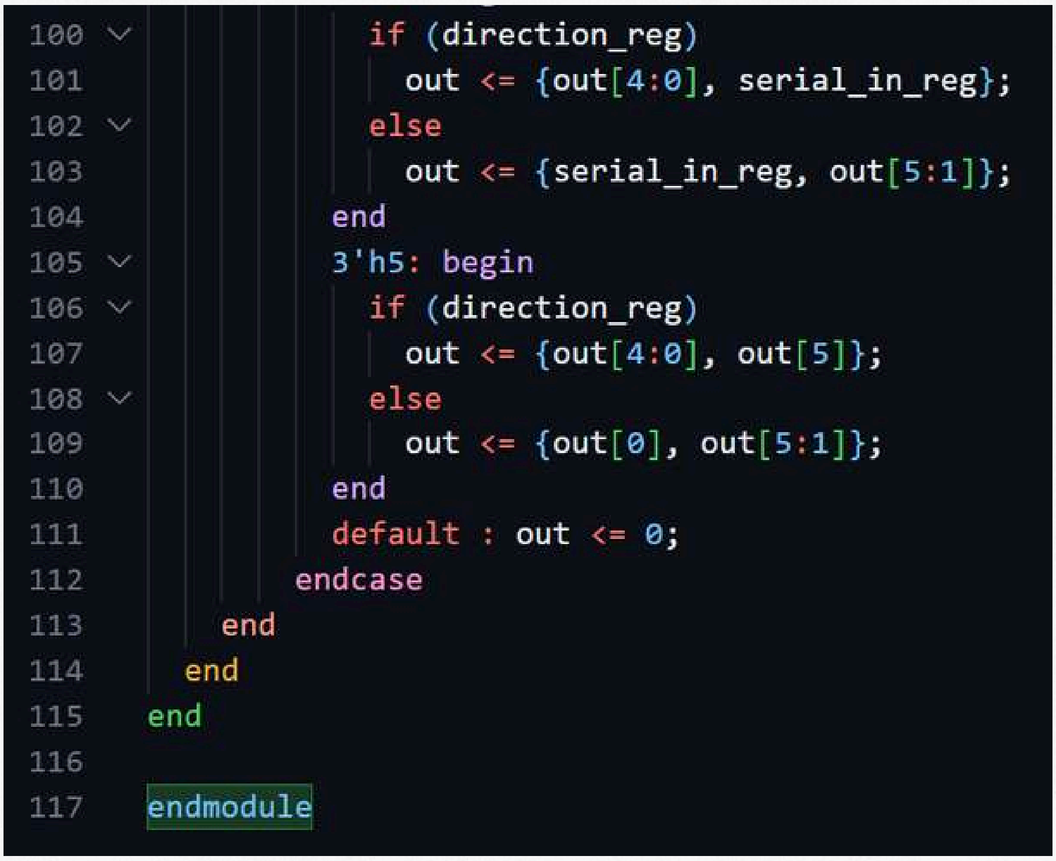
```verilog
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input [2:0] opcode;
input signed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

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

wire invalid_red_op, invalid_opcode, invalid;

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

//Registering input signals
always @(posedge clk or posedge rst) begin
  if(rst) begin
      cin_reg <= 0;
      red_op_B_reg <= 0;
      red_op_A_reg <= 0;
      bypass_B_reg <= 0;
      bypass_A_reg <= 0;
      direction_reg <= 0;
      serial_in_reg <= 0;
      opcode_reg <= 0;
      A_reg <= 0;
      B_reg <= 0;
  end else begin
      cin_reg <= cin;
      red_op_B_reg <= red_op_B;
      red_op_A_reg <= red_op_A;
      bypass_B_reg <= bypass_B;
      bypass_A_reg <= bypass_A;
      direction_reg <= direction;
      serial_in_reg <= serial_in;
      opcode_reg <= opcode;
      A_reg <= A;
      B_reg <= B;
  end
end
```

1. RTL design:



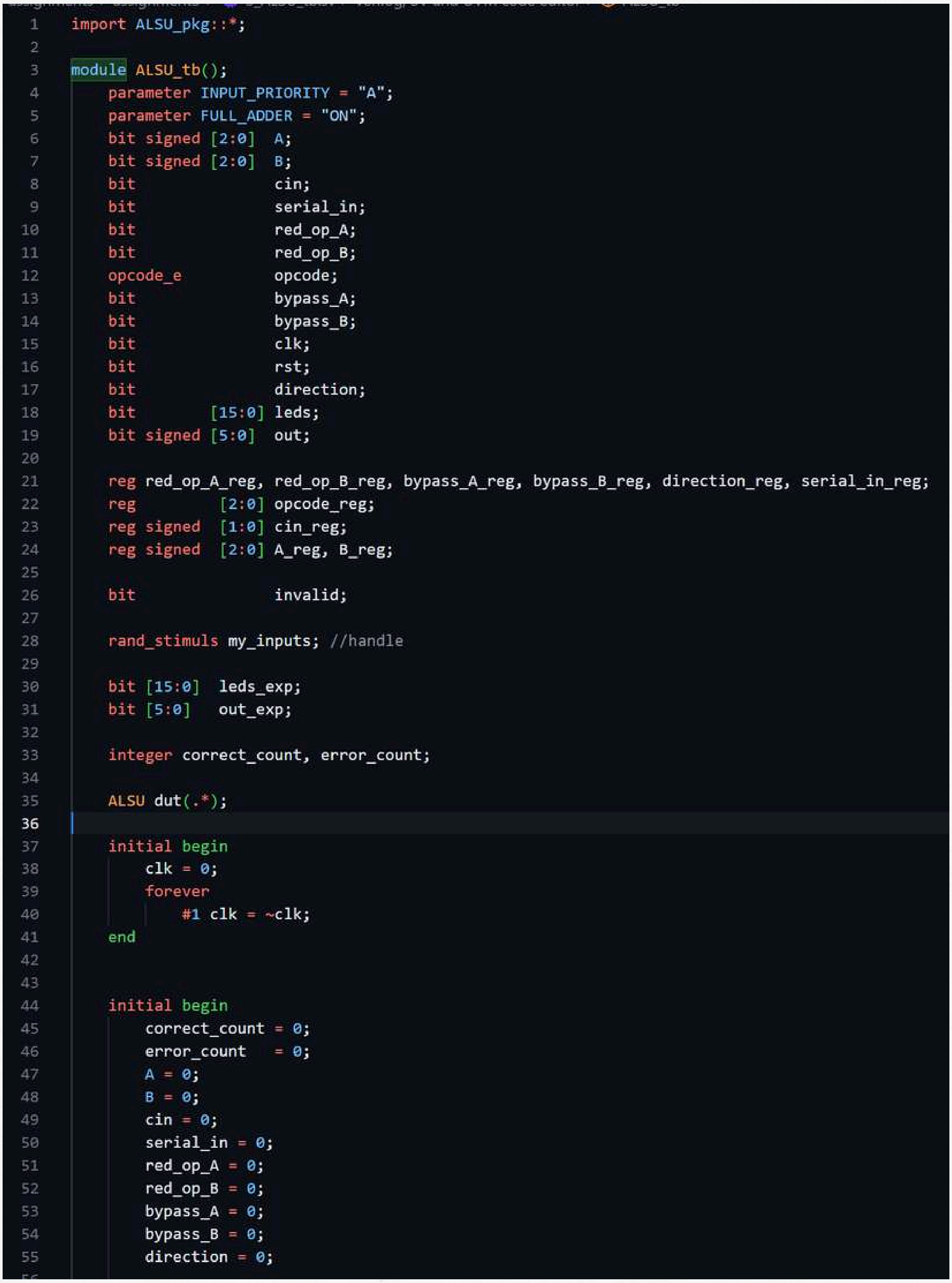
```verilog
//leds output blinking
always @(posedge clk or posedge rst) begin
  if(rst) begin
     leds <= 0;
  end else begin
      if (invalid)
        leds <= ~leds;
      else
        leds <= 0;
  end
end

//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)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;
          3'h3: out <= A_reg * B_reg;
          3'h4: begin
```

1. RTL design:



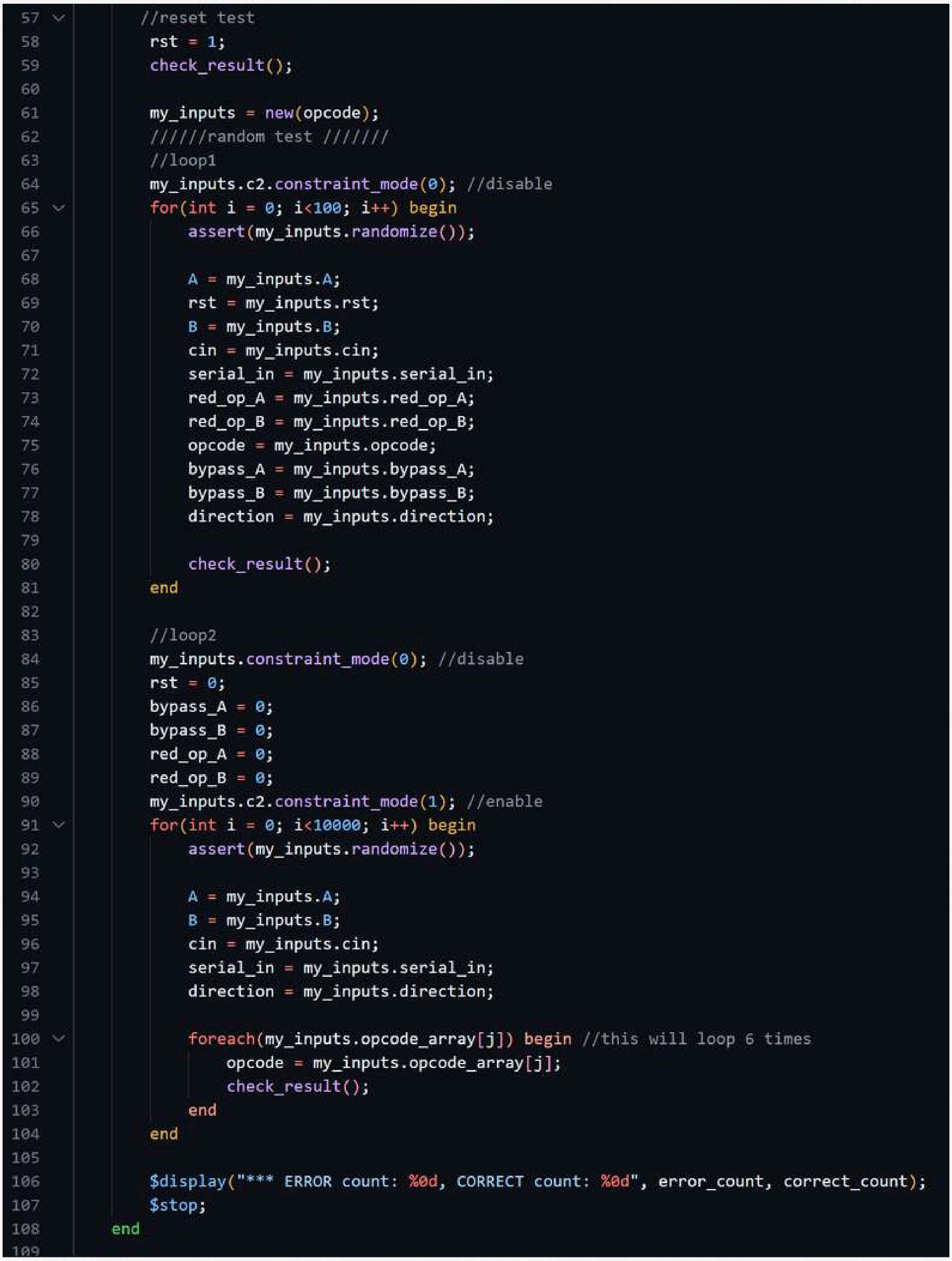
```
                if (direction_reg)
                  out <= {out[4:0], serial_in_reg};
                else
                  out <= {serial_in_reg, out[5:1]};
              end
              3'h5: begin
                if (direction_reg)
                  out <= {out[4:0], out[5]};
                else
                  out <= {out[0], out[5:1]};
              end
              default : out <= 0;
            endcase
      end
    end
end

endmodule
```

2. Testbench:



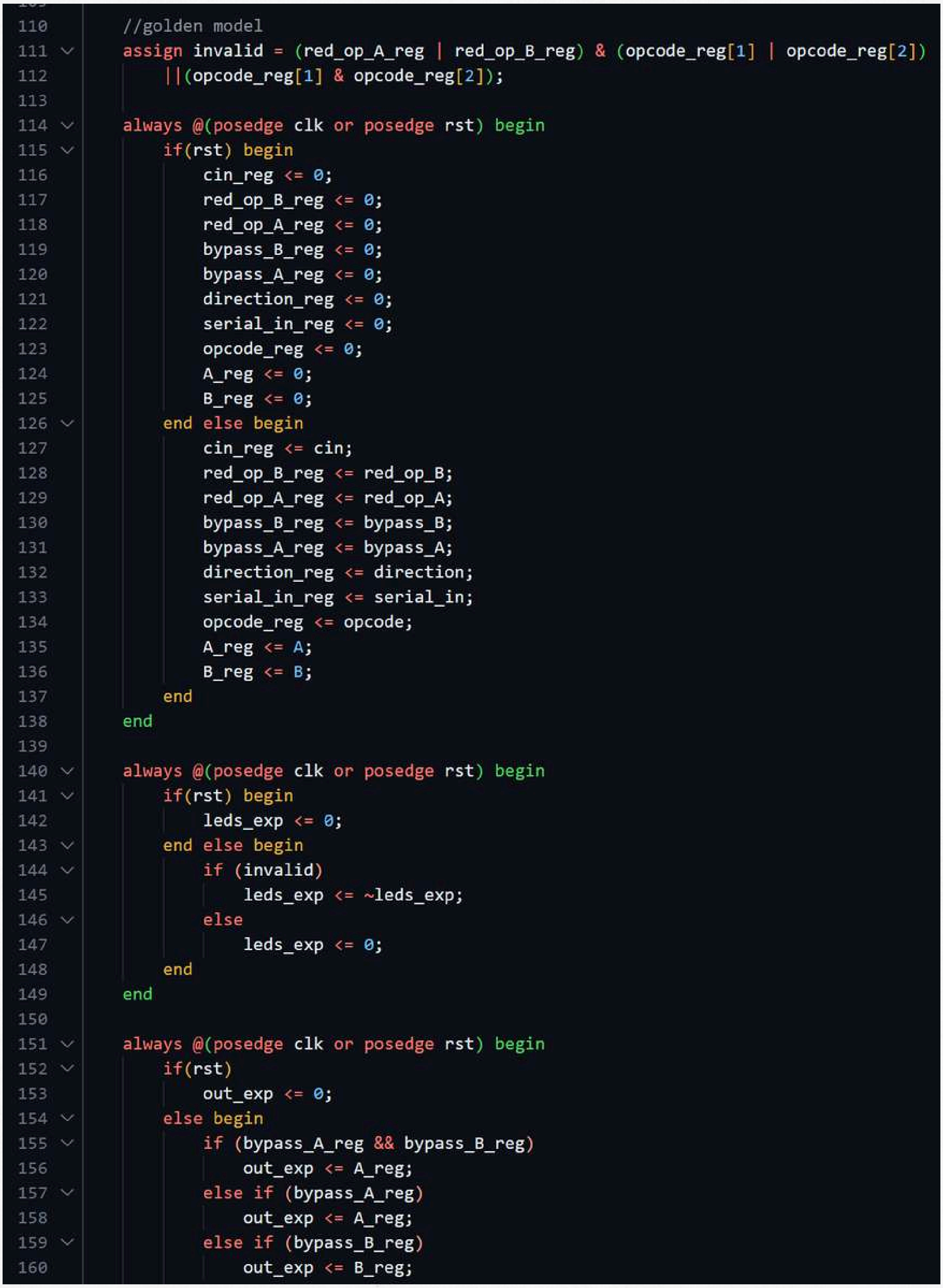
```
import ALSU_pkg::*;

module ALSU_tb();
    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    bit signed [2:0]  A;
    bit signed [2:0]  B;
    bit               cin;
    bit               serial_in;
    bit               red_op_A;
    bit               red_op_B;
    opcode_e          opcode;
    bit               bypass_A;
    bit               bypass_B;
    bit               clk;
    bit               rst;
    bit               direction;
    bit        [15:0] leds;
    bit signed [5:0]  out;

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

    bit               invalid;

    rand_stimuls my_inputs; //handle

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

    integer correct_count, error_count;

    ALSU dut(.*);

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


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

2. Testbench:



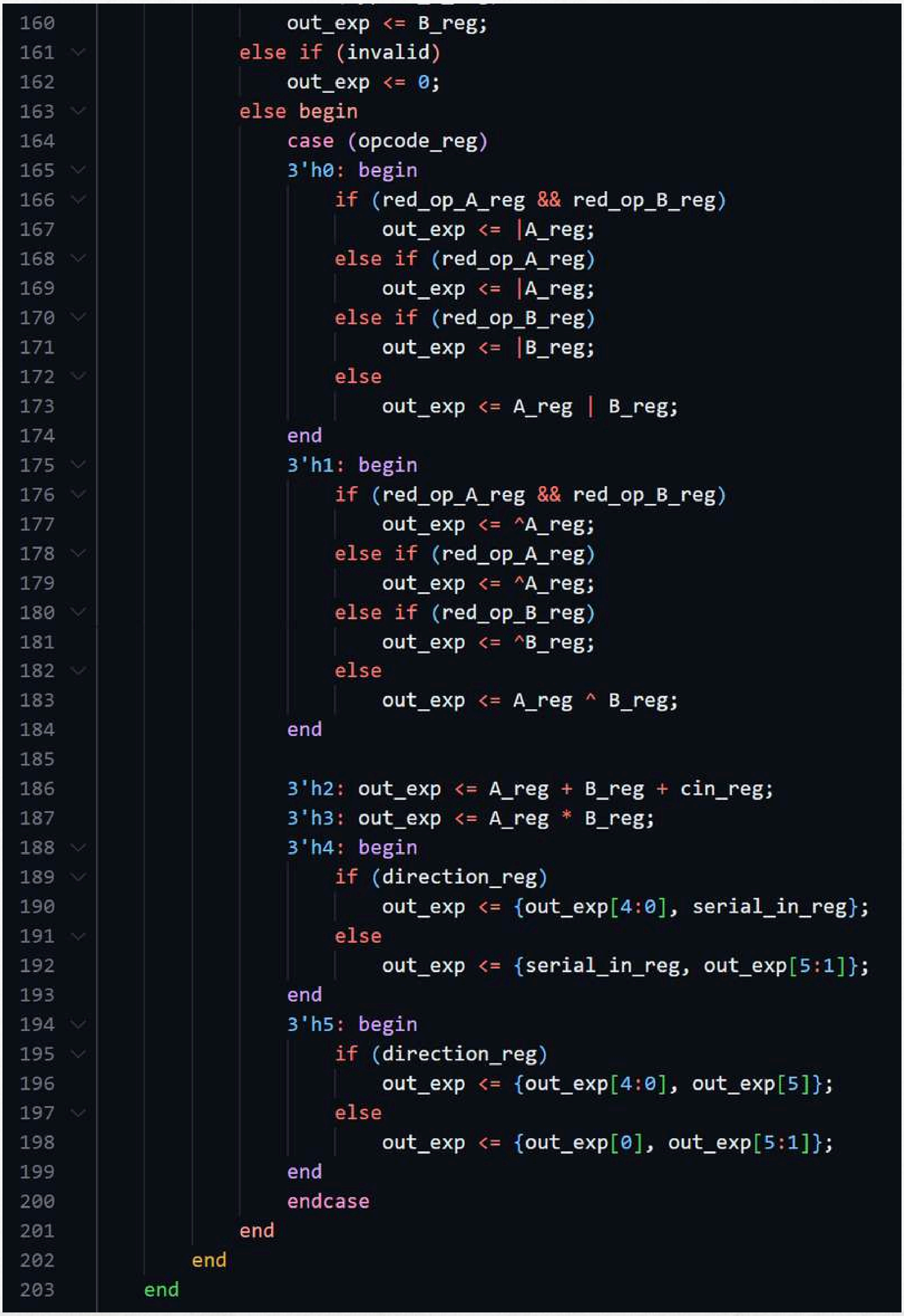
```
        //reset test
        rst = 1;
        check_result();

        my_inputs = new(opcode);
        //////random test ///////
        //loop1
        my_inputs.c2.constraint_mode(0); //disable
        for(int i = 0; i<100; i++) begin
            assert(my_inputs.randomize());

            A = my_inputs.A;
            rst = my_inputs.rst;
            B = my_inputs.B;
            cin = my_inputs.cin;
            serial_in = my_inputs.serial_in;
            red_op_A = my_inputs.red_op_A;
            red_op_B = my_inputs.red_op_B;
            opcode = my_inputs.opcode;
            bypass_A = my_inputs.bypass_A;
            bypass_B = my_inputs.bypass_B;
            direction = my_inputs.direction;

            check_result();
        end

        //loop2
        my_inputs.constraint_mode(0); //disable
        rst = 0;
        bypass_A = 0;
        bypass_B = 0;
        red_op_A = 0;
        red_op_B = 0;
        my_inputs.c2.constraint_mode(1); //enable
        for(int i = 0; i<10000; i++) begin
            assert(my_inputs.randomize());

            A = my_inputs.A;
            B = my_inputs.B;
            cin = my_inputs.cin;
            serial_in = my_inputs.serial_in;
            direction = my_inputs.direction;

            foreach(my_inputs.opcode_array[j]) begin //this will loop 6 times
                opcode = my_inputs.opcode_array[j];
                check_result();
            end
        end

        $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
        $stop;
    end

```

2. Testbench:



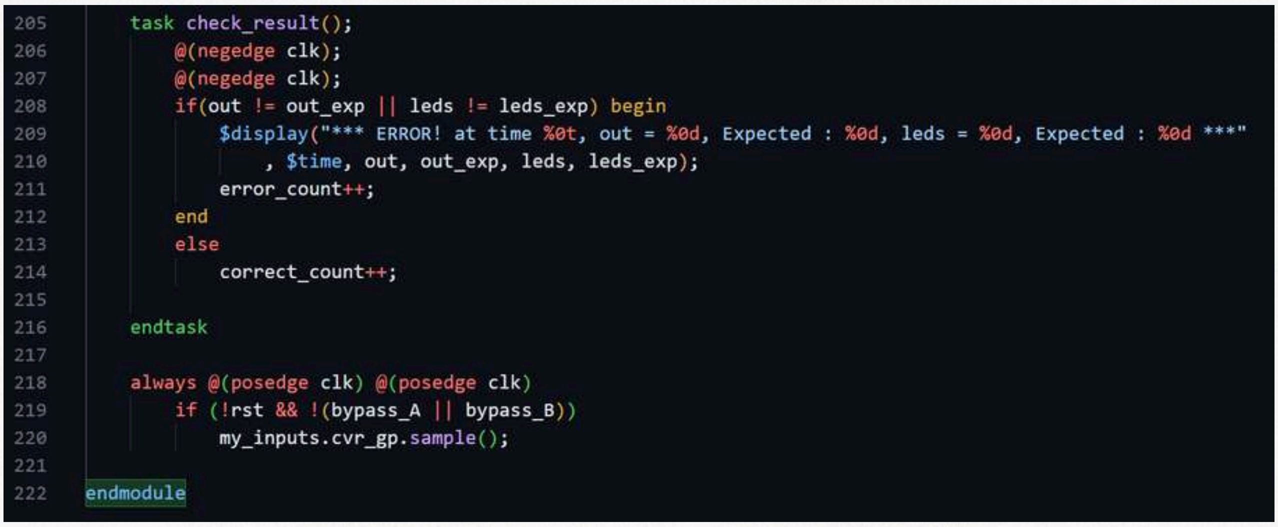
```
    //golden model
    assign invalid = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2])
        ||(opcode_reg[1] & opcode_reg[2]);

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            cin_reg <= 0;
            red_op_B_reg <= 0;
            red_op_A_reg <= 0;
            bypass_B_reg <= 0;
            bypass_A_reg <= 0;
            direction_reg <= 0;
            serial_in_reg <= 0;
            opcode_reg <= 0;
            A_reg <= 0;
            B_reg <= 0;
        end else begin
            cin_reg <= cin;
            red_op_B_reg <= red_op_B;
            red_op_A_reg <= red_op_A;
            bypass_B_reg <= bypass_B;
            bypass_A_reg <= bypass_A;
            direction_reg <= direction;
            serial_in_reg <= serial_in;
            opcode_reg <= opcode;
            A_reg <= A;
            B_reg <= B;
        end
    end

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

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

2. Testbench:



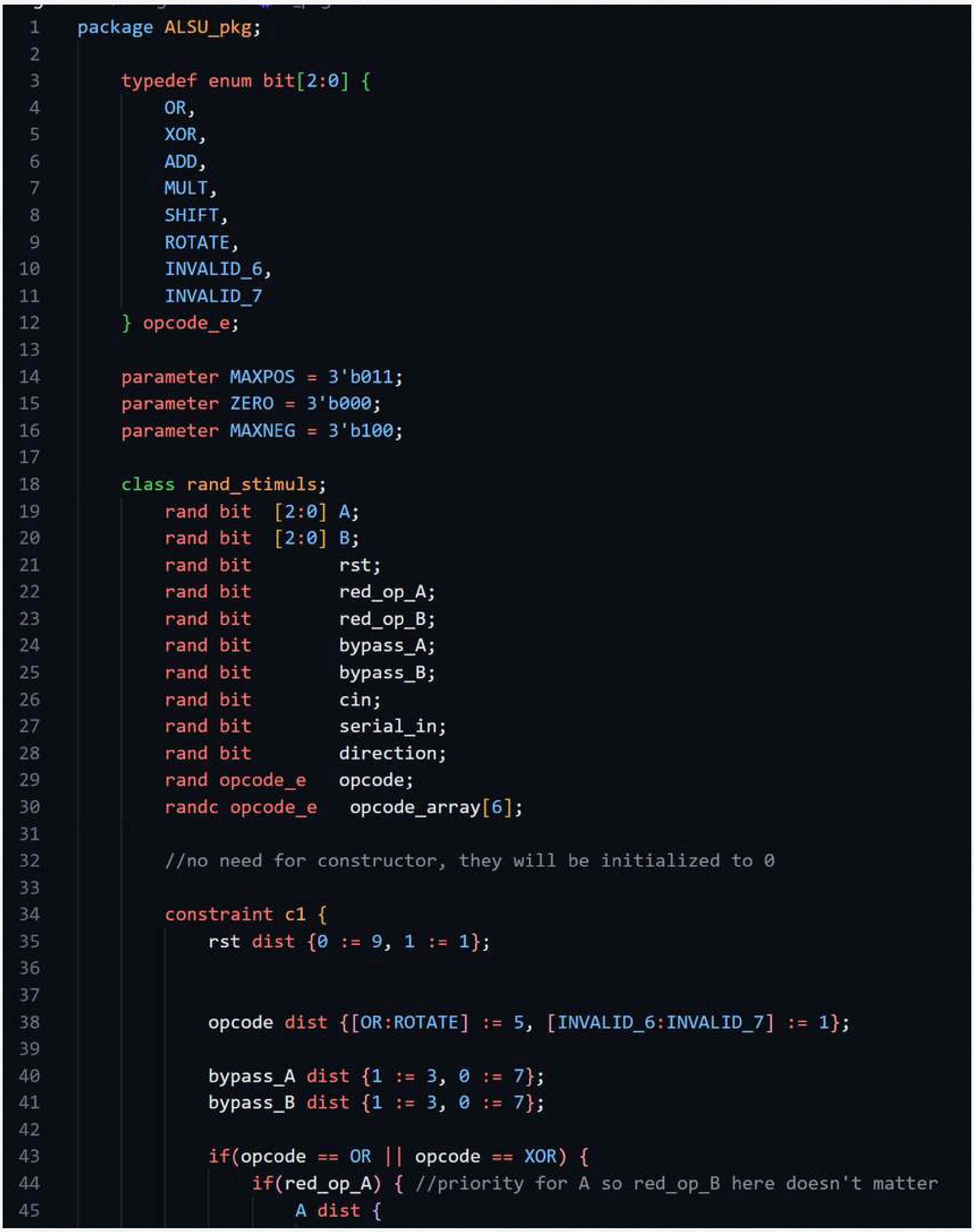
```
                out_exp <= B_reg;
            else if (invalid)
                out_exp <= 0;
            else begin
                case (opcode_reg)
                3'h0: begin
                    if (red_op_A_reg && red_op_B_reg)
                        out_exp <= |A_reg;
                    else if (red_op_A_reg)
                        out_exp <= |A_reg;
                    else if (red_op_B_reg)
                        out_exp <= |B_reg;
                    else
                        out_exp <= A_reg | B_reg;
                end
                3'h1: begin
                    if (red_op_A_reg && red_op_B_reg)
                        out_exp <= ^A_reg;
                    else if (red_op_A_reg)
                        out_exp <= ^A_reg;
                    else if (red_op_B_reg)
                        out_exp <= ^B_reg;
                    else
                        out_exp <= A_reg ^ B_reg;
                end

                3'h2: out_exp <= A_reg + B_reg + cin_reg;
                3'h3: out_exp <= A_reg * B_reg;
                3'h4: begin
                    if (direction_reg)
                        out_exp <= {out_exp[4:0], serial_in_reg};
                    else
                        out_exp <= {serial_in_reg, out_exp[5:1]};
                end
                3'h5: begin
                    if (direction_reg)
                        out_exp <= {out_exp[4:0], out_exp[5]};
                    else
                        out_exp <= {out_exp[0], out_exp[5:1]};
                end
                endcase
            end
        end
    end
```

2. Testbench:



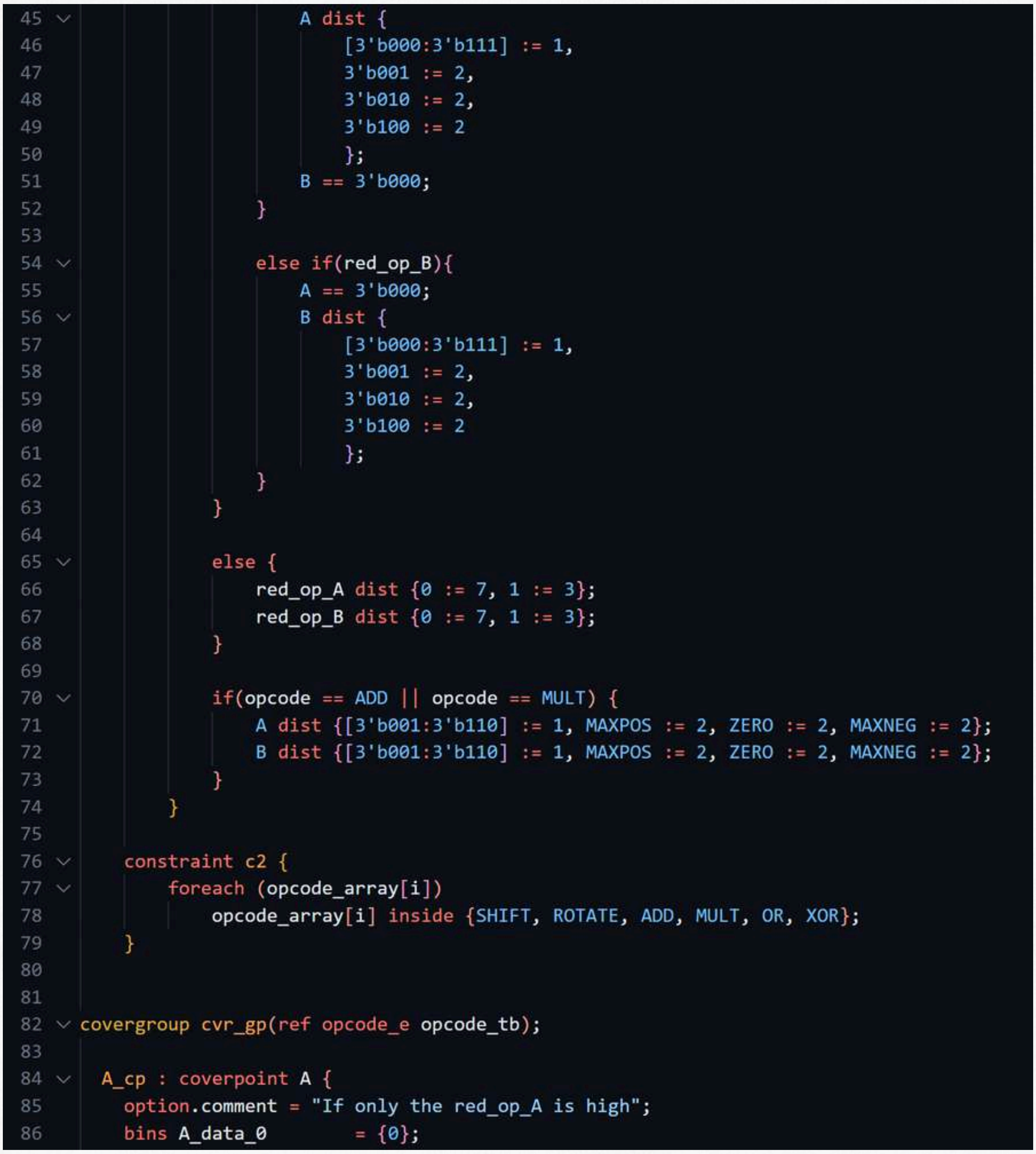
```
    task check_result();
        @(negedge clk);
        @(negedge clk);
        if(out != out_exp || leds != leds_exp) begin
            $display("*** ERROR! at time %0t, out = %0d, Expected : %0d, leds = %0d, Expected : %0d ***"
                , $time, out, out_exp, leds, leds_exp);
            error_count++;
        end
        else
            correct_count++;

    endtask

    always @(posedge clk) @(posedge clk)
        if (!rst && !(bypass_A || bypass_B))
            my_inputs.cvr_gp.sample();

endmodule
```

3. Package :



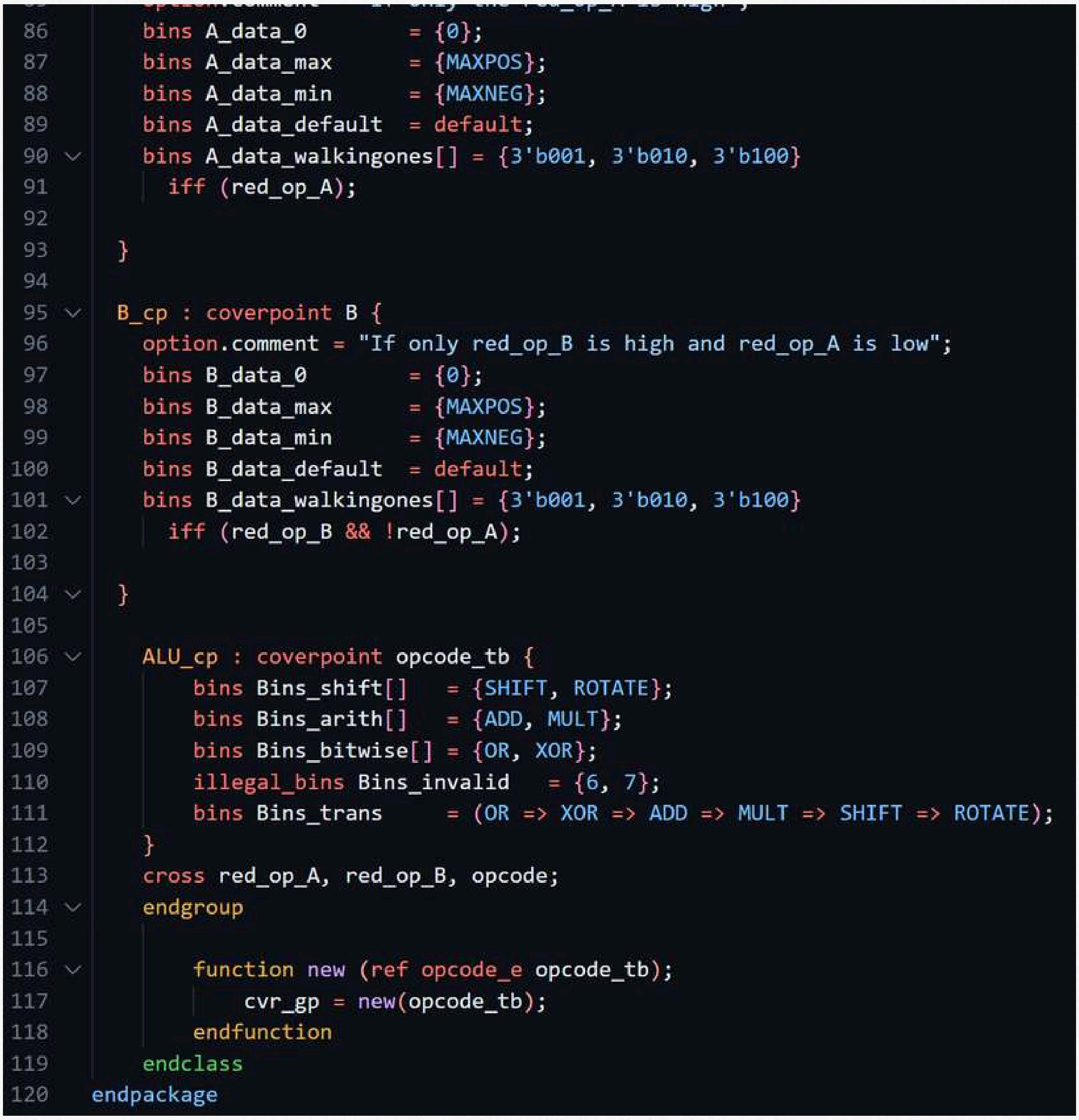
```
package ALSU_pkg;

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

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

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

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

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


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

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

            if(opcode == OR || opcode == XOR) {
                if(red_op_A) { //priority for A so red_op_B here doesn't matter
                    A dist {
```

3. Package :



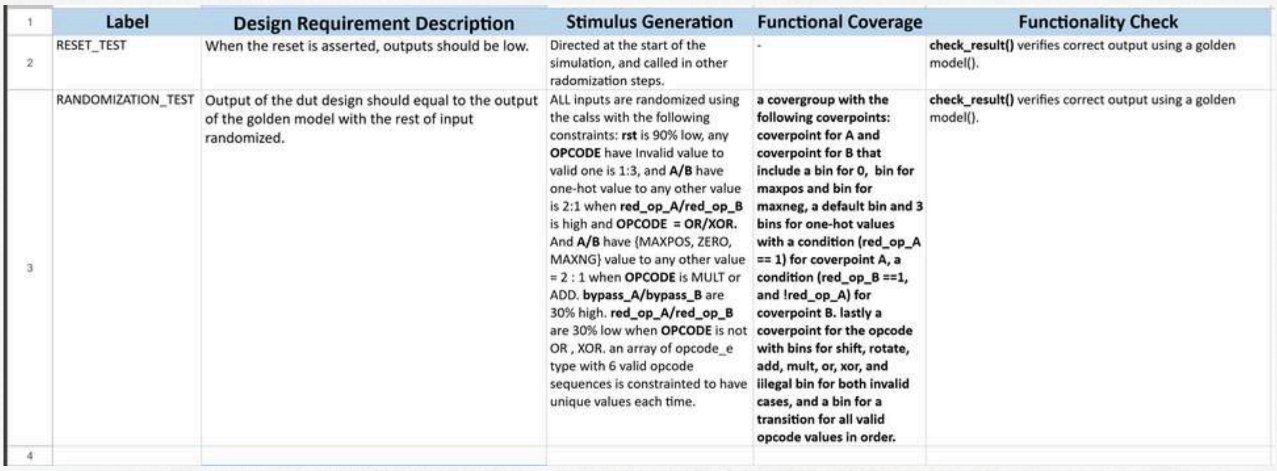
```
                        A dist {
                            [3'b000:3'b111] := 1,
                            3'b001 := 2,
                            3'b010 := 2,
                            3'b100 := 2
                            };
                        B == 3'b000;
                    }

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

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

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

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


covergroup cvr_gp(ref opcode_e opcode_tb);

  A_cp : coverpoint A {
    option.comment = "If only the red_op_A is high";
    bins A_data_0         = {0};
```

3. Package :



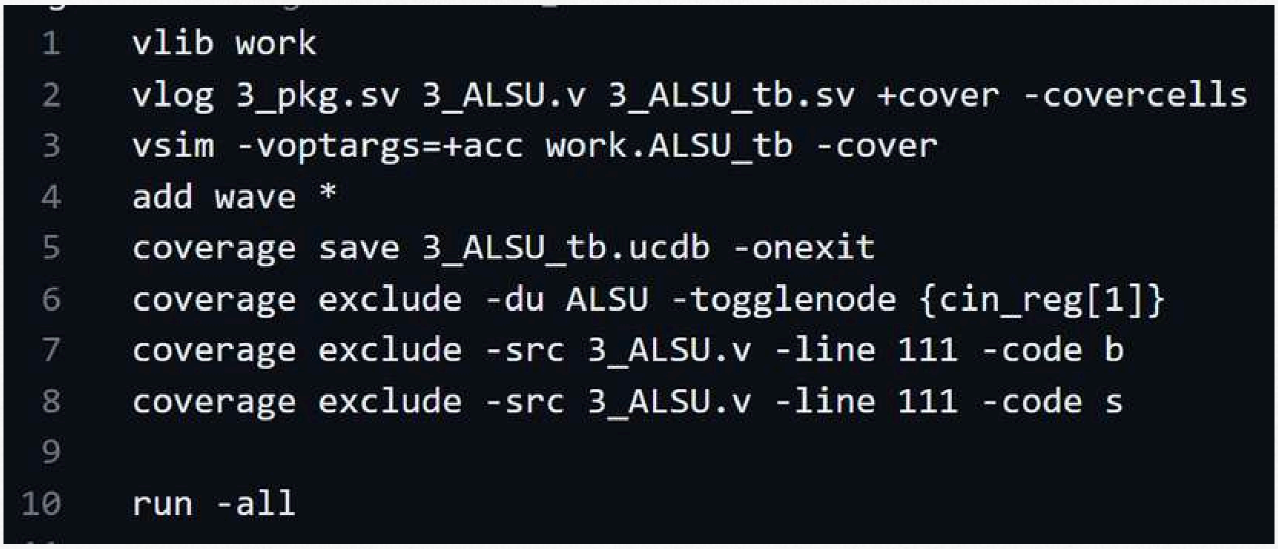
```
    bins A_data_0         = {0};
    bins A_data_max       = {MAXPOS};
    bins A_data_min       = {MAXNEG};
    bins A_data_default   = default;
    bins A_data_walkingones[] = {3'b001, 3'b010, 3'b100}
      iff (red_op_A);

  }

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

  }

    ALU_cp : coverpoint opcode_tb {
        bins Bins_shift[]   = {SHIFT, ROTATE};
        bins Bins_arith[]   = {ADD, MULT};
        bins Bins_bitwise[] = {OR, XOR};
        illegal_bins Bins_invalid   = {6, 7};
        bins Bins_trans     = (OR => XOR => ADD => MULT => SHIFT => ROTATE);
    }
    cross red_op_A, red_op_B, opcode;
    endgroup

        function new (ref opcode_e opcode_tb);
            cvr_gp = new(opcode_tb);
        endfunction
    endclass
endpackage
```

4. Verification plan



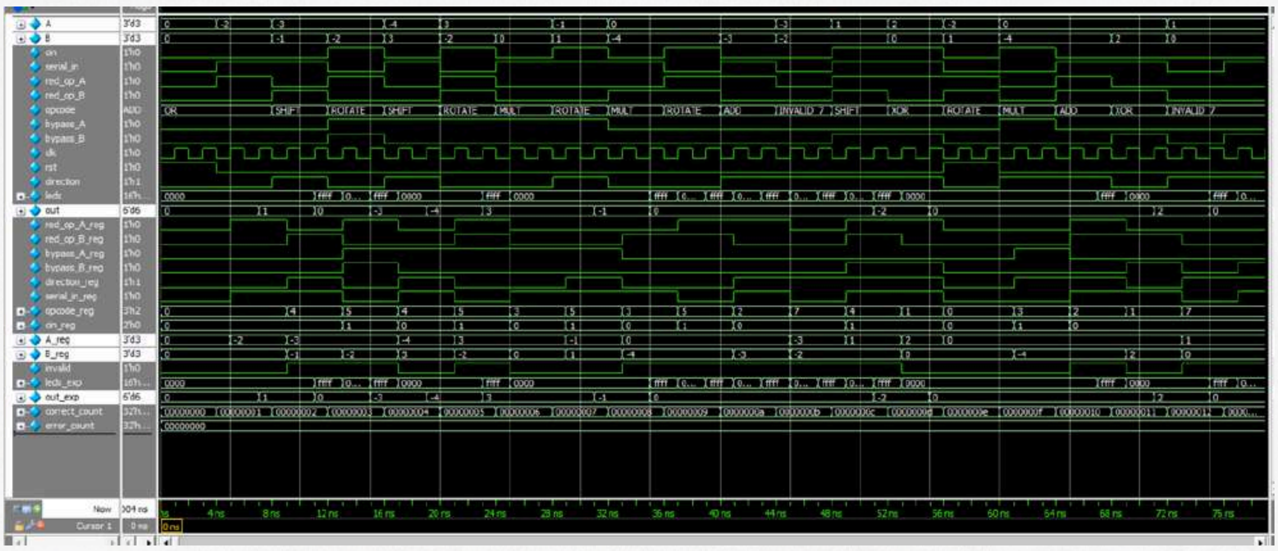
Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET_TEST	When the reset is asserted, outputs should be low.	Directed at the start of the simulation, and called in other radomization steps.	-	**check_result()** verifies correct output using a golden model().
RANDOMIZATION_TEST	Output of the dut design should equal to the output of the golden model with the rest of input randomized.	ALL inputs are randomized using the calss with the following constraints: **rst** is 90% low, any **OPCODE** have Invalid value to valid one is 1:3, and **A/B** have one-hot value to any other value is 2:1 when **red_op_A/red_op_B** is high and **OPCODE = OR/XOR.** And **A/B** have {MAXPOS, ZERO, MAXNG} value to any other value = 2 : 1 when **OPCODE** is MULT or ADD. **bypass_A/bypass_B** are 30% high. **red_op_A/red_op_B** are 30% low when **OPCODE** is not OR , XOR. an array of opcode_e type with 6 valid opcode sequences is constrainted to have unique values each time.	**a covergroup with the following coverpoints: coverpoint for A and coverpoint for B that include a bin for 0, bin for maxpos and bin for maxneg, a default bin and 3 bins for one-hot values with a condition (red_op_A == 1) for coverpoint A, a condition (red_op_B ==1, and !red_op_A) for coverpoint B. lastly a coverpoint for the opcode with bins for shift, rotate, add, mult, or, xor, and iilegal bin for both invalid cases, and a bin for a transition for all valid opcode values in order.**	**check_result()** verifies correct output using a golden model().

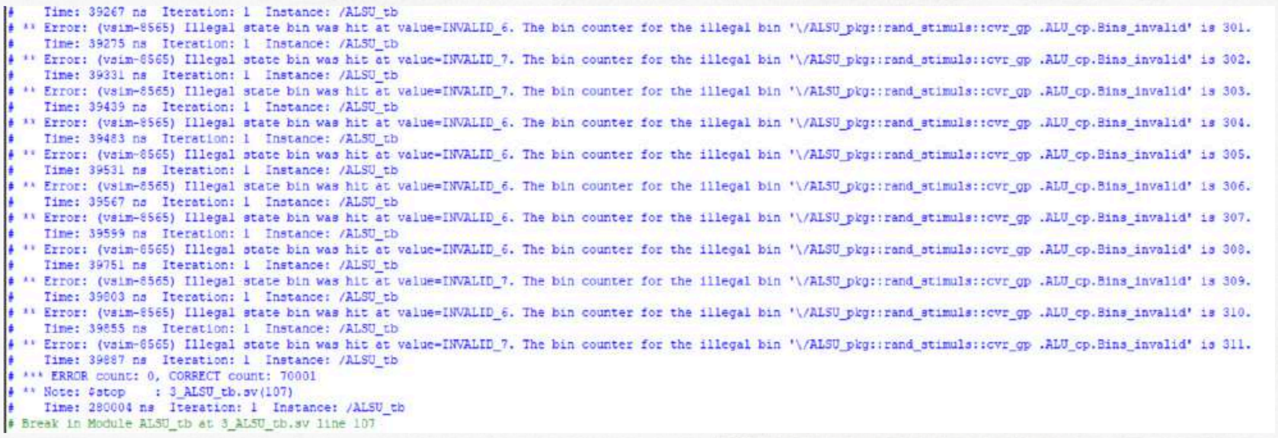
5.Do file:



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

run -all
```

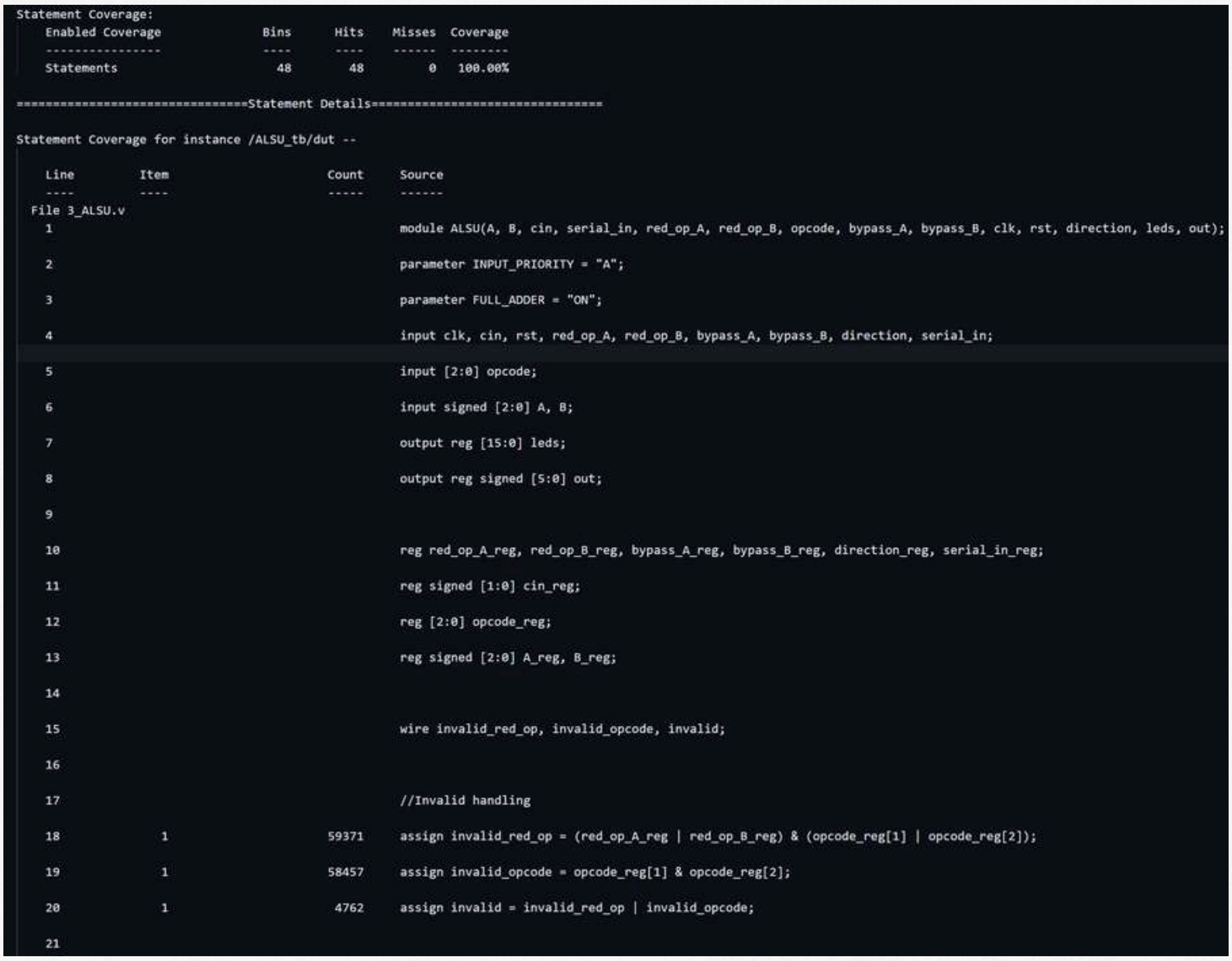
6. Wave snippets :





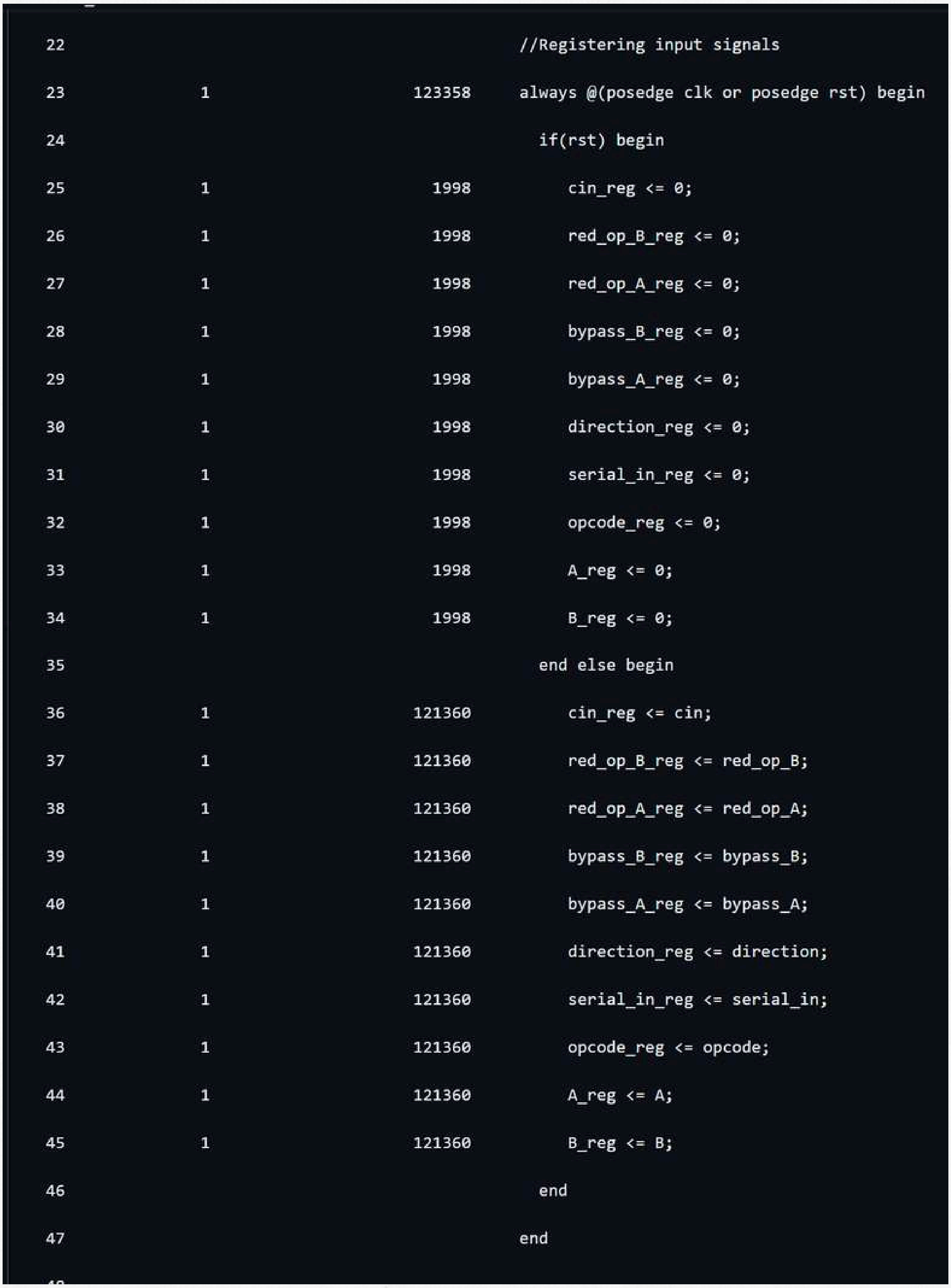
```
#    Time: 39267 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565) Illegal state bin was hit at value=INVALID_6. The bin counter for the illegal bin '\/ALSU_pkg::rand_stimuls::cvr_gp .ALU_cp.Bins_invalid' is 301.
#    Time: 39275 ns  Iteration: 1  Instance: /ALSU_tb
# ** Error: (vsim-8565)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

7. Statement coverage :



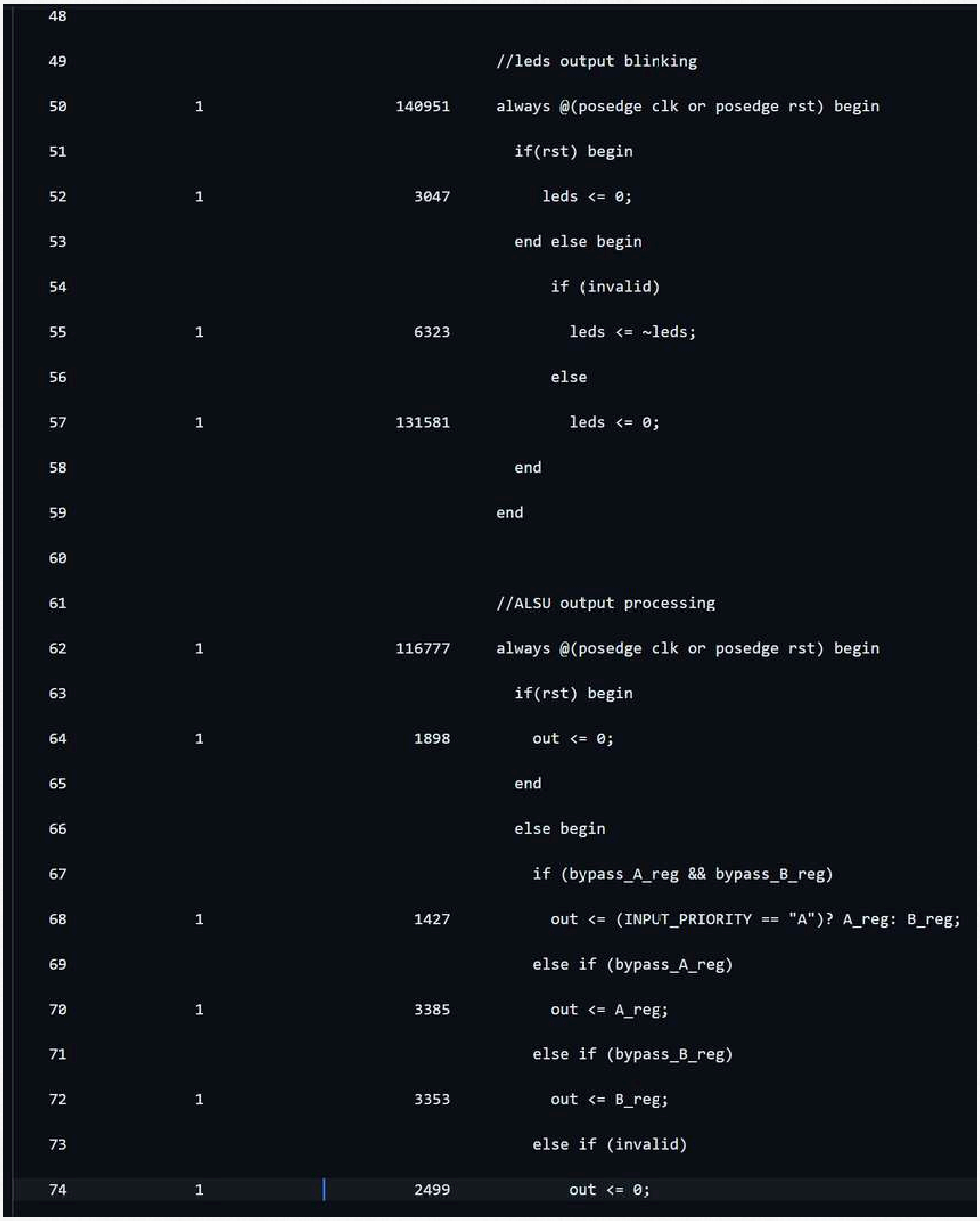
```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      48        48         0   100.00%

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

Statement Coverage for instance /ALSU_tb/dut --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File 3_ALSU.v
    1                                                module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);

    2                                                parameter INPUT_PRIORITY = "A";

    3                                                parameter FULL_ADDER = "ON";

    4                                                input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;

    5                                                input [2:0] opcode;

    6                                                input signed [2:0] A, B;

    7                                                output reg [15:0] leds;

    8                                                output reg signed [5:0] out;

    9

    10                                               reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;

    11                                               reg signed [1:0] cin_reg;

    12                                               reg [2:0] opcode_reg;

    13                                               reg signed [2:0] A_reg, B_reg;

    14

    15                                               wire invalid_red_op, invalid_opcode, invalid;

    16

    17                                               //Invalid handling

    18              1                      59371     assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);

    19              1                      58457     assign invalid_opcode = opcode_reg[1] & opcode_reg[2];

    20              1                       4762     assign invalid = invalid_red_op | invalid_opcode;

    21
```

7. Statement coverage :



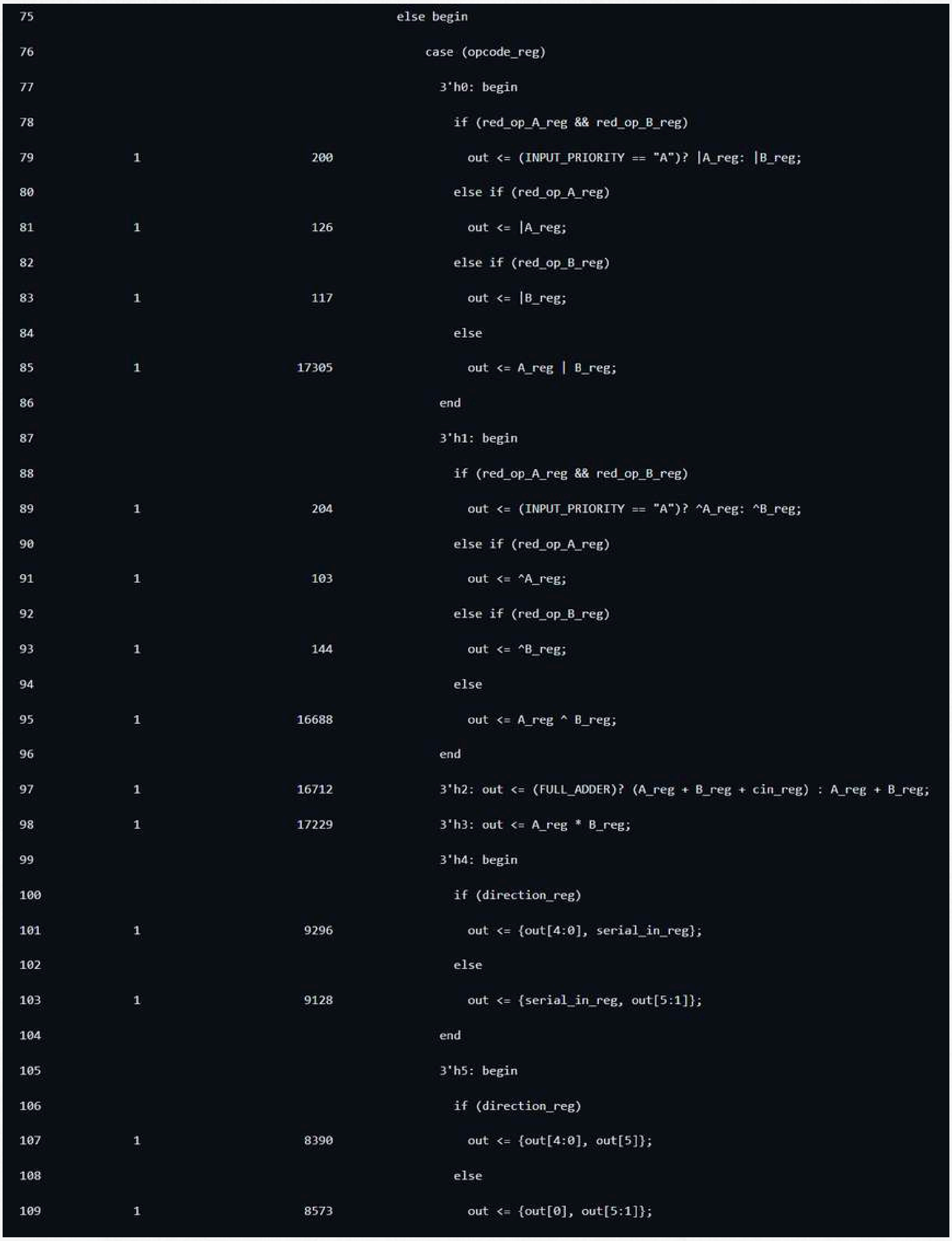
```
22                                        //Registering input signals
23          1                  123358     always @(posedge clk or posedge rst) begin
24                                          if(rst) begin
25          1                    1998          cin_reg <= 0;
26          1         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

7. Statement coverage :



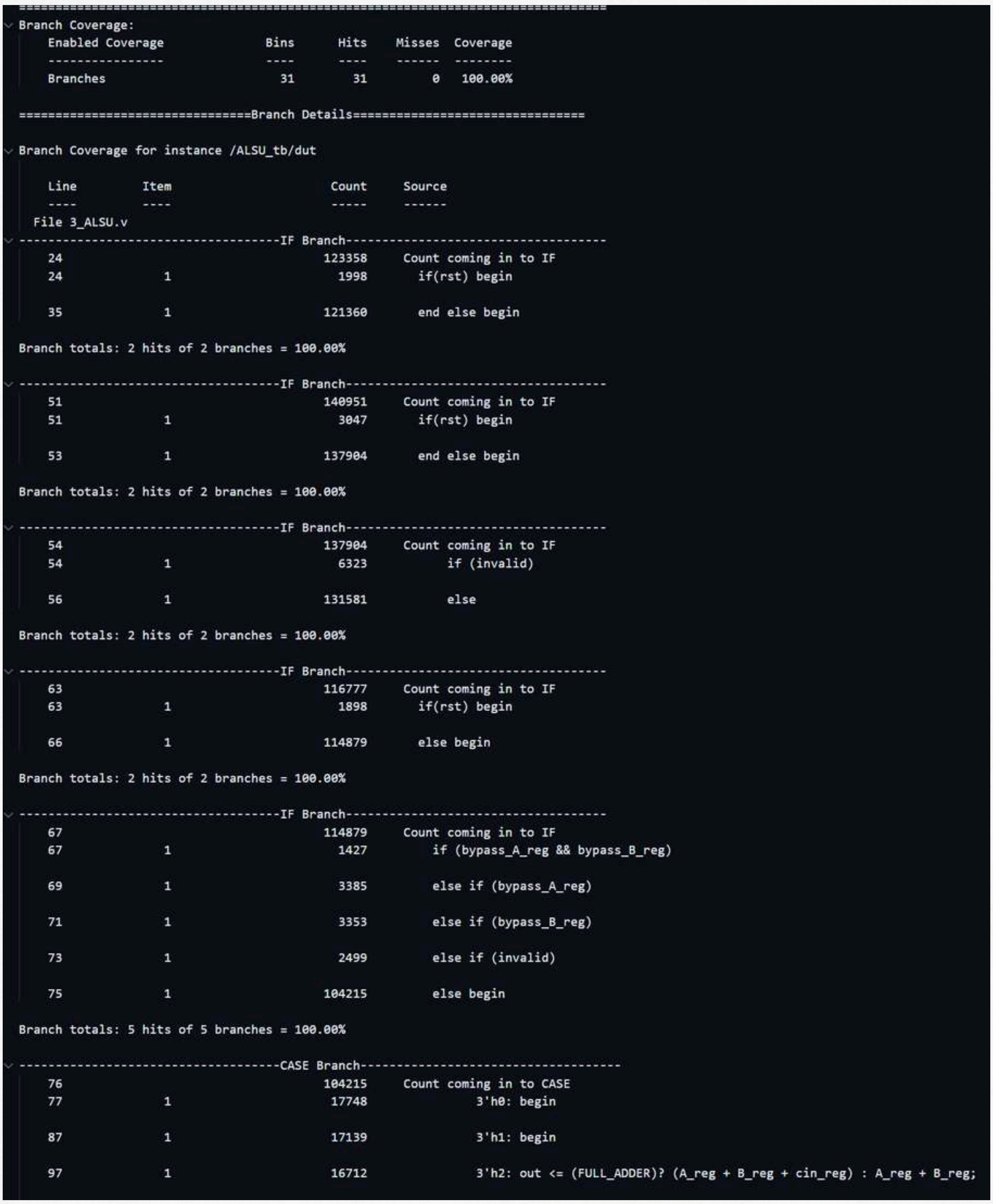
```
48
49                                     //leds output blinking
50         1               140951      always @(posedge clk or posedge rst) begin
51                                       if(rst) begin
52         1                 3047           leds <= 0;
53                                     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

7. Statement coverage :



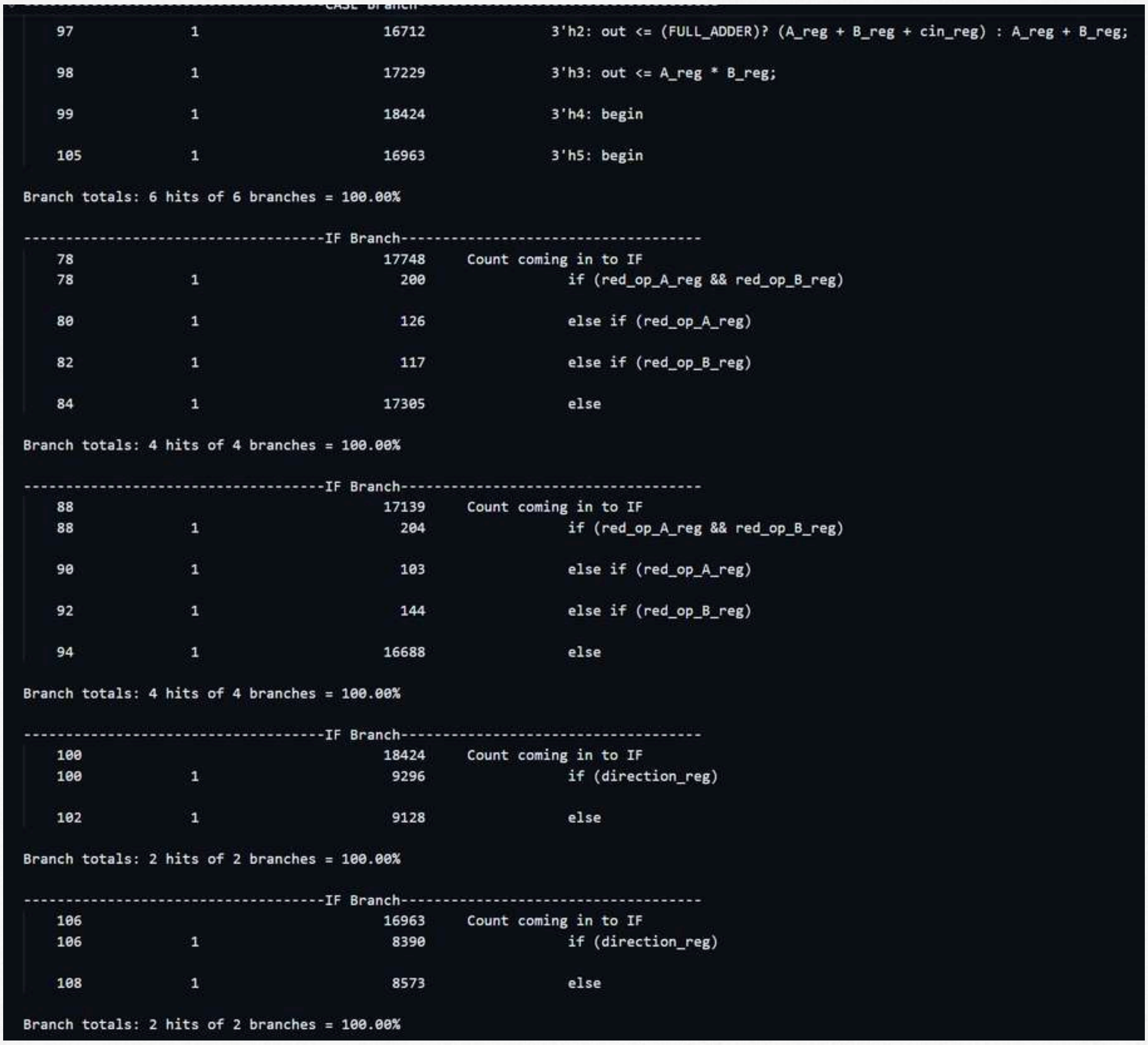
```
75                              else begin
76                                case (opcode_reg)
77                                  3'h0: begin
78                                    if (red_op_A_reg && red_op_B_reg)
79     1        200                     out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;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

8. Branch coverage :



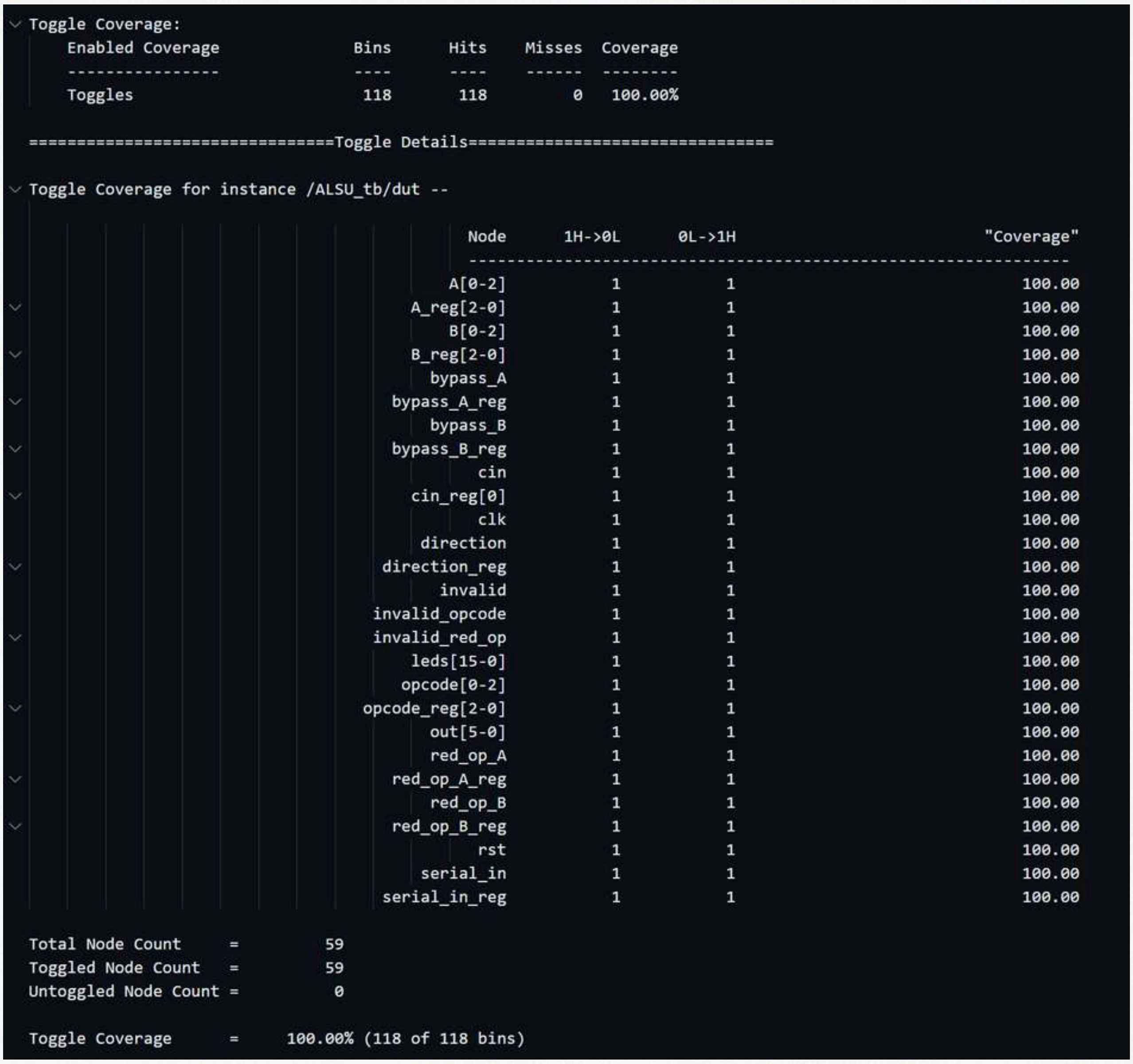
```
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        31        31         0  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

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

Branch Coverage for instance /ALSU_tb/dut

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

    35              1                     121360       end else begin

Branch totals: 2 hits of 2 branches = 100.00%

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

    53              1                     137904       end else begin

Branch totals: 2 hits of 2 branches = 100.00%

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

    56              1                     131581           else

Branch totals: 2 hits of 2 branches = 100.00%

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

    66              1                     114879       else begin

Branch totals: 2 hits of 2 branches = 100.00%

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

    69              1                       3385         else if (bypass_A_reg)

    71              1                       3353         else if (bypass_B_reg)

    73              1                       2499         else if (invalid)

    75              1                     104215         else begin

Branch totals: 5 hits of 5 branches = 100.00%

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

    87              1                      17139              3'h1: begin

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

8. Branch coverage :



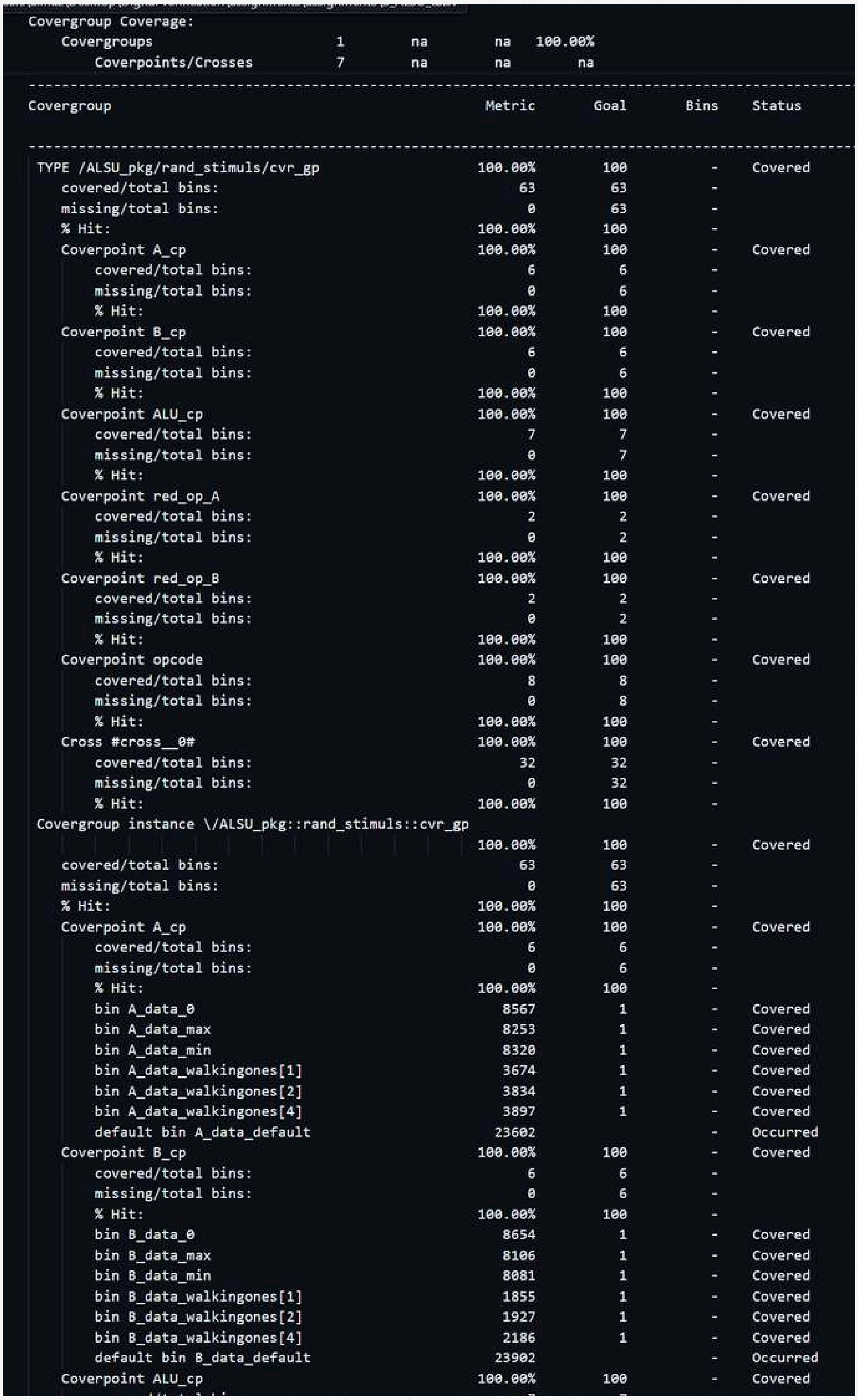
```
-----------------------------------CASE Branch-----------------------------------
    97          1                    16712          3'h2: out <= (FULL_ADDER)? (A_reg + B_reg + cin_reg) : A_reg + B_reg;

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

    99          1                    18424          3'h4: begin

    105         1                    16963          3'h5: begin

Branch totals: 6 hits of 6 branches = 100.00%

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

    80          1                      126              else if (red_op_A_reg)

    82          1                      117              else if (red_op_B_reg)

    84          1                    17305              else

Branch totals: 4 hits of 4 branches = 100.00%

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

    90          1                      103              else if (red_op_A_reg)

    92          1                      144              else if (red_op_B_reg)

    94          1                    16688              else

Branch totals: 4 hits of 4 branches = 100.00%

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

    102         1                     9128              else

Branch totals: 2 hits of 2 branches = 100.00%

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

    108         1                     8573              else

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

9. Toggle coverage:



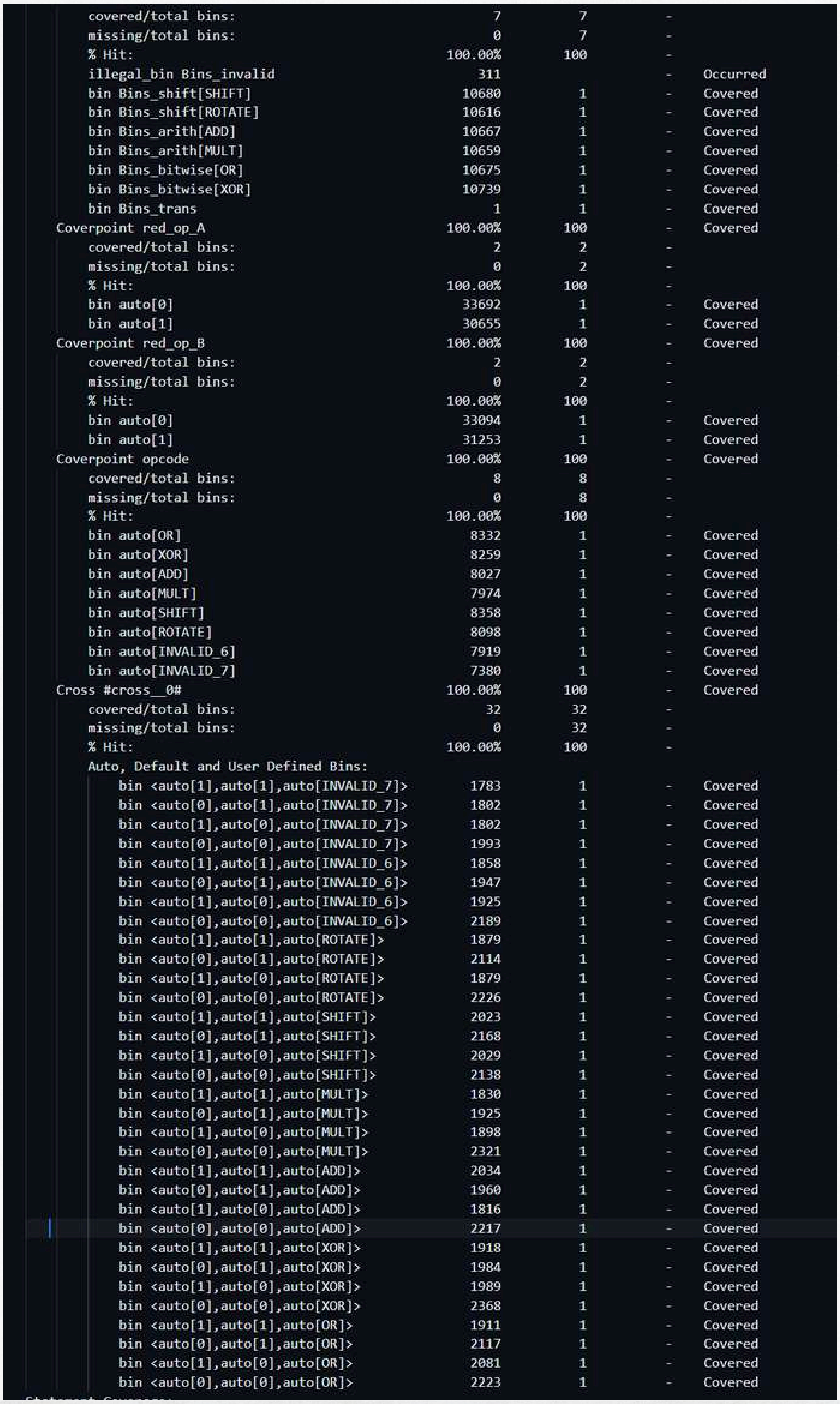
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        118       118         0   100.00%

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

Toggle Coverage for instance /ALSU_tb/dut --

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

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

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

10. Functional coverage:



```
Covergroup Coverage:
    Covergroups                      1        na         na   100.00%
        Coverpoints/Crosses          7        na         na        na
------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

------------------------------------------------------------------------------------------
 TYPE /ALSU_pkg/rand_stimuls/cvr_gp                100.00%        100          -    Covered
    covered/total bins:                                 63         63          -
    missing/total bins:                                  0         63          -
    % Hit:                                         100.00%        100          -
    Coverpoint A_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint B_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
    Coverpoint ALU_cp                              100.00%        100          -    Covered
        covered/total bins:                              7          7          -
        missing/total bins:                              0          7          -
        % Hit:                                     100.00%        100          -
    Coverpoint red_op_A                            100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint red_op_B                            100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
    Coverpoint opcode                              100.00%        100          -    Covered
        covered/total bins:                              8          8          -
        missing/total bins:                              0          8          -
        % Hit:                                     100.00%        100          -
    Cross #cross__0#                               100.00%        100          -    Covered
        covered/total bins:                             32         32          -
        missing/total bins:                              0         32          -
        % Hit:                                     100.00%        100          -
 Covergroup instance \/ALSU_pkg::rand_stimuls::cvr_gp
                                                   100.00%        100          -    Covered
    covered/total bins:                                 63         63          -
    missing/total bins:                                  0         63          -
    % Hit:                                         100.00%        100          -
    Coverpoint A_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
        bin A_data_0                                  8567          1          -    Covered
        bin A_data_max                                8253          1          -    Covered
        bin A_data_min                                8320          1          -    Covered
        bin A_data_walkingones[1]                     3674          1          -    Covered
        bin A_data_walkingones[2]                     3834          1          -    Covered
        bin A_data_walkingones[4]                     3897          1          -    Covered
        default bin A_data_default                   23602                     -    Occurred
    Coverpoint B_cp                                100.00%        100          -    Covered
        covered/total bins:                              6          6          -
        missing/total bins:                              0          6          -
        % Hit:                                     100.00%        100          -
        bin B_data_0                                  8654          1          -    Covered
        bin B_data_max                                8106          1          -    Covered
        bin B_data_min                                8081          1          -    Covered
        bin B_data_walkingones[1]                     1855          1          -    Covered
        bin B_data_walkingones[2]                     1927          1          -    Covered
        bin B_data_walkingones[4]                     2186          1          -    Covered
        default bin B_data_default                   23902                     -    Occurred
    Coverpoint ALU_cp                              100.00%        100          -    Covered
```

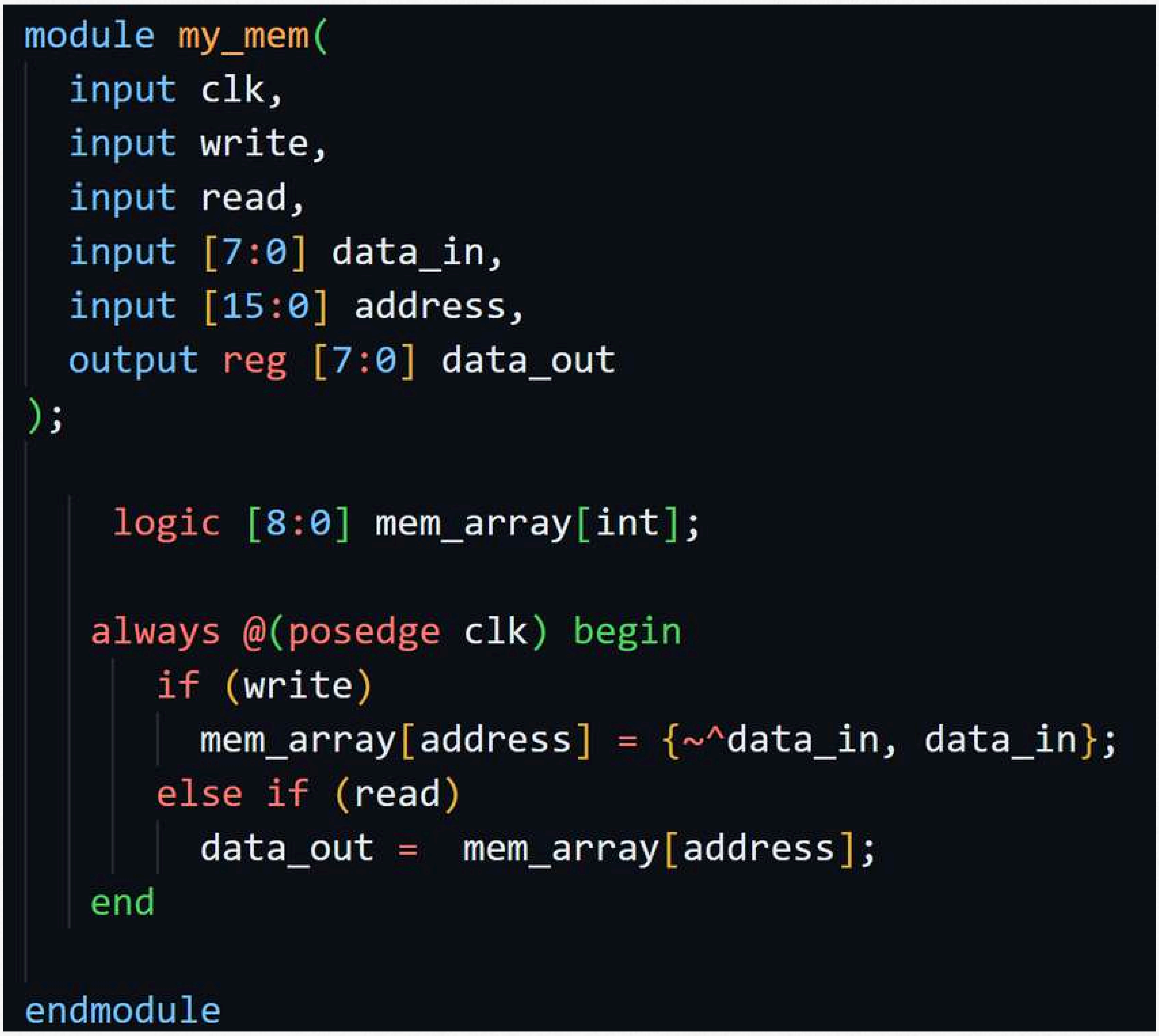
10. Functional coverage:



```
        covered/total bins:                         7          7          -
        missing/total bins:                         0          7          -
        % Hit:                                100.00%        100          -
        illegal_bin Bins_invalid                  311                     -    Occurred
        bin Bins_shift[SHIFT]                   10680          1          -    Covered
        bin Bins_shift[ROTATE]                  10616          1          -    Covered
        bin Bins_arith[ADD]                     10667          1          -    Covered
        bin Bins_arith[MULT]                    10659          1          -    Covered
        bin Bins_bitwise[OR]                    10675          1          -    Covered
        bin Bins_bitwise[XOR]                   10739          1          -    Covered
        bin Bins_trans                              1          1          -    Covered
    Coverpoint red_op_A                       100.00%        100          -    Covered
        covered/total bins:                         2          2          -
        missing/total bins:                         0          2          -
        % Hit:                                100.00%        100          -
        bin auto[0]                             33692          1          -    Covered
        bin auto[1]                             30655          1          -    Covered
    Coverpoint red_op_B                       100.00%        100          -    Covered
        covered/total bins:                         2          2          -
        missing/total bins:                         0          2          -
        % Hit:                                100.00%        100          -
        bin auto[0]                             33094          1          -    Covered
        bin auto[1]                             31253          1          -    Covered
    Coverpoint opcode                         100.00%        100          -    Covered
        covered/total bins:                         8          8          -
        missing/total bins:                         0          8          -
        % Hit:                                100.00%        100          -
        bin auto[OR]                             8332          1          -    Covered
        bin auto[XOR]                            8259          1          -    Covered
        bin auto[ADD]                            8027          1          -    Covered
        bin auto[MULT]                           7974          1          -    Covered
        bin auto[SHIFT]                          8358          1          -    Covered
        bin auto[ROTATE]                         8098          1          -    Covered
        bin auto[INVALID_6]                      7919          1          -    Covered
        bin auto[INVALID_7]                      7380          1          -    Covered
    Cross #cross__0#                          100.00%        100          -    Covered
        covered/total bins:                        32         32          -
        missing/total bins:                         0         32          -
        % Hit:                                100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <auto[1],auto[1],auto[INVALID_7]>    1783          1          -    Covered
            bin <auto[0],auto[1],auto[INVALID_7]>    1802          1          -    Covered
            bin <auto[1],auto[0],auto[INVALID_7]>    1802          1          -    Covered
            bin <auto[0],auto[0],auto[INVALID_7]>    1993          1          -    Covered
            bin <auto[1],auto[1],auto[INVALID_6]>    1858          1          -    Covered
            bin <auto[0],auto[1],auto[INVALID_6]>    1947          1          -    Covered
            bin <auto[1],auto[0],auto[INVALID_6]>    1925          1          -    Covered
            bin <auto[0],auto[0],auto[INVALID_6]>    2189          1          -    Covered
            bin <auto[1],auto[1],auto[ROTATE]>       1879          1          -    Covered
            bin <auto[0],auto[1],auto[ROTATE]>       2114          1          -    Covered
            bin <auto[1],auto[0],auto[ROTATE]>       1879          1          -    Covered
            bin <auto[0],auto[0],auto[ROTATE]>       2226          1          -    Covered
            bin <auto[1],auto[1],auto[SHIFT]>        2023          1          -    Covered
            bin <auto[0],auto[1],auto[SHIFT]>        2168          1          -    Covered
            bin <auto[1],auto[0],auto[SHIFT]>        2029          1          -    Covered
            bin <auto[0],auto[0],auto[SHIFT]>        2138          1          -    Covered
            bin <auto[1],auto[1],auto[MULT]>         1830          1          -    Covered
            bin <auto[0],auto[1],auto[MULT]>         1925          1          -    Covered
            bin <auto[1],auto[0],auto[MULT]>         1898          1          -    Covered
            bin <auto[0],auto[0],auto[MULT]>         2321          1          -    Covered
            bin <auto[1],auto[1],auto[ADD]>          2034          1          -    Covered
            bin <auto[0],auto[1],auto[ADD]>          1960          1          -    Covered
            bin <auto[1],auto[0],auto[ADD]>          1816          1          -    Covered
            bin <auto[0],auto[0],auto[ADD]>          2217          1          -    Covered
            bin <auto[1],auto[1],auto[XOR]>          1918          1          -    Covered
            bin <auto[0],auto[1],auto[XOR]>          1984          1          -    Covered
            bin <auto[1],auto[0],auto[XOR]>          1989          1          -    Covered
            bin <auto[0],auto[0],auto[XOR]>          2368          1          -    Covered
            bin <auto[1],auto[1],auto[OR]>           1911          1          -    Covered
            bin <auto[0],auto[1],auto[OR]>           2117          1          -    Covered
            bin <auto[1],auto[0],auto[OR]>           2081          1          -    Covered
            bin <auto[0],auto[0],auto[OR]>           2223          1          -    Covered
```

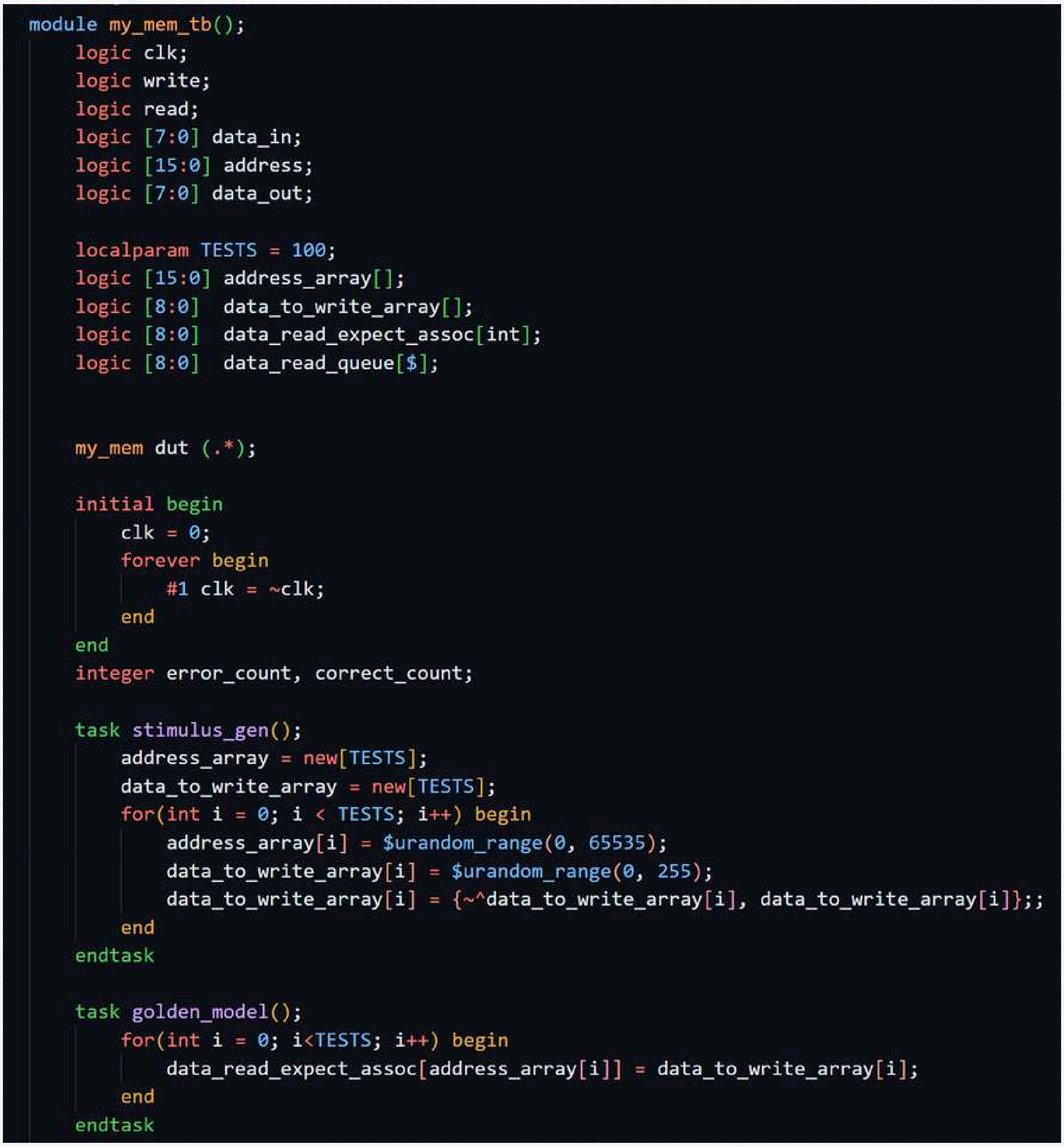
Question 4:

1. RTL design:



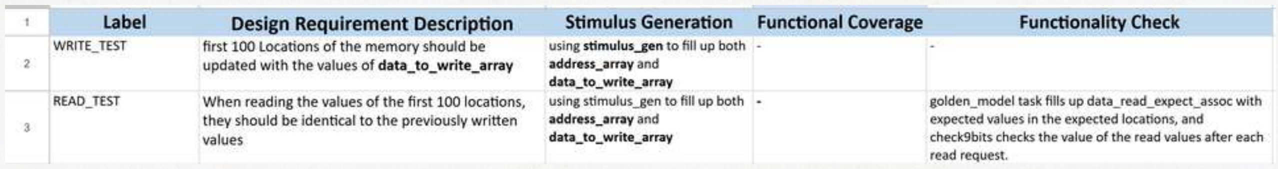
```
module my_mem(
  input clk,
  input write,
  input read,
  input [7:0] data_in,
  input [15:0] address,
  output reg [7:0] data_out
);

   logic [8:0] mem_array[int];

  always @(posedge clk) begin
     if (write)
       mem_array[address] = {~^data_in, data_in};
     else if (read)
       data_out =  mem_array[address];
  end

endmodule
```

2.Testbench:



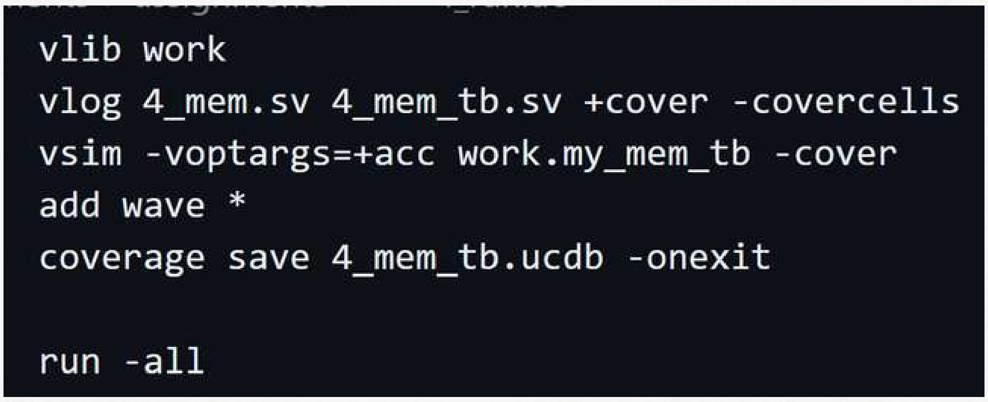
```
module my_mem_tb();
    logic clk;
    logic write;
    logic read;
    logic [7:0] data_in;
    logic [15:0] address;
    logic [7:0] data_out;

    localparam TESTS = 100;
    logic [15:0] address_array[];
    logic [8:0]  data_to_write_array[];
    logic [8:0]  data_read_expect_assoc[int];
    logic [8:0]  data_read_queue[$];


    my_mem dut (.*);

    initial begin
        clk = 0;
        forever begin
            #1 clk = ~clk;
        end
    end
    integer error_count, correct_count;

    task stimulus_gen();
        address_array = new[TESTS];
        data_to_write_array = new[TESTS];
        for(int i = 0; i < TESTS; i++) begin
            address_array[i] = $urandom_range(0, 65535);
            data_to_write_array[i] = $urandom_range(0, 255);
            data_to_write_array[i] = {~^data_to_write_array[i], data_to_write_array[i]};;
        end
    endtask

    task golden_model();
        for(int i = 0; i<TESTS; i++) begin
            data_read_expect_assoc[address_array[i]] = data_to_write_array[i];
        end
    endtask
```

3. Verification plan:



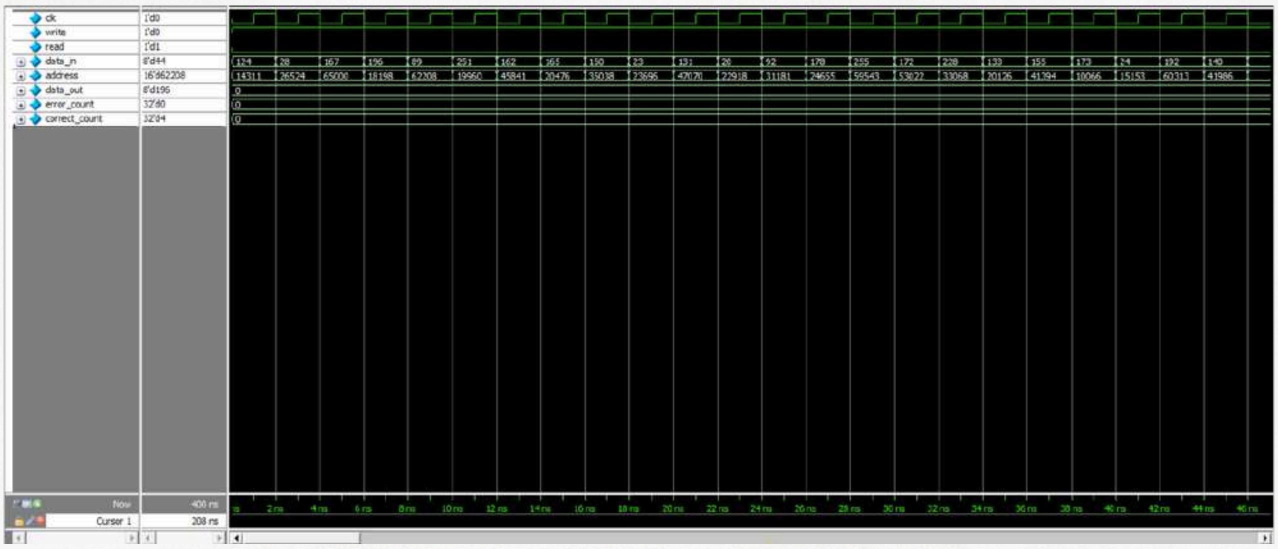
Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
WRITE_TEST	first 100 Locations of the memory should be updated with the values of **data_to_write_array**	using **stimulus_gen** to fill up both **address_array** and **data_to_write_array**	-	-
READ_TEST	When reading the values of the first 100 locations, they should be identical to the previously written values	using stimulus_gen to fill up both **address_array** and **data_to_write_array**	-	golden_model task fills up data_read_expect_assoc with expected values in the expected locations, and check9bits checks the value of the read values after each read request.

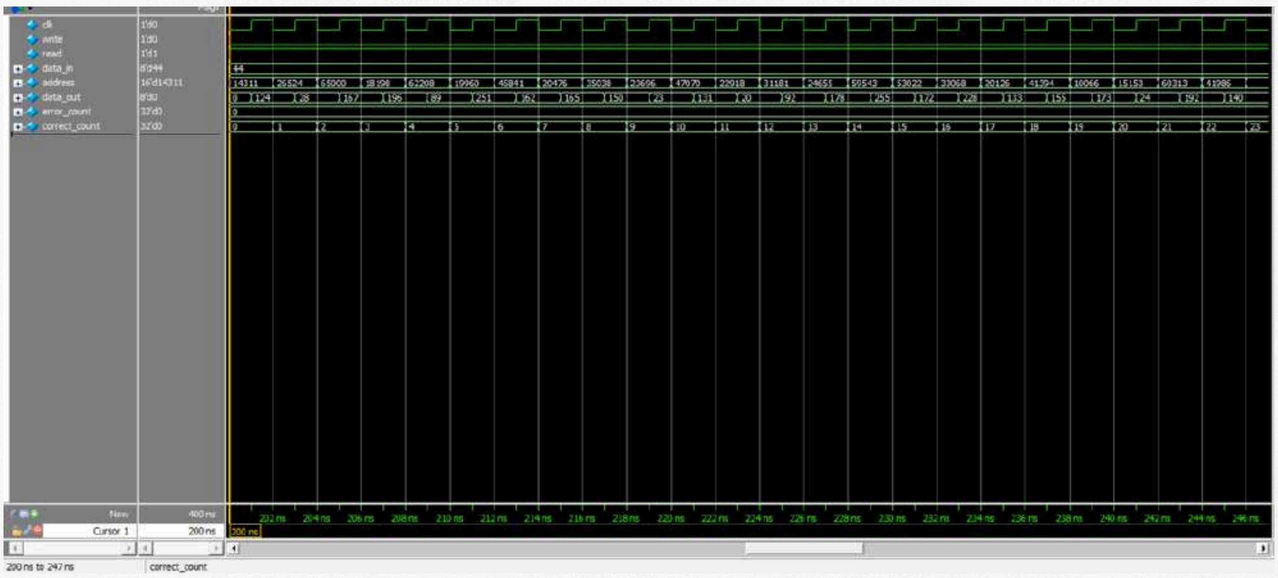
4. Do file:

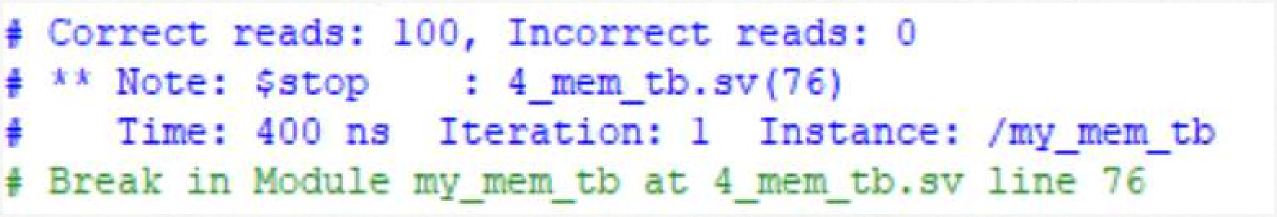


```
vlib work
vlog 4_mem.sv 4_mem_tb.sv +cover -covercells
vsim -voptargs=+acc work.my_mem_tb -cover
add wave *
coverage save 4_mem_tb.ucdb -onexit

run -all
```

5. Questa sim snippets:







```
# Correct reads: 100, Incorrect reads: 0
# ** Note: $stop    : 4_mem_tb.sv(76)
#    Time: 400 ns  Iteration: 1  Instance: /my_mem_tb
# Break in Module my_mem_tb at 4_mem_tb.sv line 76
```