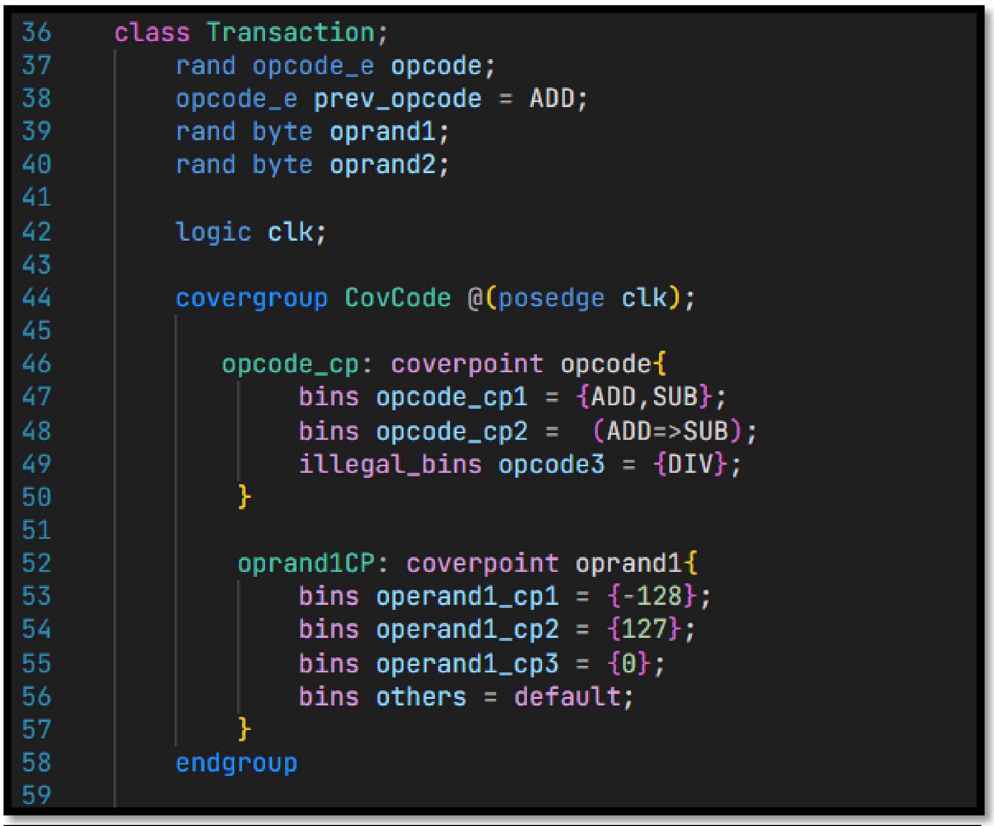
Digital Verification using SV and UVM Assignment-3

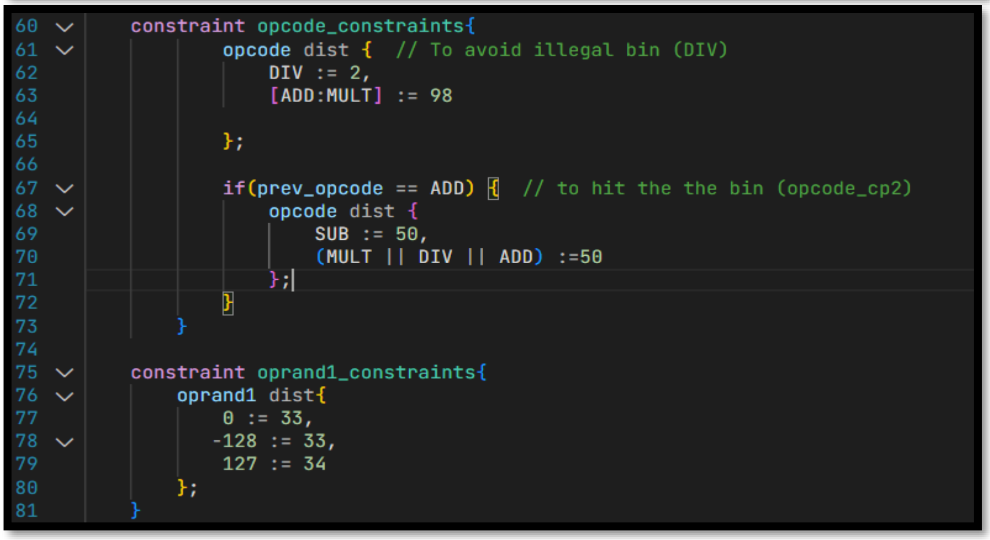
Name: Fares Khalaf Sultan

Q1) ALU:

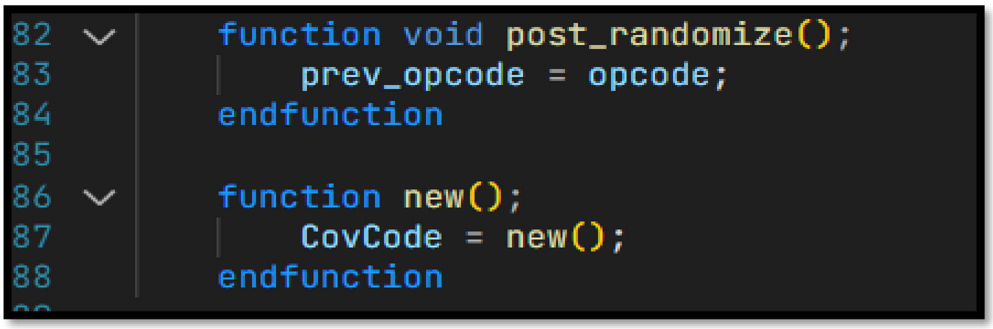
❖ Package Code:



```
class Transaction;
    rand opcode_e opcode;
    opcode_e prev_opcode = ADD;
    rand byte oprand1;
    rand byte oprand2;

    logic clk;

    covergroup CovCode @(posedge clk);

       opcode_cp: coverpoint opcode{
            bins opcode_cp1 = {ADD,SUB};
            bins opcode_cp2 =  (ADD=>SUB);
            illegal_bins opcode3 = {DIV};
        }

        oprand1CP: coverpoint oprand1{
            bins operand1_cp1 = {-128};
            bins operand1_cp2 = {127};
            bins operand1_cp3 = {0};
            bins others = default;
        }
    endgroup

```

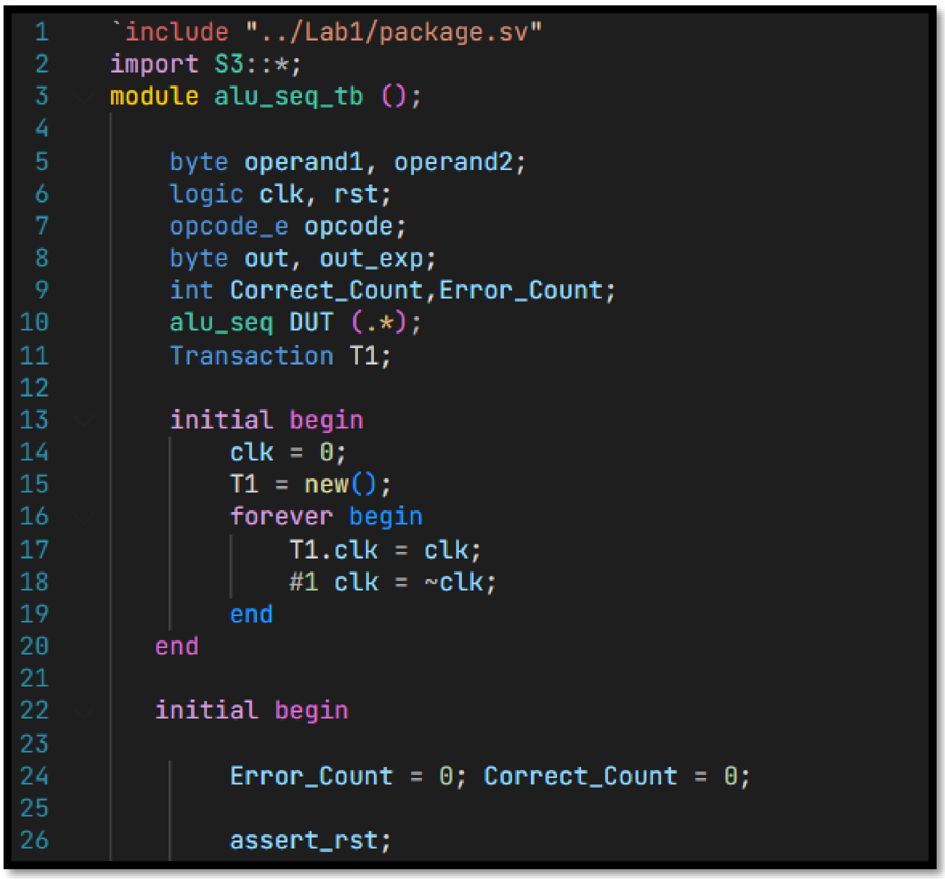


```
    constraint opcode_constraints{
            opcode dist {  // To avoid illegal bin (DIV)
                DIV := 2,
                [ADD:MULT] := 98

            };

            if(prev_opcode == ADD) {  // to hit the the bin (opcode_cp2)
                opcode dist {
                    SUB := 50,
                    (MULT || DIV || ADD) :=50
                };
            }
        }

    constraint oprand1_constraints{
        oprand1 dist{
            0 := 33,
           -128 := 33,
            127 := 34
        };
    }
```

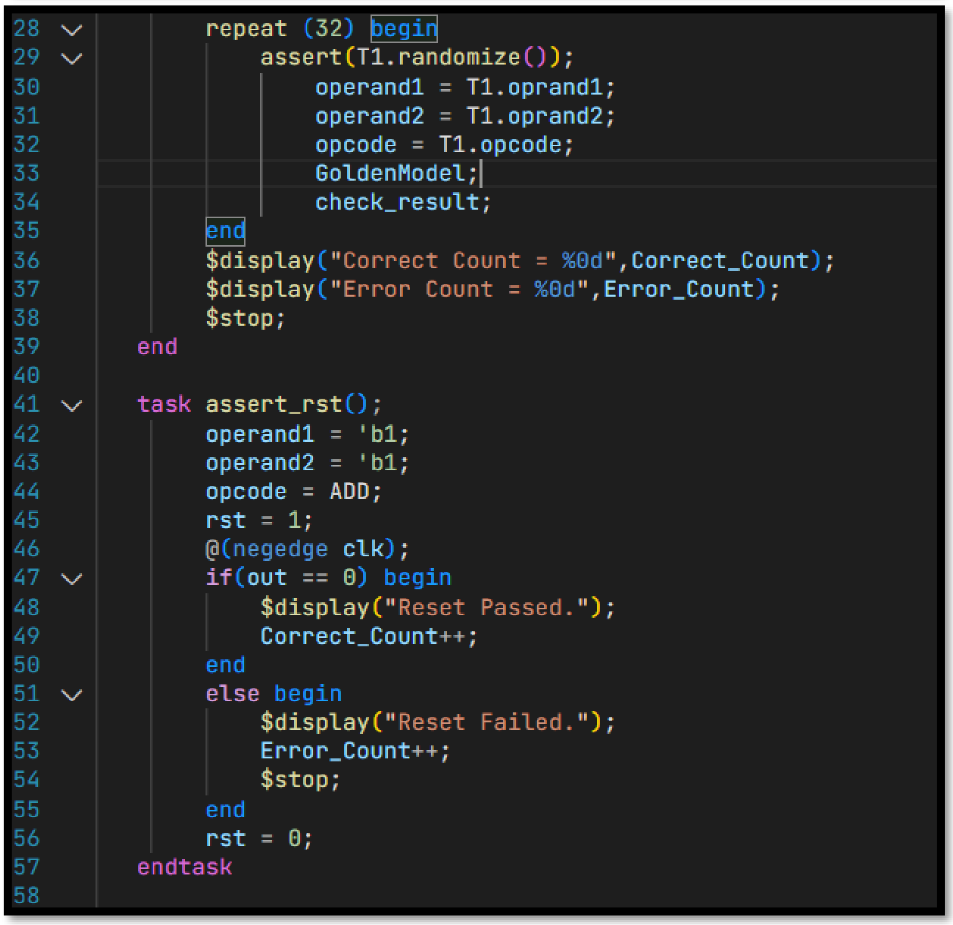


```
    function void post_randomize();
        prev_opcode = opcode;
    endfunction

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

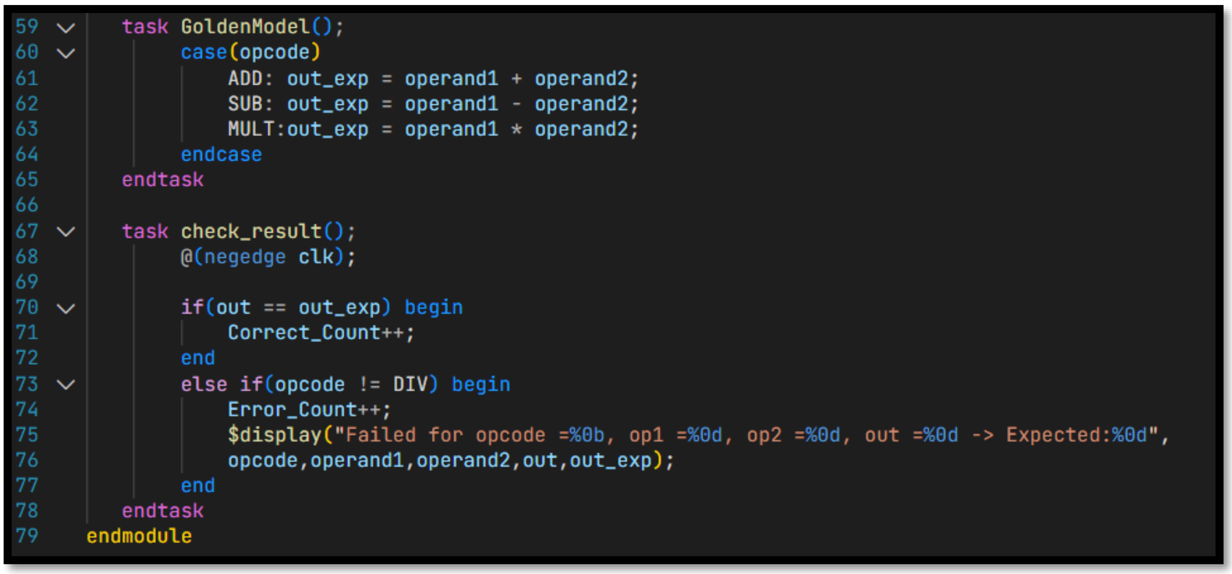
❖ **testbench:**



```systemverilog
`include "../Lab1/package.sv"
import S3::*;
module alu_seq_tb ();

    byte operand1, operand2;
    logic clk, rst;
    opcode_e opcode;
    byte out, out_exp;
    int Correct_Count,Error_Count;
    alu_seq DUT (.*);
    Transaction T1;

    initial begin
        clk = 0;
        T1 = new();
        forever begin
            T1.clk = clk;
            #1 clk = ~clk;
        end
    end

    initial begin

        Error_Count = 0; Correct_Count = 0;

        assert_rst;
```

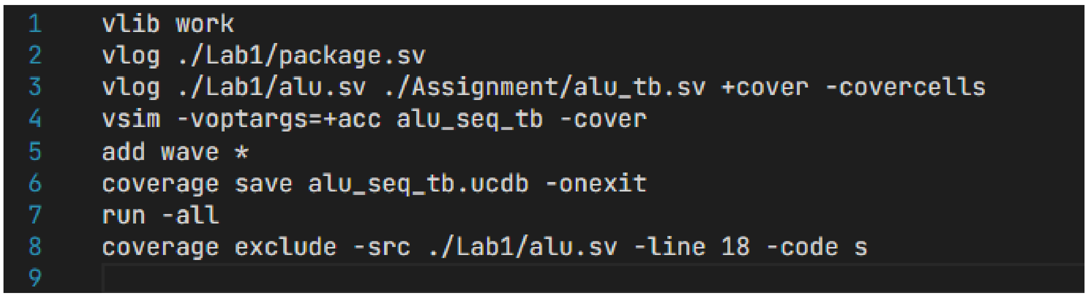


```systemverilog
        repeat (32) begin
            assert(T1.randomize());
                operand1 = T1.oprand1;
                operand2 = T1.oprand2;
                opcode = T1.opcode;
                GoldenModel;
                check_result;
        end
        $display("Correct Count = %0d",Correct_Count);
        $display("Error Count = %0d",Error_Count);
        $stop;
    end

    task assert_rst();
        operand1 = 'b1;
        operand2 = 'b1;
        opcode = ADD;
        rst = 1;
        @(negedge clk);
        if(out == 0) begin
            $display("Reset Passed.");
            Correct_Count++;
        end
        else begin
            $display("Reset Failed.");
            Error_Count++;
            $stop;
        end
        rst = 0;
    endtask

```



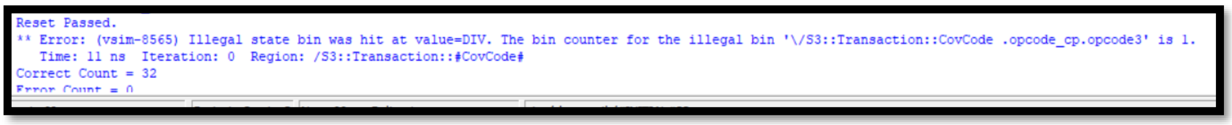
```
    task GoldenModel();
        case(opcode)
            ADD: out_exp = operand1 + operand2;
            SUB: out_exp = operand1 - operand2;
            MULT:out_exp = operand1 * operand2;
        endcase
    endtask

    task check_result();
        @(negedge clk);

        if(out == out_exp) begin
            Correct_Count++;
        end
        else if(opcode != DIV) begin
            Error_Count++;
            $display("Failed for opcode =%0b, op1 =%0d, op2 =%0d, out =%0d -> Expected:%0d",
            opcode,operand1,operand2,out,out_exp);
        end
    endtask
endmodule
```

❖ Do file:



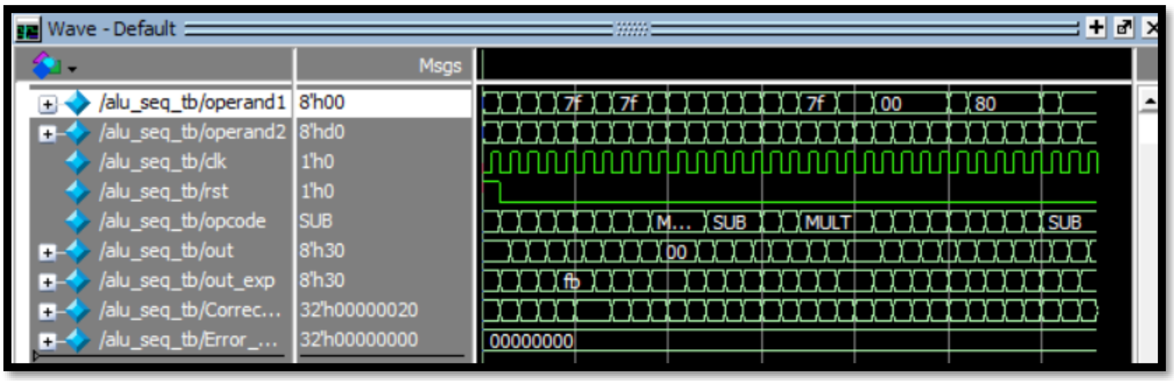
```
vlib work
vlog ./Lab1/package.sv
vlog ./Lab1/alu.sv ./Assignment/alu_tb.sv +cover -covercells
vsim -voptargs=+acc alu_seq_tb -cover
add wave *
coverage save alu_seq_tb.ucdb -onexit
run -all
coverage exclude -src ./Lab1/alu.sv -line 18 -code s
```

❖ Result:



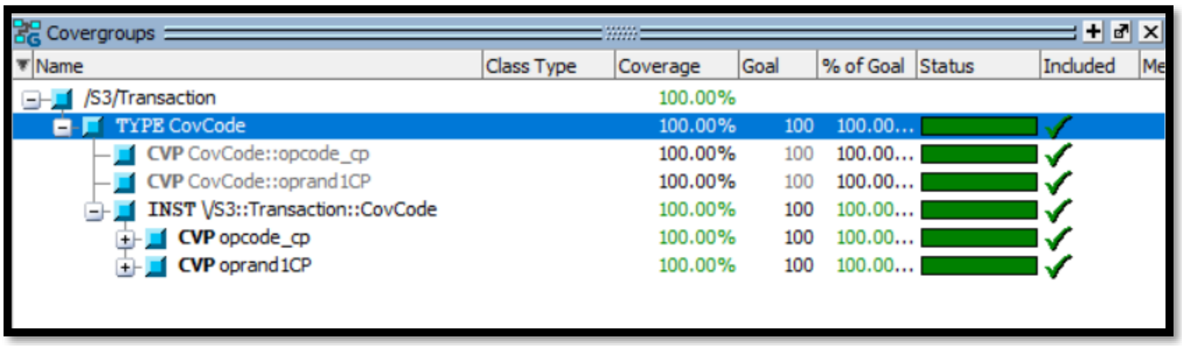
```
Reset Passed.
** Error: (vsim-8565) Illegal state bin was hit at value=DIV. The bin counter for the illegal bin '\/S3::Transaction::CovCode .opcode_cp.opcode3' is 1.
   Time: 11 ns  Iteration: 0  Region: /S3::Transaction::#CovCode#
Correct Count = 32
Error Count = 0
```

Waveform:



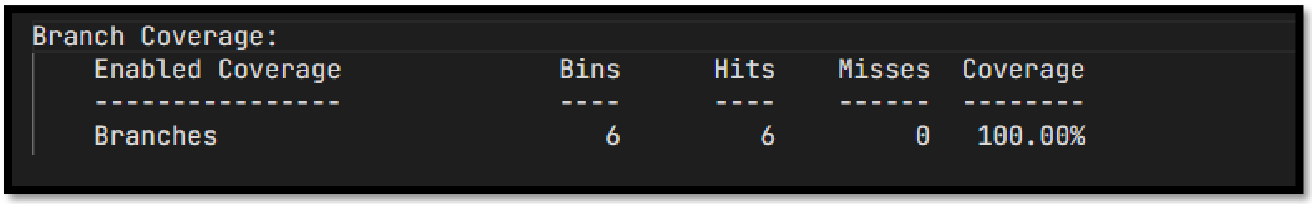
```
Wave - Default
                          Msgs
/alu_seq_tb/operand1      8'h00
/alu_seq_tb/operand2      8'hd0
/alu_seq_tb/clk           1'h0
/alu_seq_tb/rst           1'h0
/alu_seq_tb/opcode        SUB
/alu_seq_tb/out           8'h30
/alu_seq_tb/out_exp       8'h30
/alu_seq_tb/Correc...     32'h00000020
/alu_seq_tb/Error_...     32'h00000000
```

❖ Functional coverage:



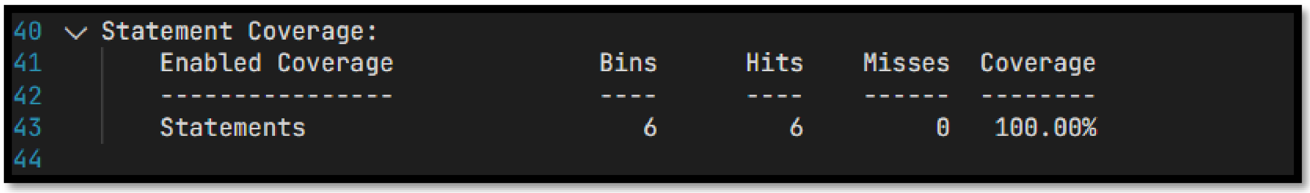
Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/S3/Transaction		100.00%				
TYPE CovCode		100.00%	100	100.00...	🟩	✓
CVP CovCode::opcode_cp		100.00%	100	100.00...	🟩	✓
CVP CovCode::oprand1CP		100.00%	100	100.00...	🟩	✓
INST \/S3::Transaction::CovCode		100.00%	100	100.00...	🟩	✓
CVP opcode_cp		100.00%	100	100.00...	🟩	✓
CVP oprand1CP		100.00%	100	100.00...	🟩	✓

❖ Code coverage:

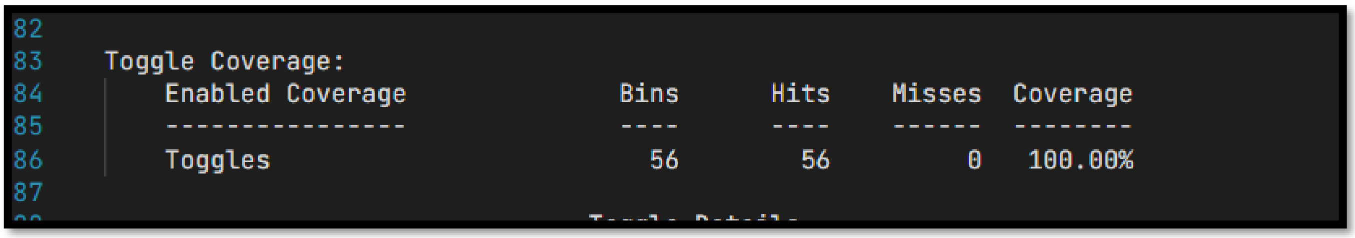


```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         6         6         0   100.00%
```

Excluded the default case for the opcode, because it is unreachable since all possible opcode values are handled in the case statement.



```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       6         6         0   100.00%
```



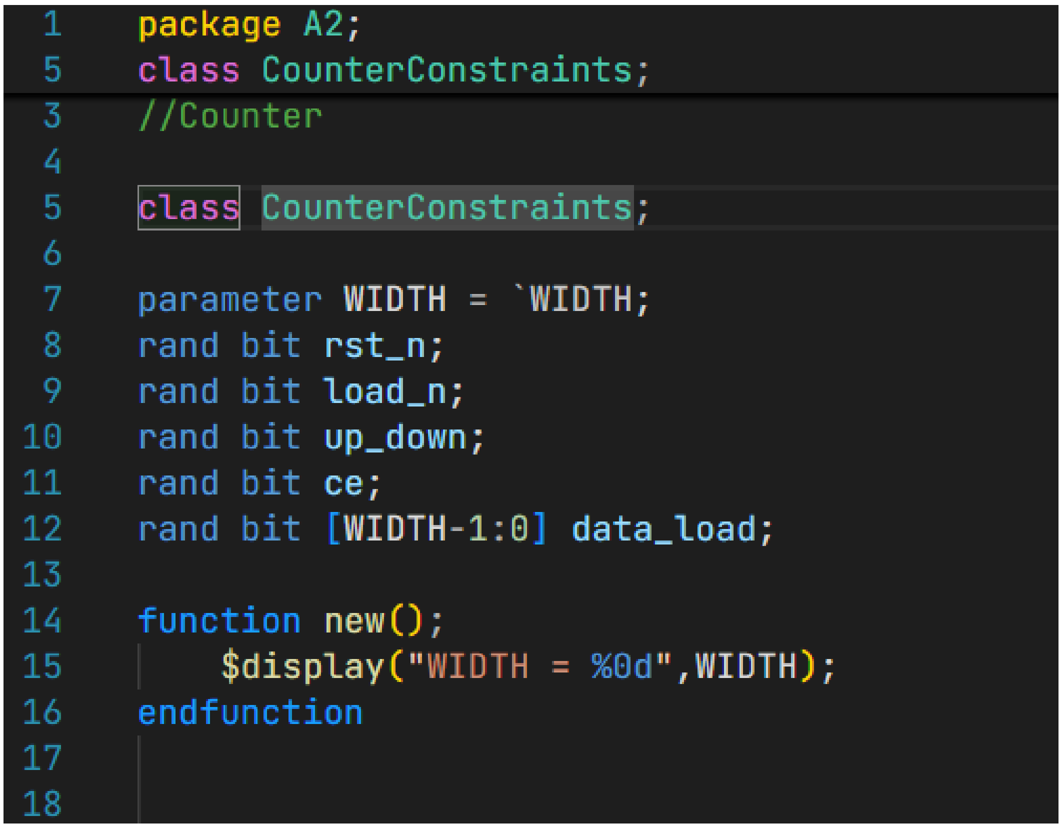
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         56        56         0   100.00%
```

Q2) Counter:

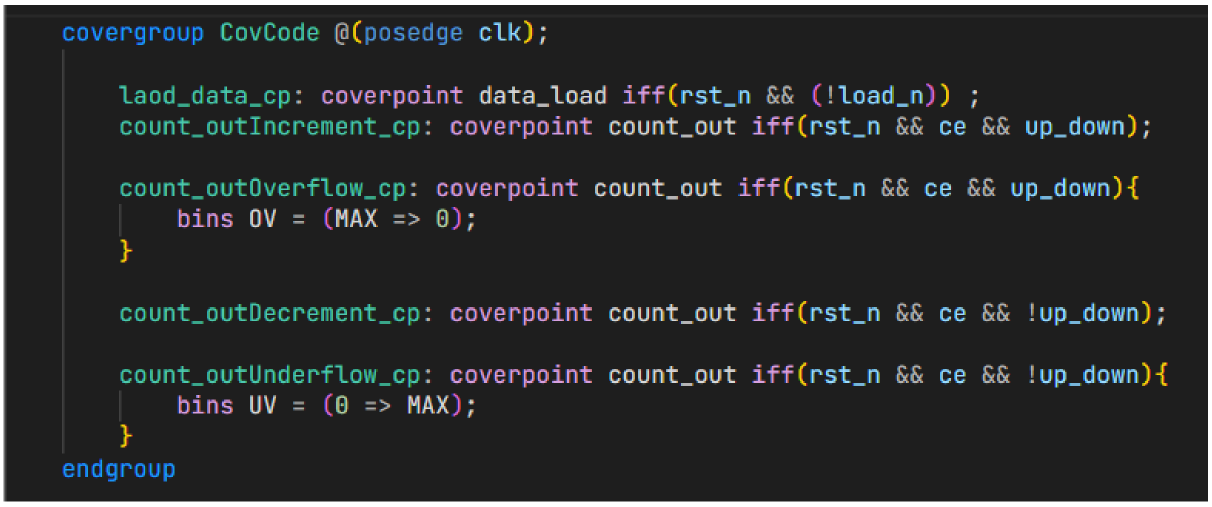
❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
Counter1	when reset is asserted, Output should be low, and **zero** should be high	Directed at the start of the simulation, then randomized with constraint to be inactive 90% of time during the simulation	-	A checker in the testbench to make sure the output is correct
Counter2	When **load_n** is low, **count_out** should take the value of **load_data** input	Randomization with constraint on **load_n** to be high 70% of simulation time	Cover all values of load_data	Output Checked against golden model
Counter3	Counter should only increment or decrement if **rst_n** is inactive and, **ce** signal is high else keep the current **count_out** value.	Randomization with constraint on **ce** to be high 70% of simulation time	Cover all values of count_out& transition bin from max to zero	Output Checked against golden model
Counter4	If rst_n is disabled and ce is enabeled, if: **up_down = o → decrement** **Up_down = 1 → increment**	Randomization with constraint on **up_down** to be high 50% of simulation time	Bin for decrement & another for increment	Output Checked against golden model
Counter5	Check that when the bus **count_out** value equals the max possible value, **max_count** should be high	Randomized with no constraints	Bin for max_count	Output Checked against golden model

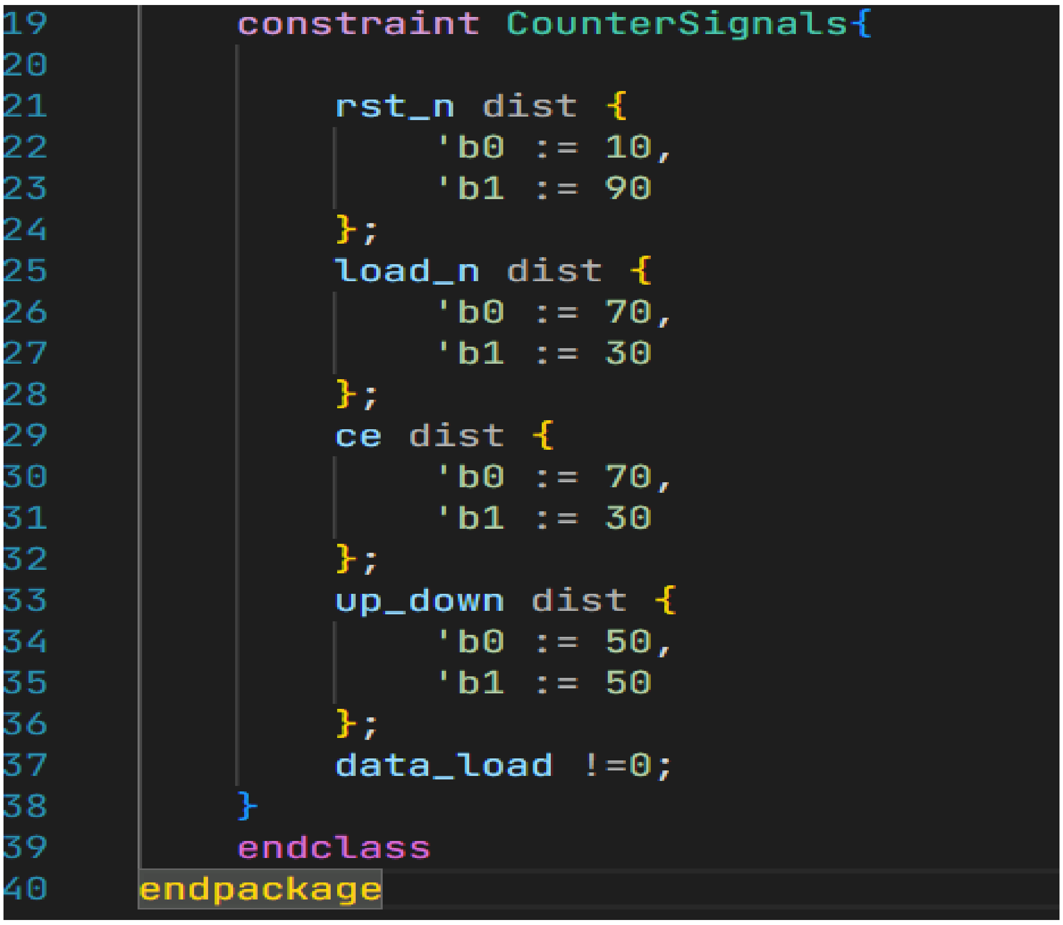
❖ Constraints class:



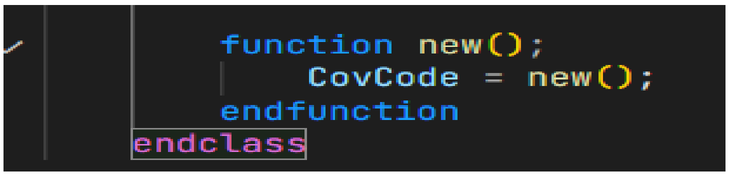
```
package A2;
class CounterConstraints;
//Counter

class CounterConstraints;

parameter WIDTH = `WIDTH;
rand bit rst_n;
rand bit load_n;
rand bit up_down;
rand bit ce;
rand bit [WIDTH-1:0] data_load;

function new();
    $display("WIDTH = %0d",WIDTH);
endfunction


```



```
covergroup CovCode @(posedge clk);

    laod_data_cp: coverpoint data_load iff(rst_n && (!load_n)) ;
    count_outIncrement_cp: coverpoint count_out iff(rst_n && ce && up_down);

    count_outOverflow_cp: coverpoint count_out iff(rst_n && ce && up_down){
        bins OV = (MAX => 0);
    }

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

    count_outUnderflow_cp: coverpoint count_out iff(rst_n && ce && !up_down){
        bins UV = (0 => MAX);
    }
endgroup
```

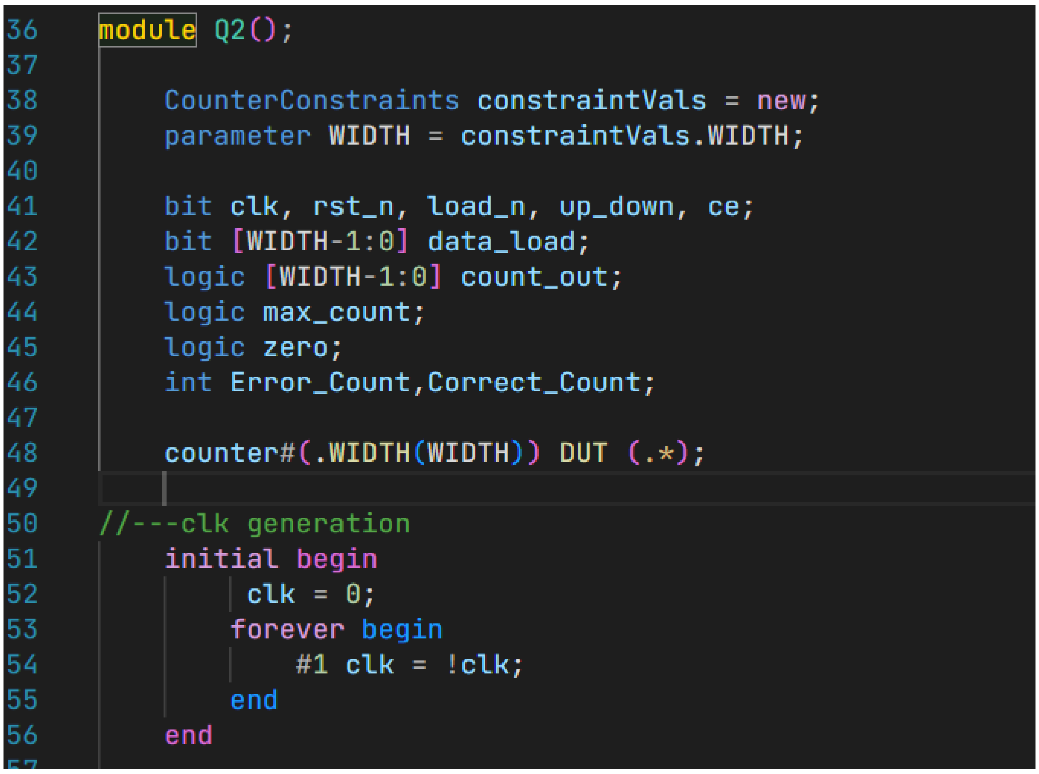


```
    constraint CounterSignals{

        rst_n dist {
            'b0 := 10,
            'b1 := 90
        };
        load_n dist {
            'b0 := 70,
            'b1 := 30
        };
        ce dist {
            'b0 := 70,
            'b1 := 30
        };
        up_down dist {
            'b0 := 50,
            'b1 := 50
        };
        data_load !=0;
    }
    endclass
endpackage
```

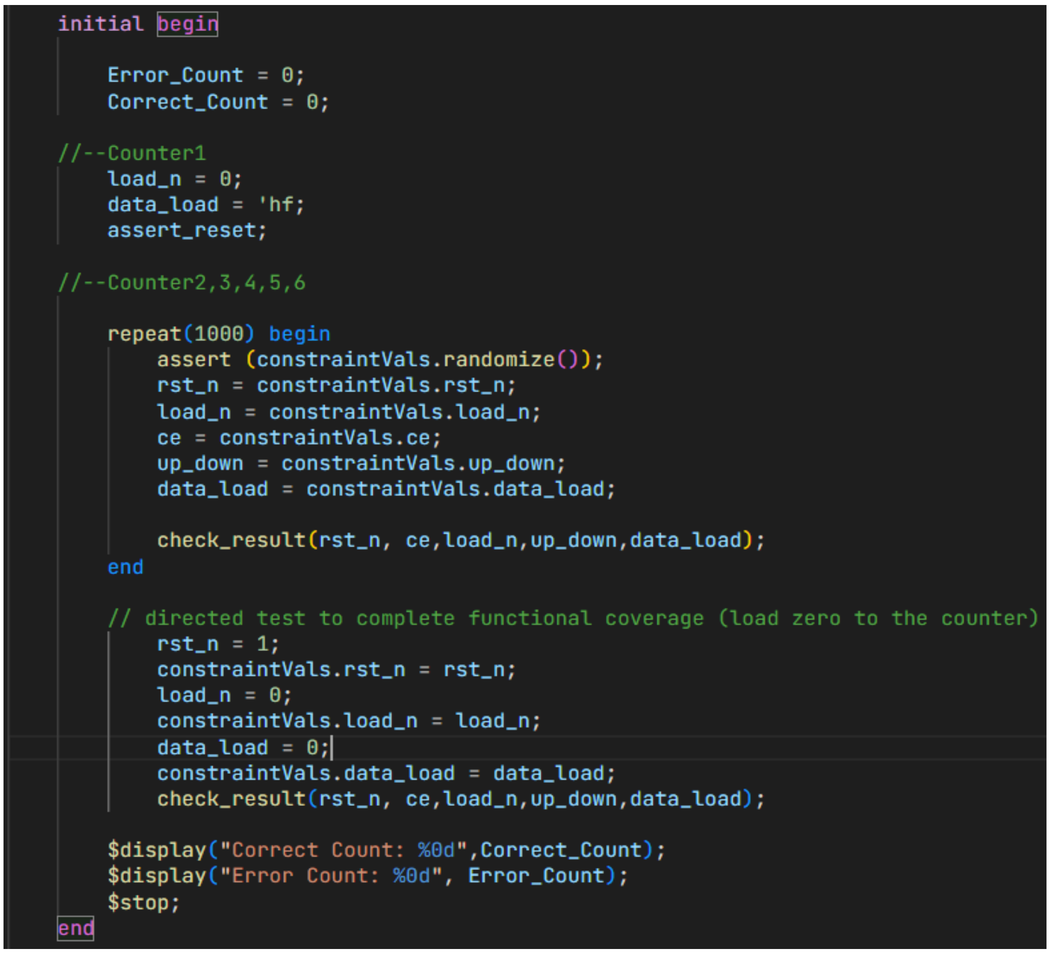


```
        function new();
            CovCode = new();
        endfunction
    endclass
```

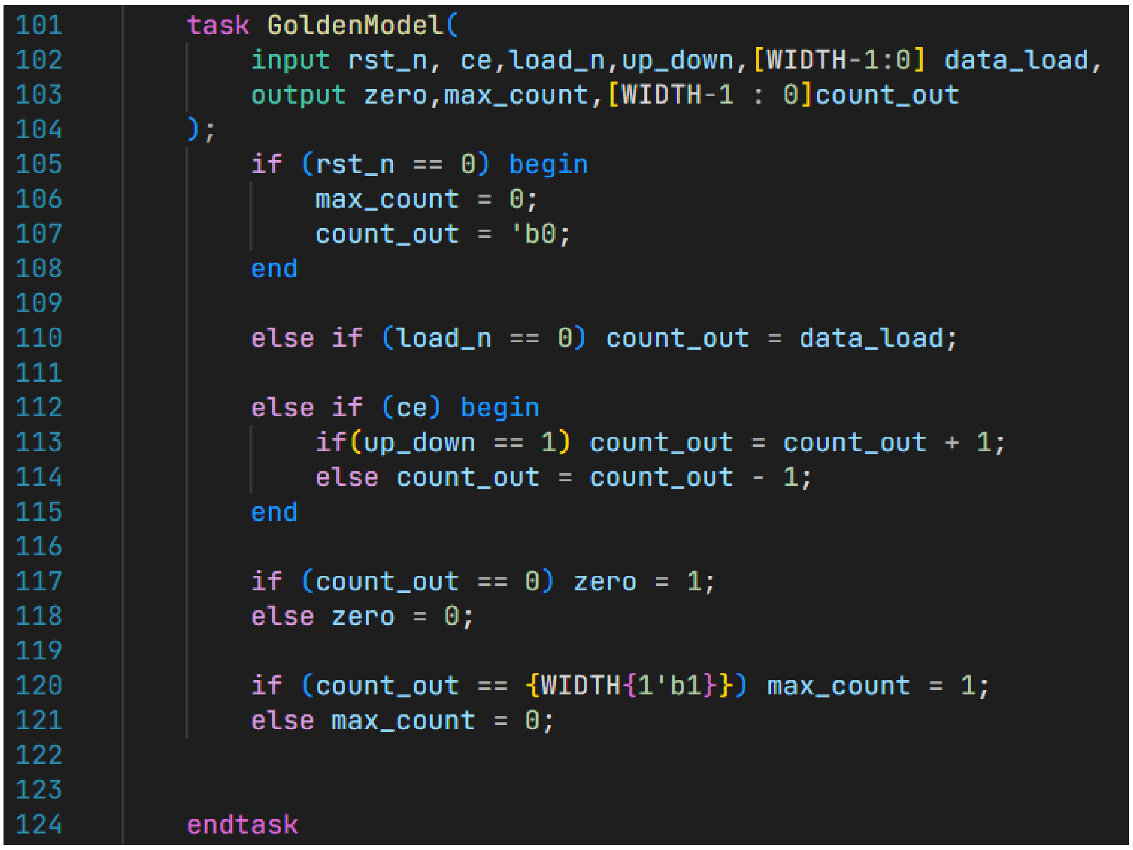
❖ Test bench:



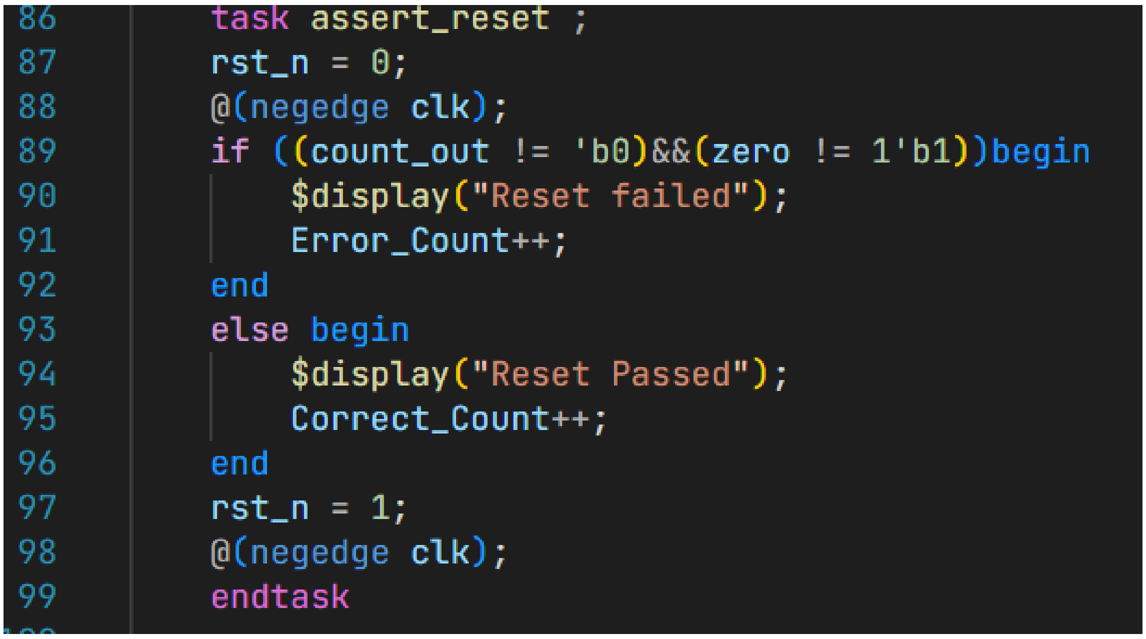
```
module Q2();

    CounterConstraints constraintVals = new;
    parameter WIDTH = constraintVals.WIDTH;

    bit clk, rst_n, load_n, up_down, ce;
    bit [WIDTH-1:0] data_load;
    logic [WIDTH-1:0] count_out;
    logic max_count;
    logic zero;
    int Error_Count,Correct_Count;

    counter#(.WIDTH(WIDTH)) DUT (.*);

//---clk generation
    initial begin
        clk = 0;
        forever begin
            #1 clk = !clk;
        end
    end
```



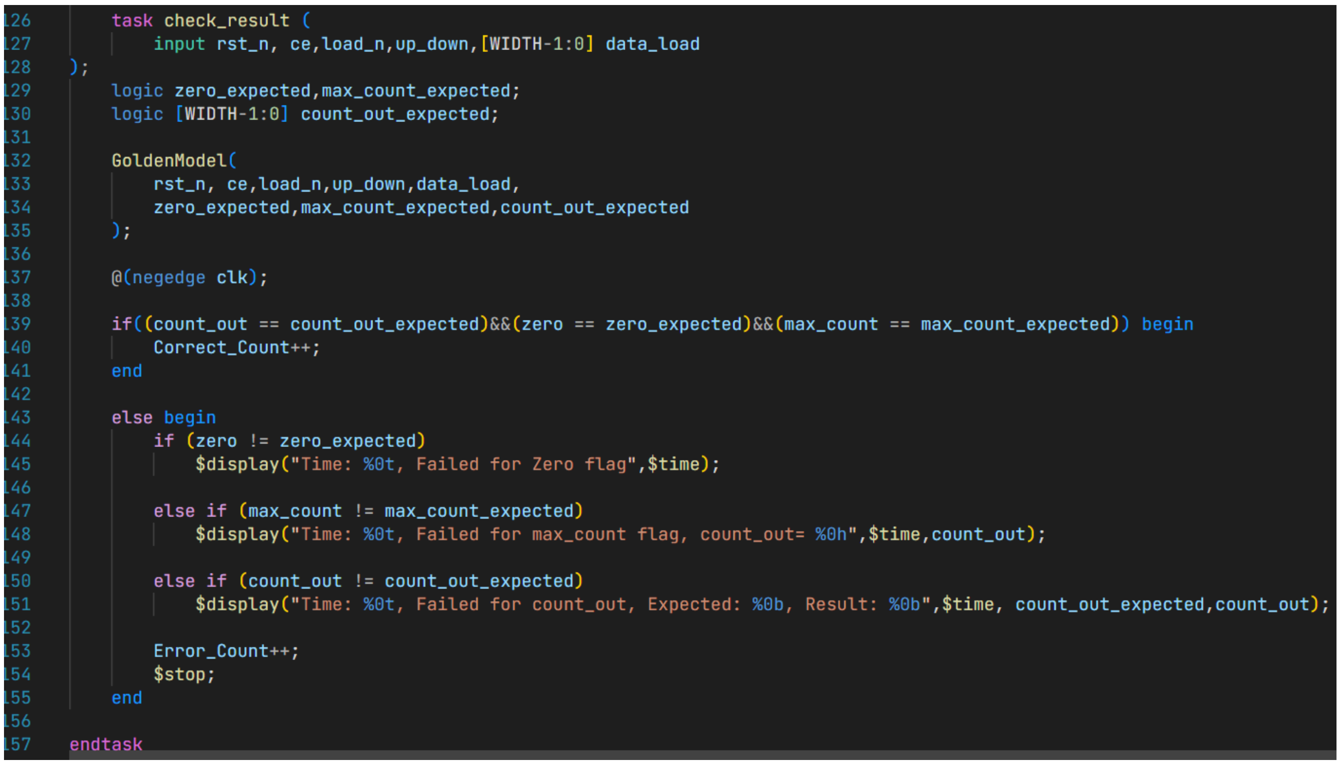
```
initial begin

    Error_Count = 0;
    Correct_Count = 0;

//--Counter1
    load_n = 0;
    data_load = 'hf;
    assert_reset;

//--Counter2,3,4,5,6

    repeat(1000) begin
        assert (constraintVals.randomize());
        rst_n = constraintVals.rst_n;
        load_n = constraintVals.load_n;
        ce = constraintVals.ce;
        up_down = constraintVals.up_down;
        data_load = constraintVals.data_load;

        check_result(rst_n, ce,load_n,up_down,data_load);
    end

    // directed test to complete functional coverage (load zero to the counter)
        rst_n = 1;
        constraintVals.rst_n = rst_n;
        load_n = 0;
        constraintVals.load_n = load_n;
        data_load = 0;
        constraintVals.data_load = data_load;
        check_result(rst_n, ce,load_n,up_down,data_load);

    $display("Correct Count: %0d",Correct_Count);
    $display("Error Count: %0d", Error_Count);
    $stop;
end
```



```
task GoldenModel(
    input rst_n, ce,load_n,up_down,[WIDTH-1:0] data_load,
    output zero,max_count,[WIDTH-1 : 0]count_out
);
    if (rst_n == 0) begin
        max_count = 0;
        count_out = 'b0;
    end

    else if (load_n == 0) count_out = data_load;

    else if (ce) begin
        if(up_down == 1) count_out = count_out + 1;
        else count_out = count_out - 1;
    end

    if (count_out == 0) zero = 1;
    else zero = 0;

    if (count_out == {WIDTH{1'b1}}) max_count = 1;
    else max_count = 0;


endtask
```

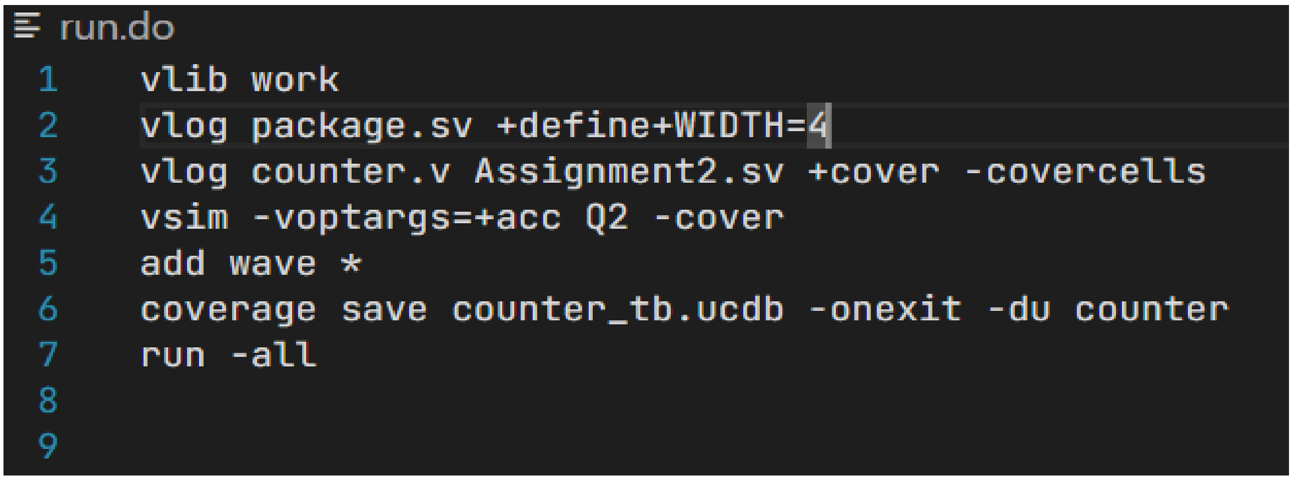


```
task assert_reset ;
rst_n = 0;
@(negedge clk);
if ((count_out != 'b0)&&(zero != 1'b1))begin
    $display("Reset failed");
    Error_Count++;
end
else begin
    $display("Reset Passed");
    Correct_Count++;
end
rst_n = 1;
@(negedge clk);
endtask
```



```
    task check_result (
        input rst_n, ce,load_n,up_down,[WIDTH-1:0] data_load
);
    logic zero_expected,max_count_expected;
    logic [WIDTH-1:0] count_out_expected;

    GoldenModel(
        rst_n, ce,load_n,up_down,data_load,
        zero_expected,max_count_expected,count_out_expected
    );

    @(negedge clk);

    if((count_out == count_out_expected)&&(zero == zero_expected)&&(max_count == max_count_expected)) begin
        Correct_Count++;
    end

    else begin
        if (zero != zero_expected)
            $display("Time: %0t, Failed for Zero flag",$time);

        else if (max_count != max_count_expected)
            $display("Time: %0t, Failed for max_count flag, count_out= %0h",$time,count_out);

        else if (count_out != count_out_expected)
            $display("Time: %0t, Failed for count_out, Expected: %0b, Result: %0b",$time, count_out_expected,count_out);

        Error_Count++;
        $stop;
    end

endtask
```

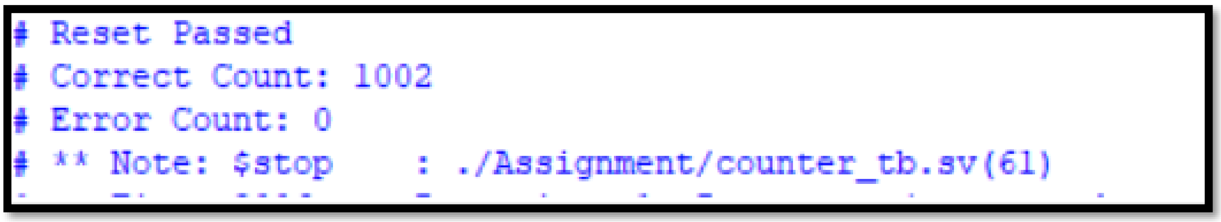
❖ **<u>Do file:</u>**



run.do

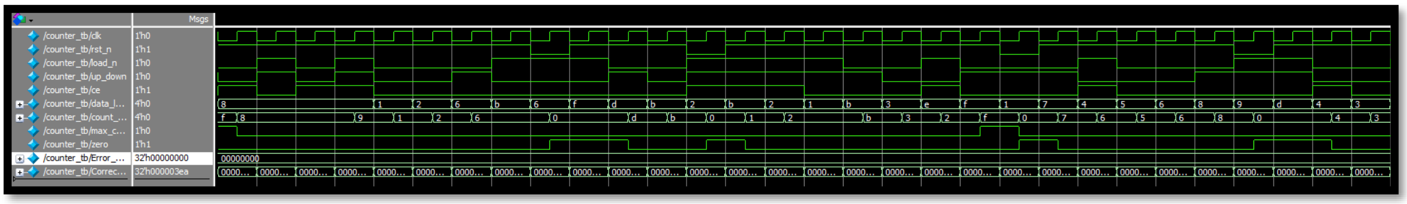
```
vlib work
vlog package.sv +define+WIDTH=4
vlog counter.v Assignment2.sv +cover -covercells
vsim -voptargs=+acc Q2 -cover
add wave *
coverage save counter_tb.ucdb -onexit -du counter
run -all
```

❖ **Result:**



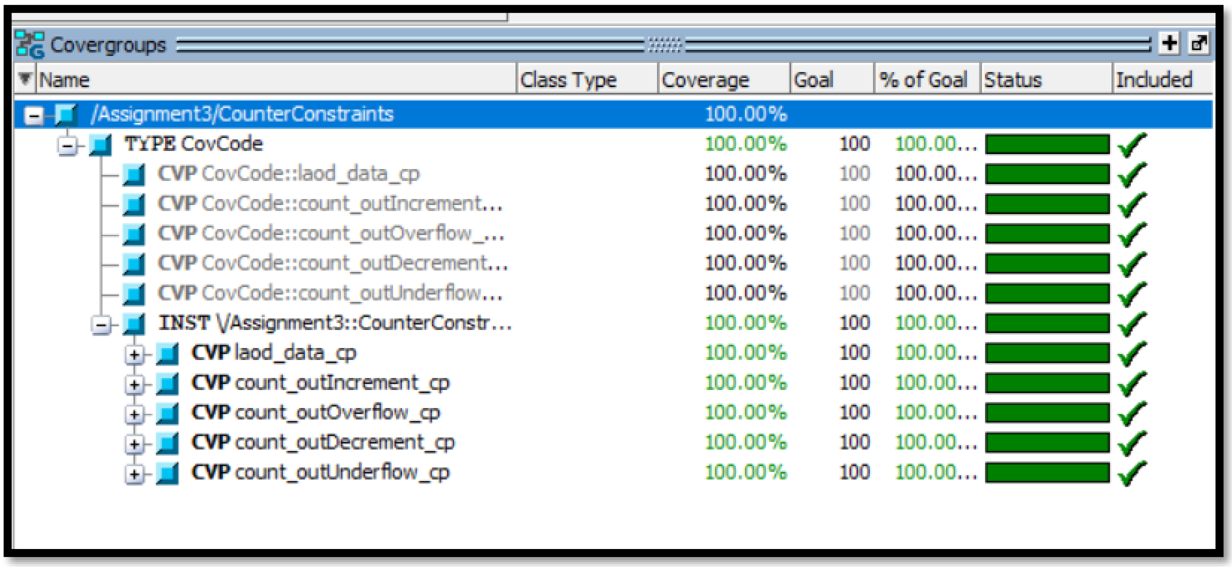
```
# Reset Passed
# Correct Count: 1002
# Error Count: 0
# ** Note: $stop    : ./Assignment/counter_tb.sv(61)
```

Waveform:



```
                              Msgs
/counter_tb/clk         1'h0
/counter_tb/rst_n       1'h1
/counter_tb/load_n      1'h0
/counter_tb/up_down     1'h0
/counter_tb/ce          1'h1
/counter_tb/data_l...   4'h0
/counter_tb/count_...   4'h0
/counter_tb/max_c...    1'h0
/counter_tb/zero        1'h1
/counter_tb/Error_...   32'h00000000
/counter_tb/Correc...   32'h000003ea
```

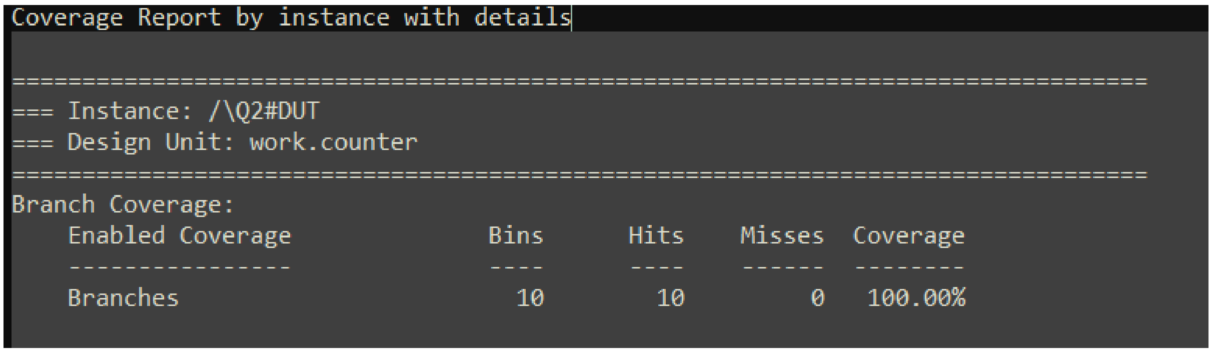
Functional Coverage:



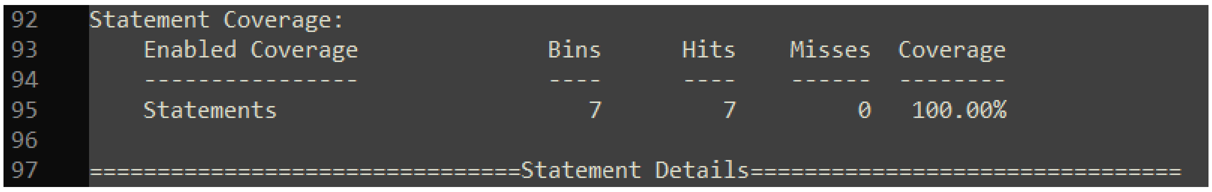
Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/Assignment3/CounterConstraints		100.00%				
TYPE CovCode		100.00%	100	100.00...	🟩	✓
CVP CovCode::laod_data_cp		100.00%	100	100.00...	🟩	✓
CVP CovCode::count_outIncrement...		100.00%	100	100.00...	🟩	✓
CVP CovCode::count_outOverflow_...		100.00%	100	100.00...	🟩	✓
CVP CovCode::count_outDecrement...		100.00%	100	100.00...	🟩	✓
CVP CovCode::count_outUnderflow...		100.00%	100	100.00...	🟩	✓
INST \/Assignment3::CounterConstr...		100.00%	100	100.00...	🟩	✓
CVP laod_data_cp		100.00%	100	100.00...	🟩	✓
CVP count_outIncrement_cp		100.00%	100	100.00...	🟩	✓
CVP count_outOverflow_cp		100.00%	100	100.00...	🟩	✓
CVP count_outDecrement_cp		100.00%	100	100.00...	🟩	✓
CVP count_outUnderflow_cp		100.00%	100	100.00...	🟩	✓

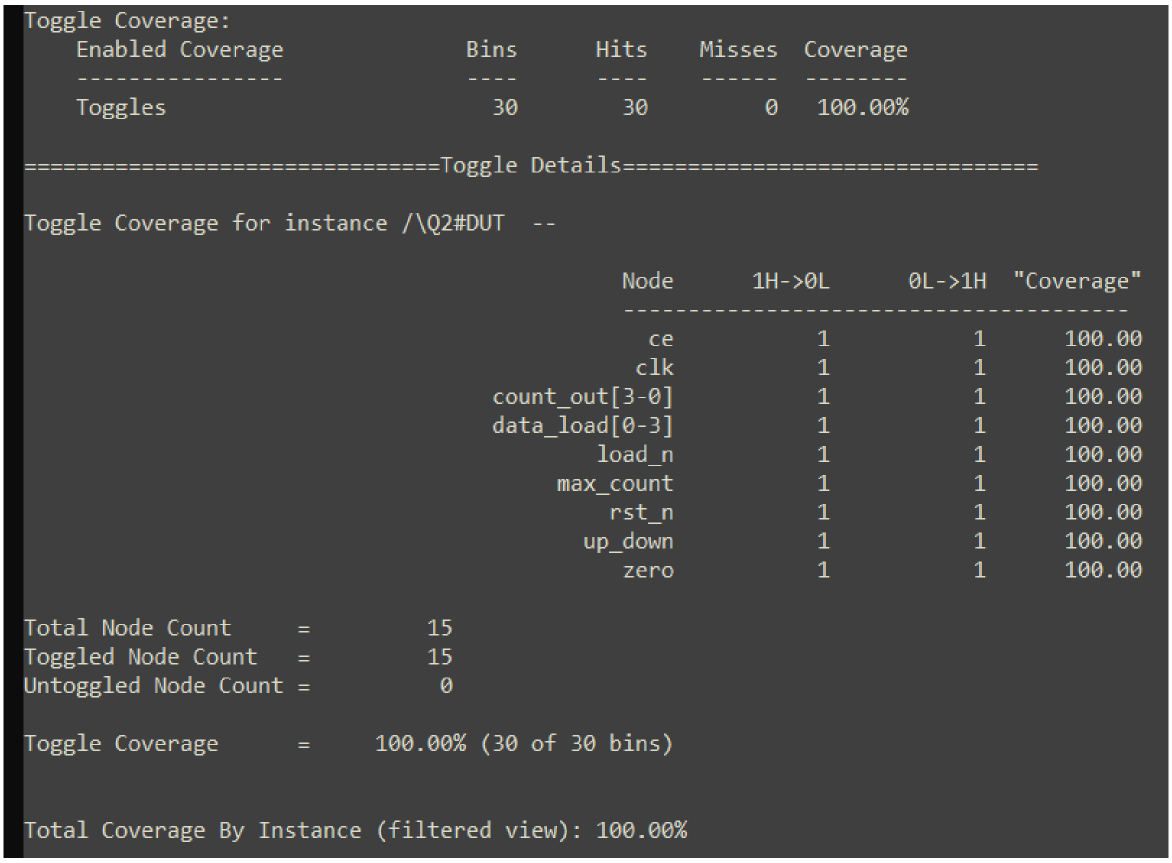
❖ Code Coverage :



```
Coverage Report by instance with details

=====================================================================================
=== Instance: /\Q2#DUT
=== Design Unit: work.counter
=====================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%
```



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

================================Statement Details================================
```



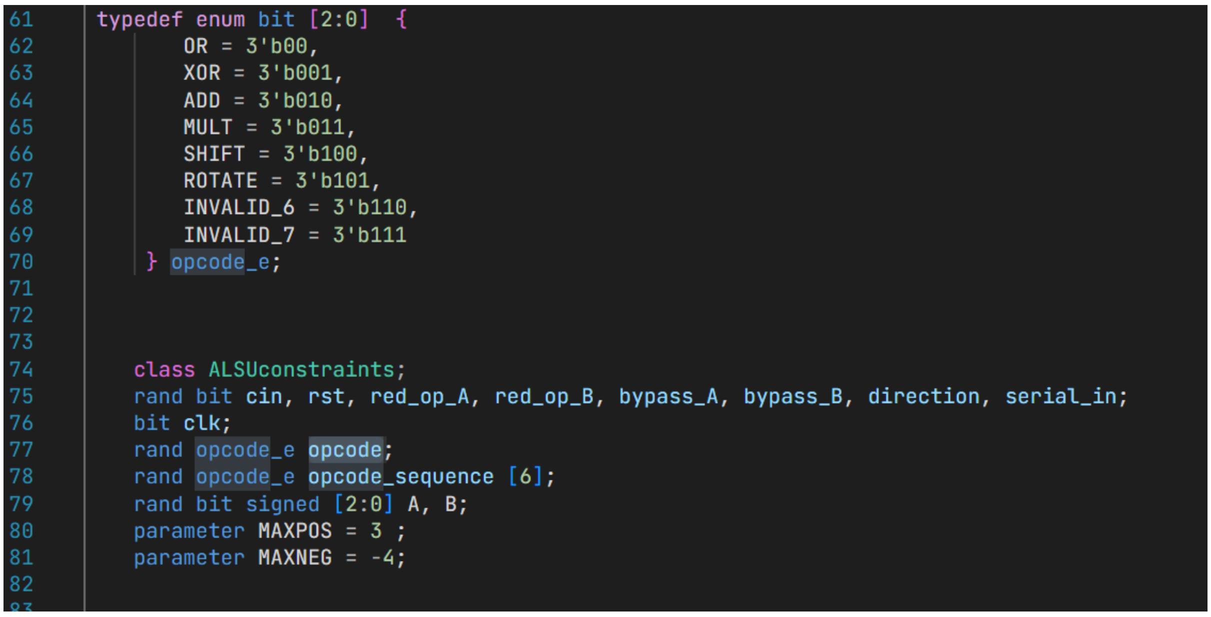
```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         30        30         0   100.00%

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

Toggle Coverage for instance /\Q2#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)


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

Q3) ALSU

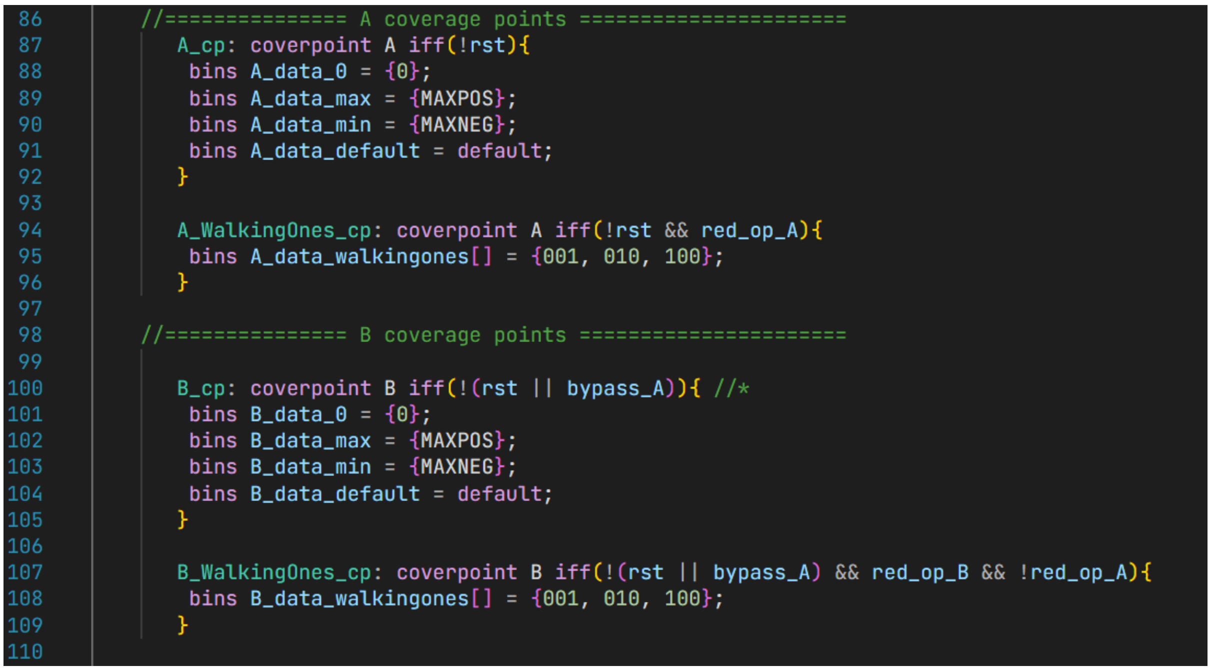
❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
ALSU1	when reset is asserted, Outputs should be Low for at least one clk cycle.	Directed at the start of the simulation	-	A checker in the testbench to make sure the output is correct
ALSU2	In case opcode = ADD/MULT, & no invalid case, **LEDs** should = 0 &**out** should = A + B or A*B	Randomized under constraint of having A and B equals to(Max., Min., Zero) most of the time,	**Bins_arith[],** Ensures the happeness of this opcode values.	Comparing results to a refrence golden model
ALSU3	If **opcode = OR/XOR** and no invalid cases, if both red_op_A/B are low, **out = A OR/XOR B**, else if red_op_A is high, **out = redOR/XOR A**, else if red_op_A is low and red_op_b is high, **out = redOR/XOR B**	Randomized under constraint of having A/B having a high bit is reduction operation	**Bins_bitwise[],** Ensures the happeness of this opcode values.	Comparing results to a refrence golden model
ALSU4	verify Shift/ Rotate operations	Random during the simulation	**Bins_shift[],** Ensures the happeness of this opcode values.	Comparing results to a refrence golden model
ALSU5	Verifying correct behaviour if invalid operation happens, **LEDs** should blink and **out** should be low, if bypass → out = the bypassed signal with higher priority for A	Randomized under constraints of having less invalid operations, and low probabity of having a bypass signal high	Bin_invalid[]	Comparing results to a refrence golden model

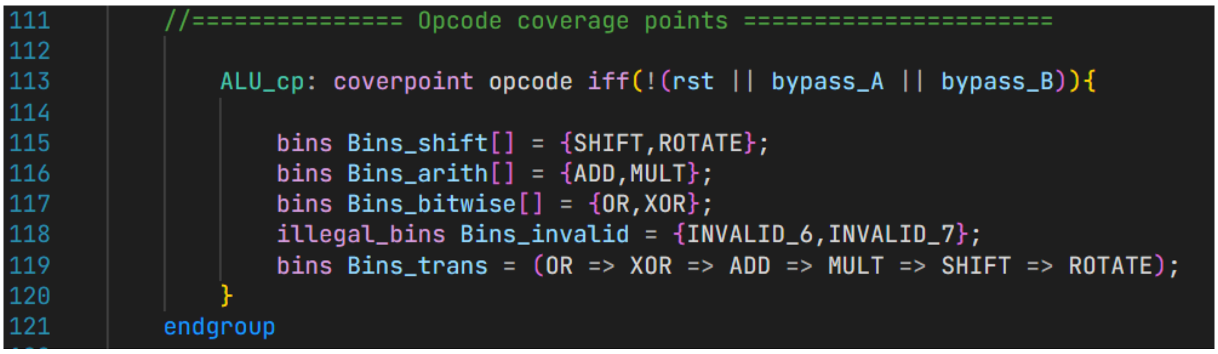
❖ Package class:



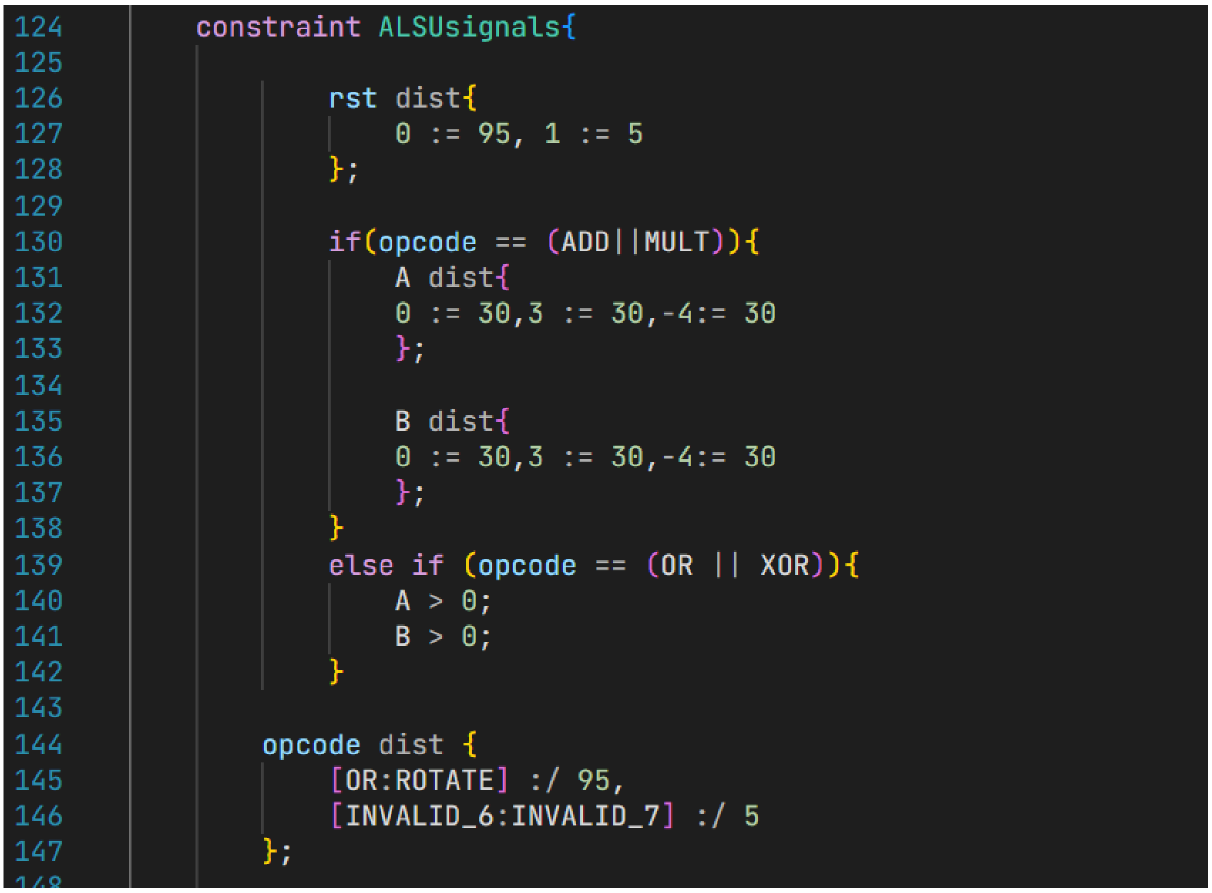
```
typedef enum bit [2:0]  {
        OR = 3'b00,
        XOR = 3'b001,
        ADD = 3'b010,
        MULT = 3'b011,
        SHIFT = 3'b100,
        ROTATE = 3'b101,
        INVALID_6 = 3'b110,
        INVALID_7 = 3'b111
    } opcode_e;



    class ALSUconstraints;
    rand bit cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    bit clk;
    rand opcode_e opcode;
    rand opcode_e opcode_sequence [6];
    rand bit signed [2:0] A, B;
    parameter MAXPOS = 3 ;
    parameter MAXNEG = -4;
```



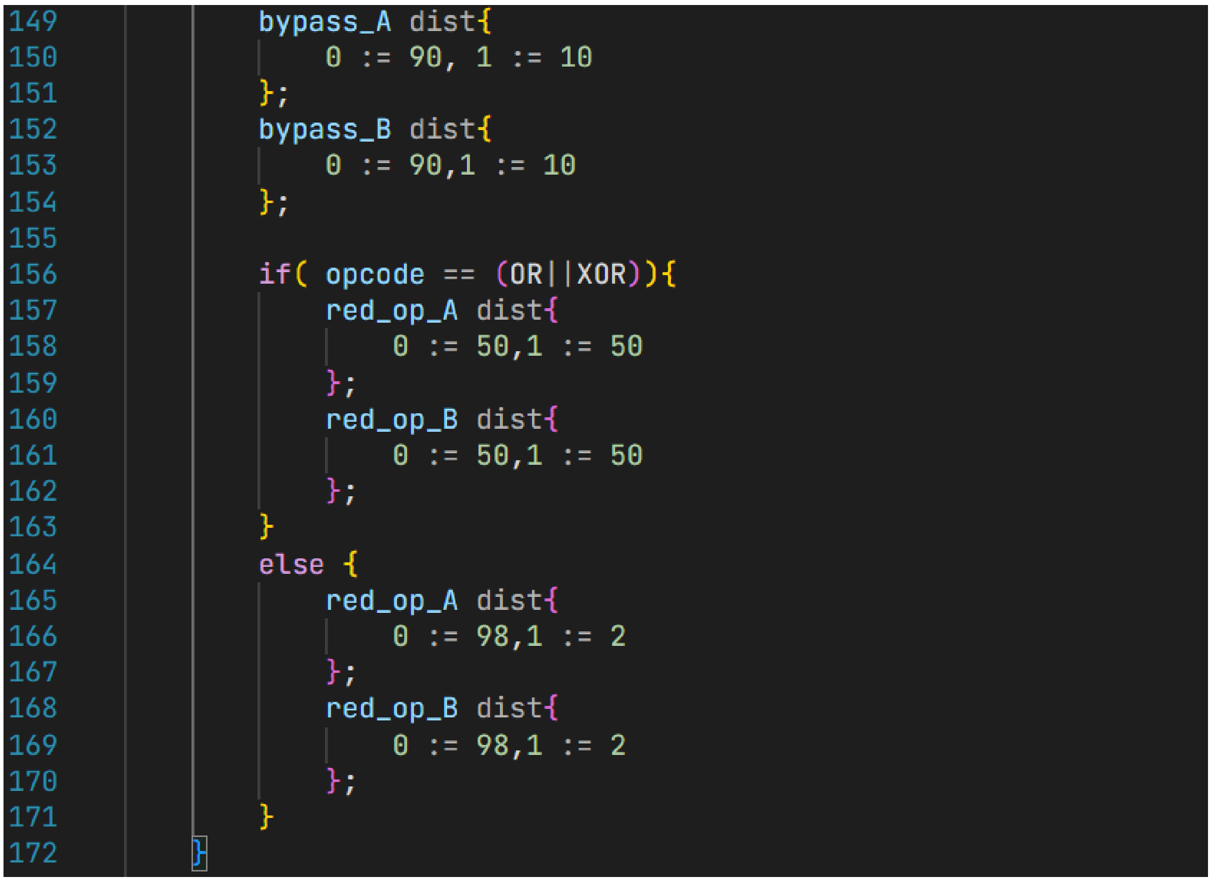
```
    //=============== A coverage points ======================
       A_cp: coverpoint A iff(!rst){
        bins A_data_0 = {0};
        bins A_data_max = {MAXPOS};
        bins A_data_min = {MAXNEG};
        bins A_data_default = default;
       }

       A_WalkingOnes_cp: coverpoint A iff(!rst && red_op_A){
        bins A_data_walkingones[] = {001, 010, 100};
       }

    //=============== B coverage points ======================

       B_cp: coverpoint B iff(!(rst || bypass_A)){ //*
        bins B_data_0 = {0};
        bins B_data_max = {MAXPOS};
        bins B_data_min = {MAXNEG};
        bins B_data_default = default;
       }

       B_WalkingOnes_cp: coverpoint B iff(!(rst || bypass_A) && red_op_B && !red_op_A){
        bins B_data_walkingones[] = {001, 010, 100};
       }
```



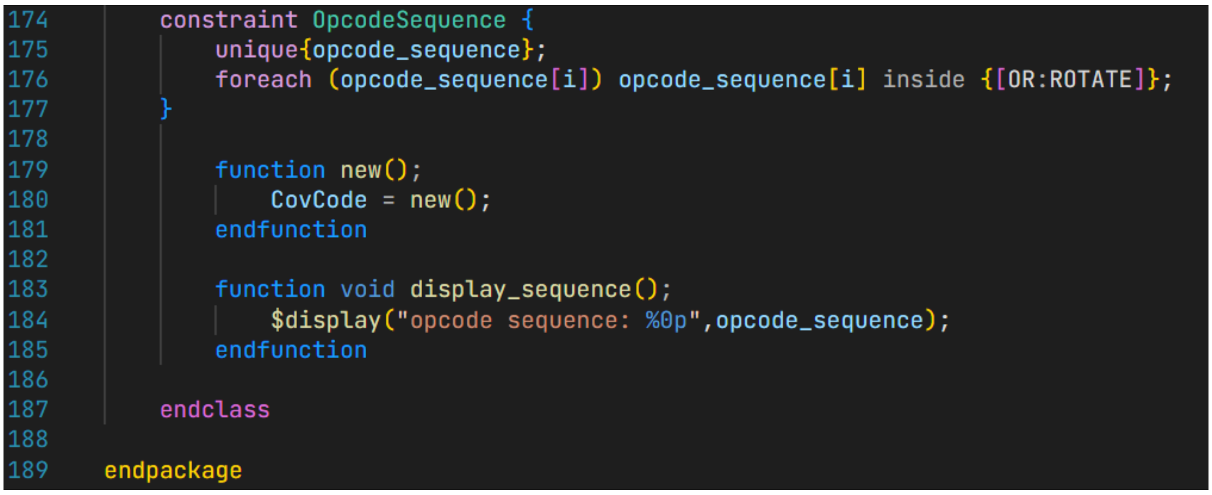
```
    //=============== Opcode coverage points ======================

        ALU_cp: coverpoint opcode iff(!(rst || bypass_A || bypass_B)){

            bins Bins_shift[] = {SHIFT,ROTATE};
            bins Bins_arith[] = {ADD,MULT};
            bins Bins_bitwise[] = {OR,XOR};
            illegal_bins Bins_invalid = {INVALID_6,INVALID_7};
            bins Bins_trans = (OR => XOR => ADD => MULT => SHIFT => ROTATE);
        }
    endgroup
```



```
    constraint ALSUsignals{

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

            if(opcode == (ADD||MULT)){
                A dist{
                0 := 30,3 := 30,-4:= 30
                };

                B dist{
                0 := 30,3 := 30,-4:= 30
                };
            }
            else if (opcode == (OR || XOR)){
                A > 0;
                B > 0;
            }

        opcode dist {
            [OR:ROTATE] :/ 95,
            [INVALID_6:INVALID_7] :/ 5
        };
```

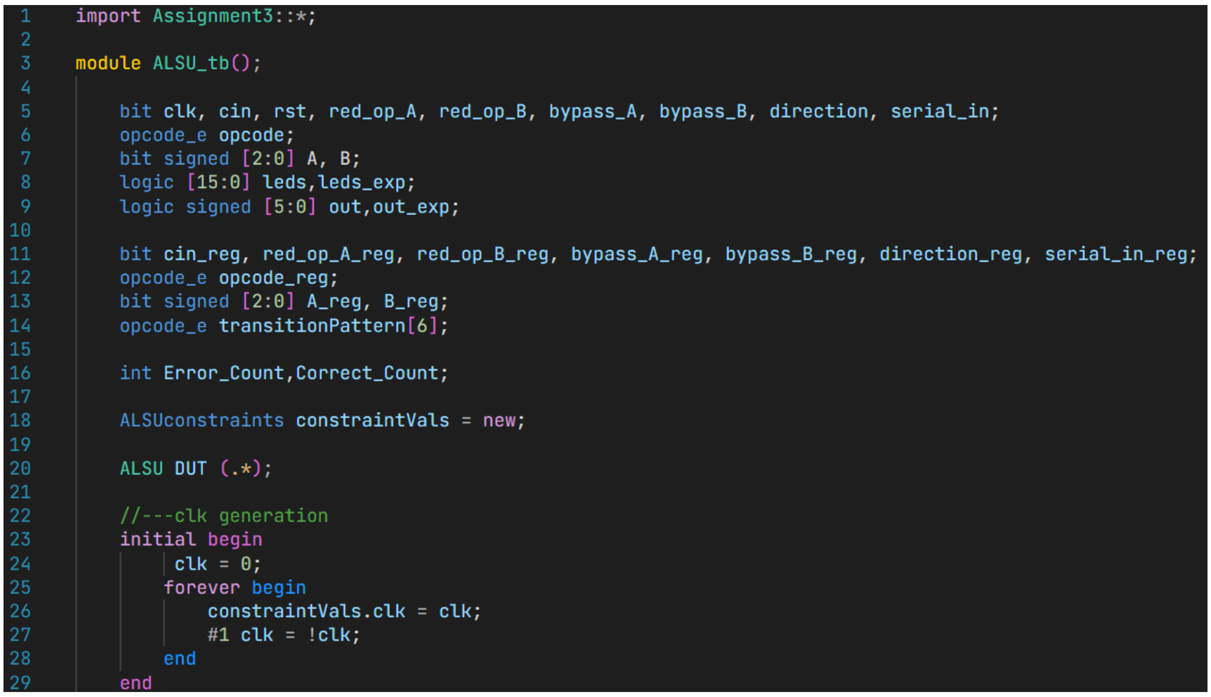


```
        bypass_A dist{
            0 := 90, 1 := 10
        };
        bypass_B dist{
            0 := 90,1 := 10
        };

        if( opcode == (OR||XOR)){
            red_op_A dist{
                0 := 50,1 := 50
            };
            red_op_B dist{
                0 := 50,1 := 50
            };
        }
        else {
            red_op_A dist{
                0 := 98,1 := 2
            };
            red_op_B dist{
                0 := 98,1 := 2
            };
        }
    }
```

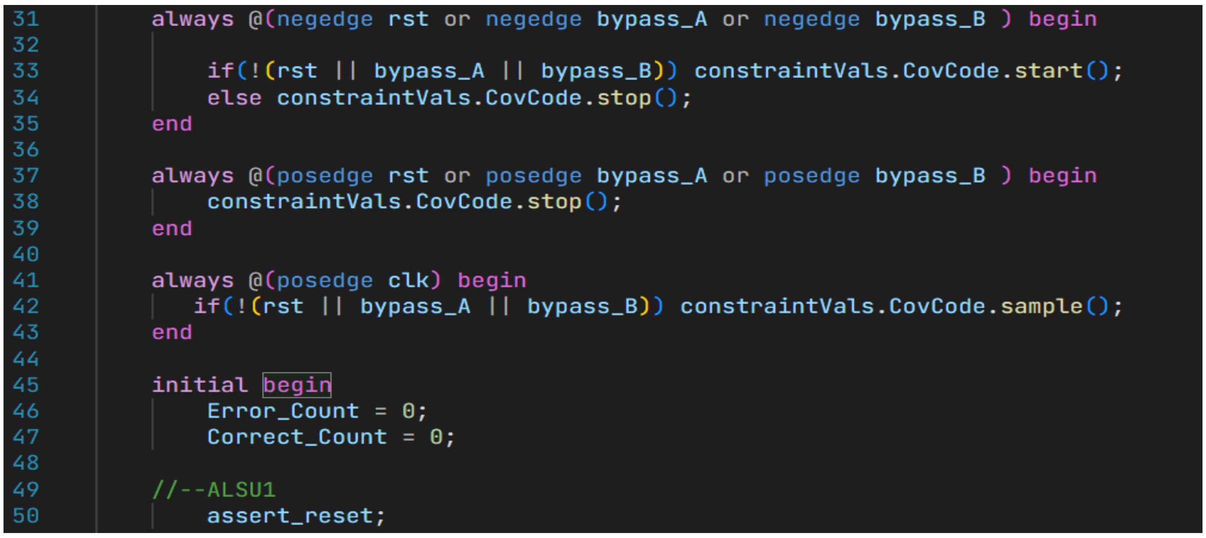


```systemverilog
    constraint OpcodeSequence {
        unique{opcode_sequence};
        foreach (opcode_sequence[i]) opcode_sequence[i] inside {[OR:ROTATE]};
    }

        function new();
            CovCode = new();
        endfunction

        function void display_sequence();
            $display("opcode sequence: %0p",opcode_sequence);
        endfunction

    endclass

endpackage
```

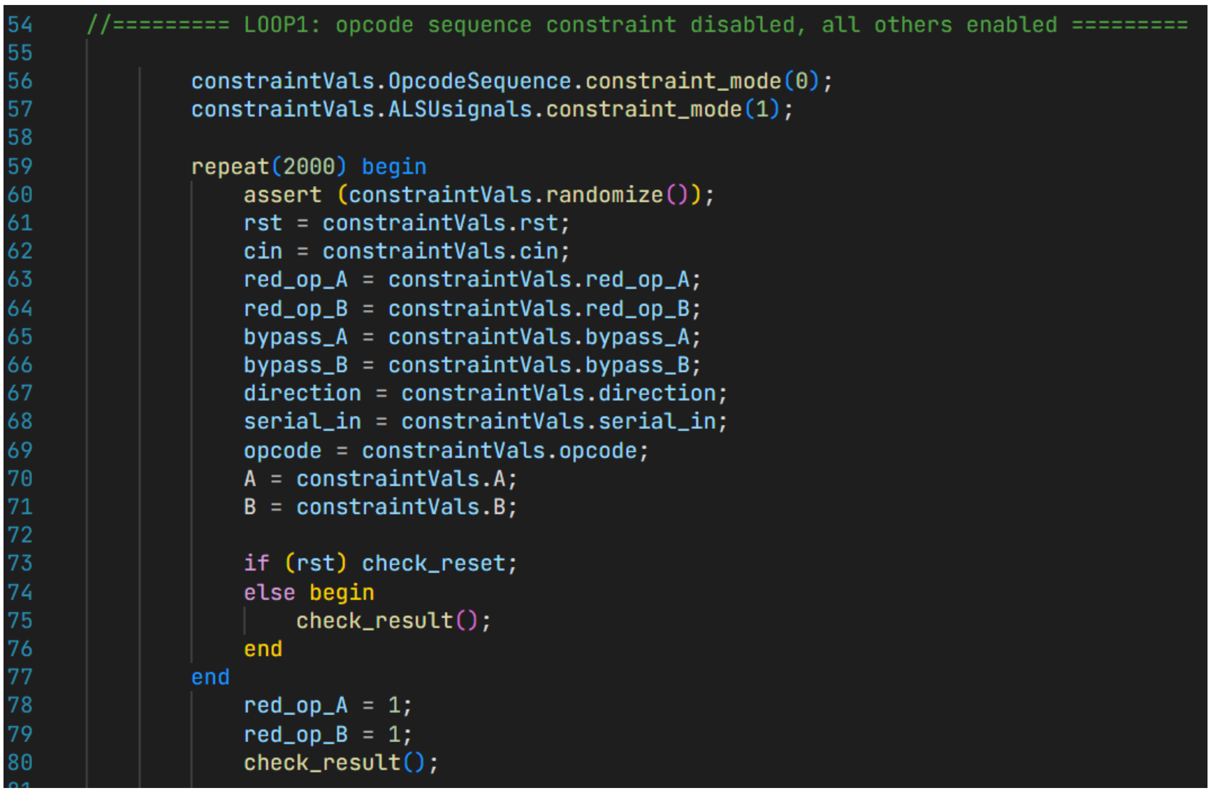
❖ **TestBench:**



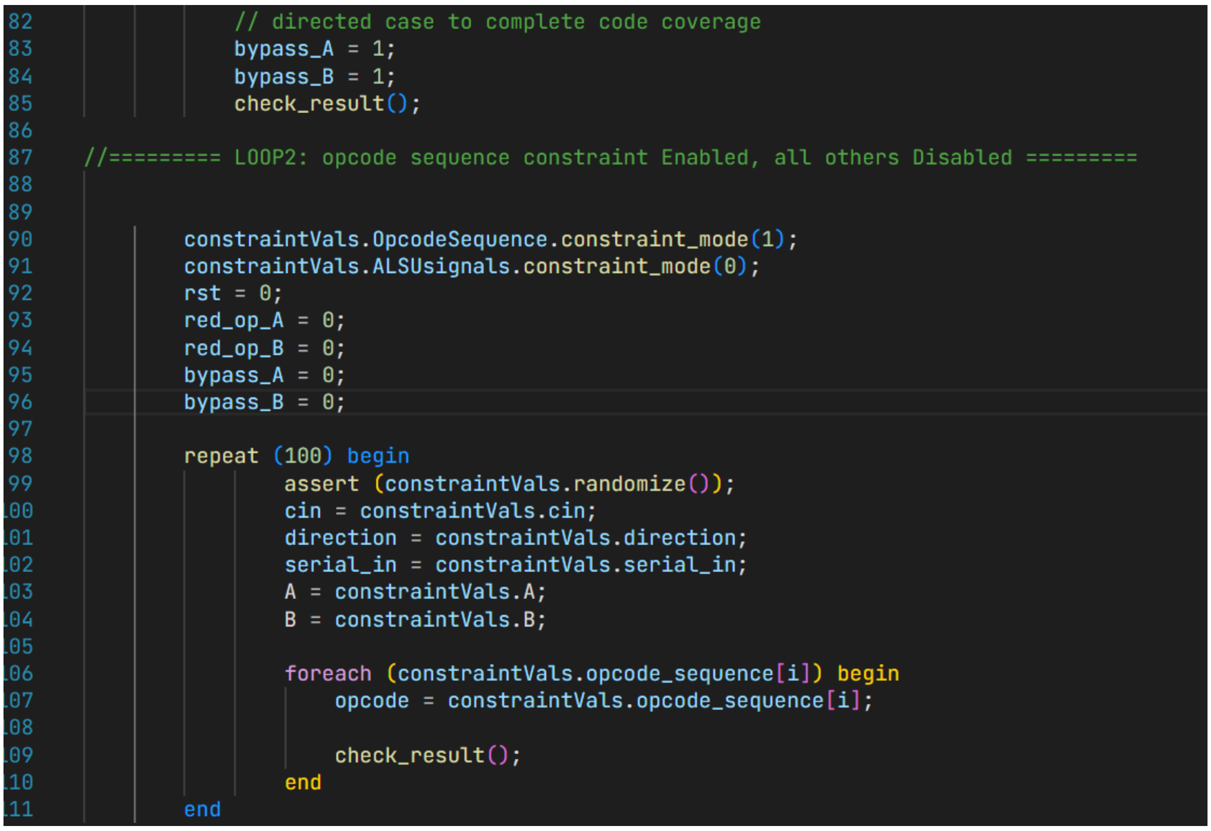
```systemverilog
import Assignment3::*;

module ALSU_tb();

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

    bit cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
    opcode_e opcode_reg;
    bit signed [2:0] A_reg, B_reg;
    opcode_e transitionPattern[6];

    int Error_Count,Correct_Count;

    ALSUconstraints constraintVals = new;

    ALSU DUT (.*);

    //---clk generation
    initial begin
        clk = 0;
        forever begin
            constraintVals.clk = clk;
            #1 clk = !clk;
        end
    end
```



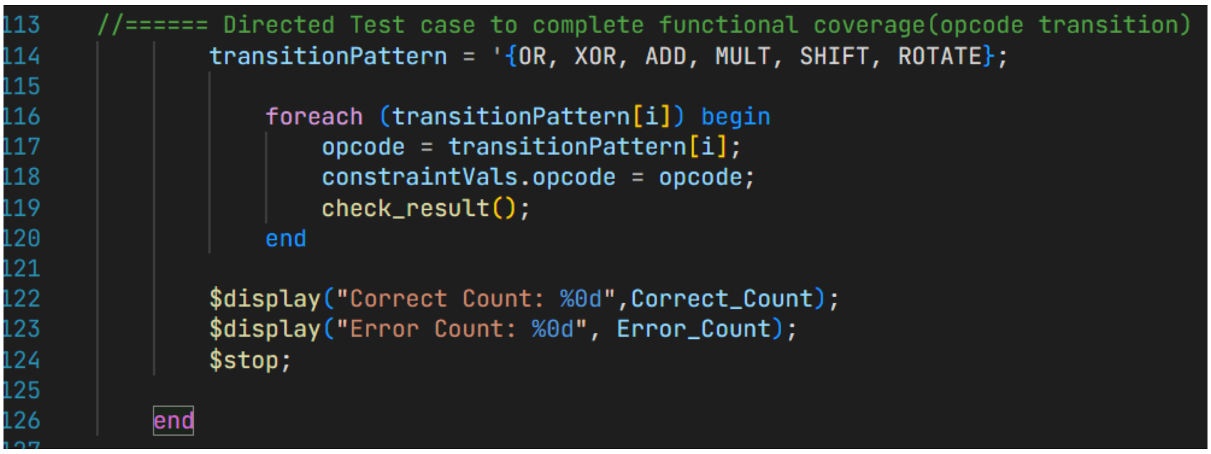
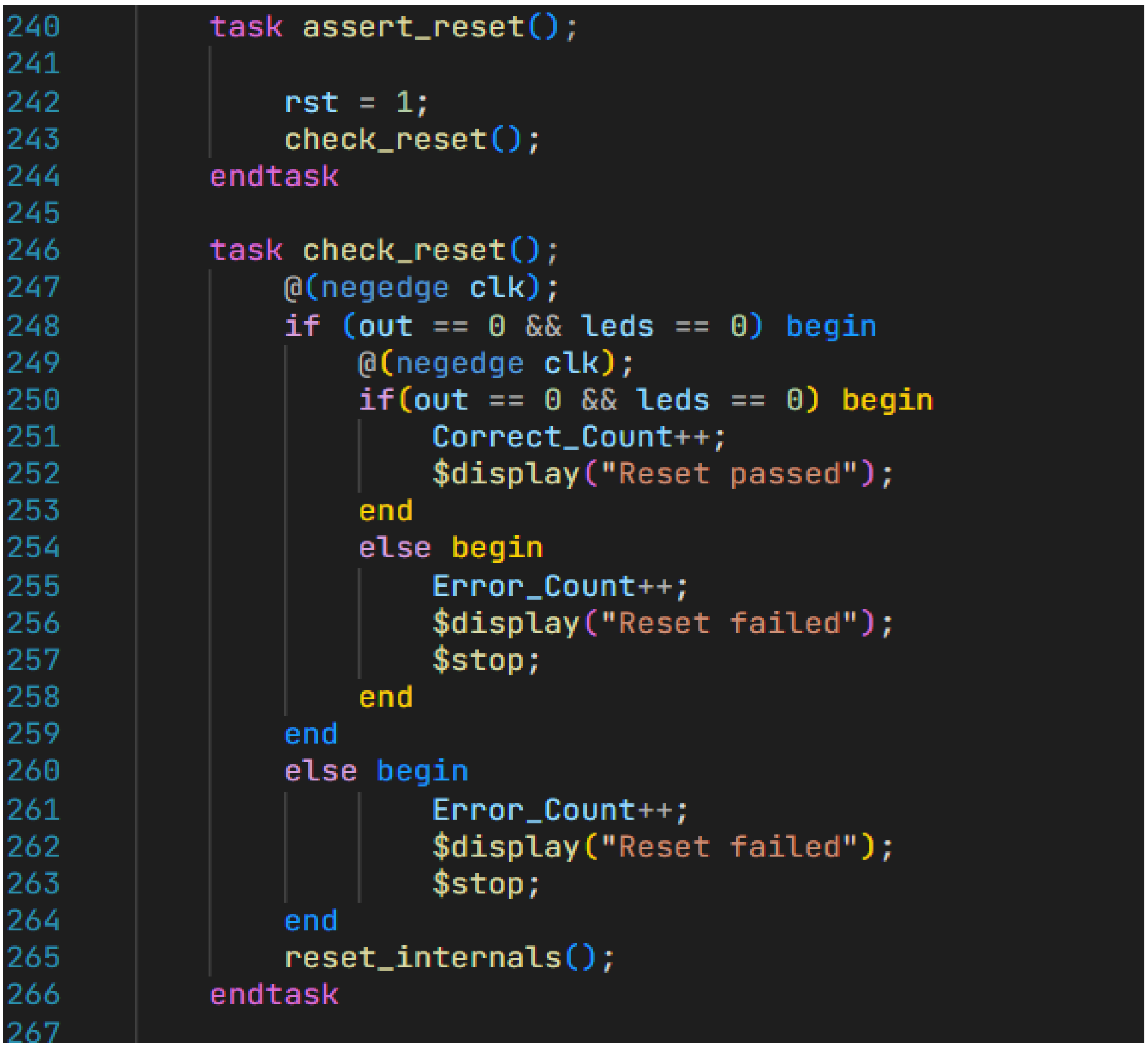
```systemverilog
    always @(negedge rst or negedge bypass_A or negedge bypass_B ) begin

        if(!(rst || bypass_A || bypass_B)) constraintVals.CovCode.start();
        else constraintVals.CovCode.stop();
    end

    always @(posedge rst or posedge bypass_A or posedge bypass_B ) begin
        constraintVals.CovCode.stop();
    end

    always @(posedge clk) begin
        if(!(rst || bypass_A || bypass_B)) constraintVals.CovCode.sample();
    end

    initial begin
        Error_Count = 0;
        Correct_Count = 0;

    //--ALSU1
        assert_reset;
```

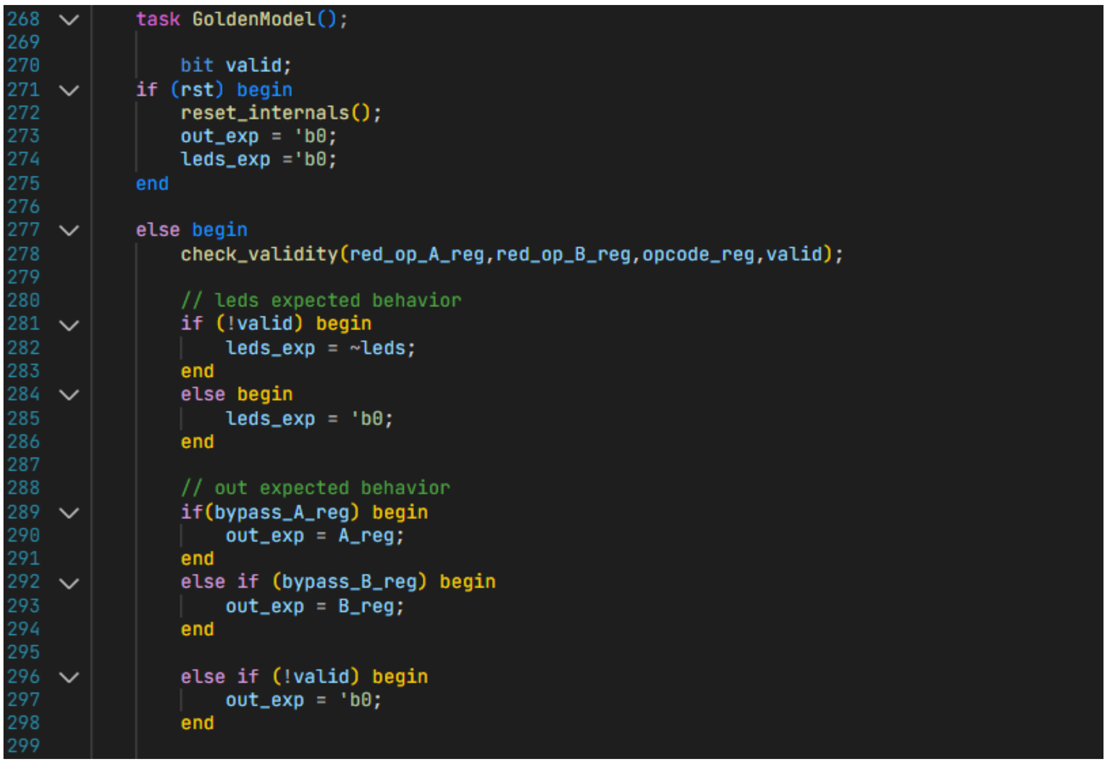


```
//========= LOOP1: opcode sequence constraint disabled, all others enabled =========

        constraintVals.OpcodeSequence.constraint_mode(0);
        constraintVals.ALSUsignals.constraint_mode(1);

        repeat(2000) begin
            assert (constraintVals.randomize());
            rst = constraintVals.rst;
            cin = constraintVals.cin;
            red_op_A = constraintVals.red_op_A;
            red_op_B = constraintVals.red_op_B;
            bypass_A = constraintVals.bypass_A;
            bypass_B = constraintVals.bypass_B;
            direction = constraintVals.direction;
            serial_in = constraintVals.serial_in;
            opcode = constraintVals.opcode;
            A = constraintVals.A;
            B = constraintVals.B;

            if (rst) check_reset;
            else begin
                check_result();
            end
        end
            red_op_A = 1;
            red_op_B = 1;
            check_result();
```

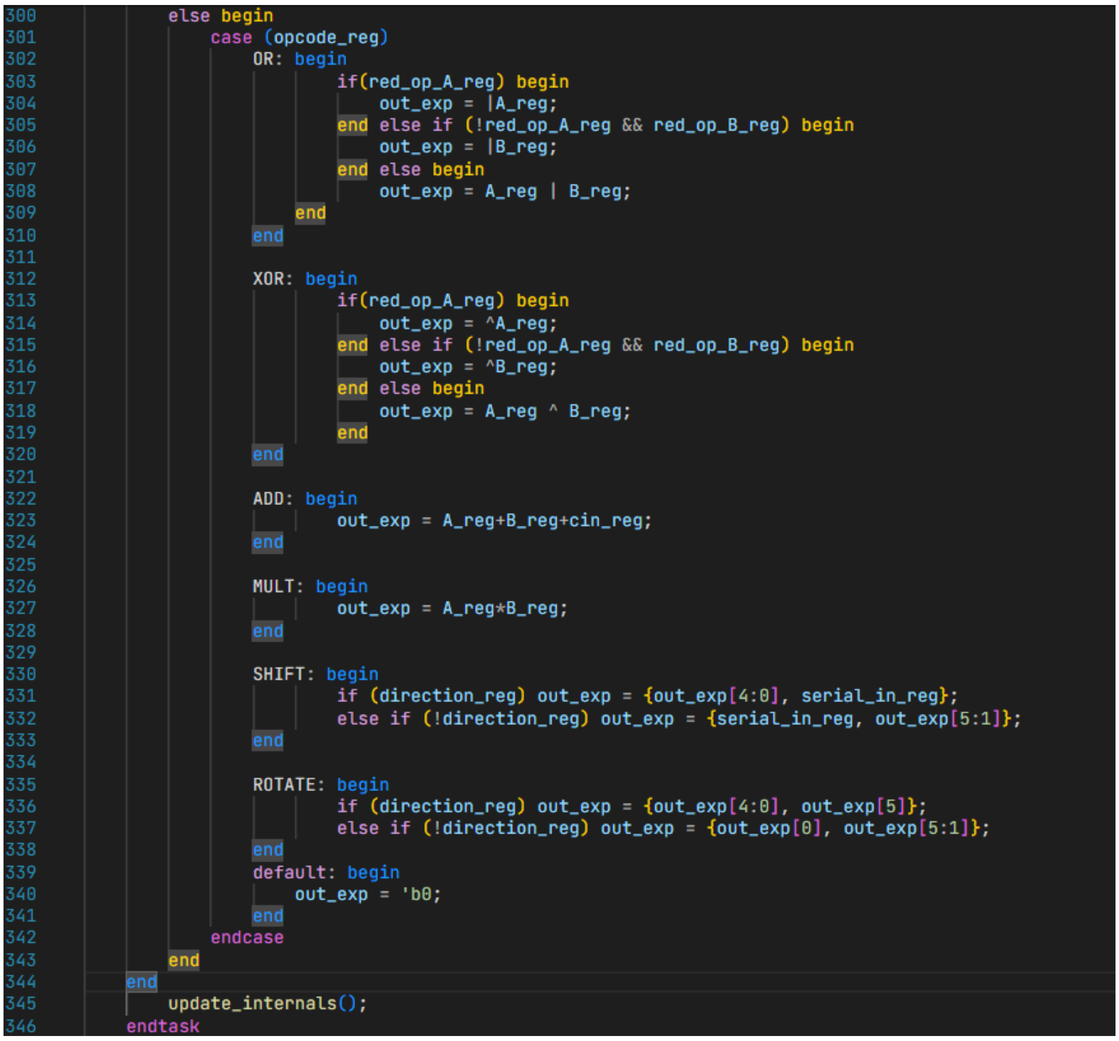


```
            // directed case to complete code coverage
            bypass_A = 1;
            bypass_B = 1;
            check_result();

//========= LOOP2: opcode sequence constraint Enabled, all others Disabled =========


        constraintVals.OpcodeSequence.constraint_mode(1);
        constraintVals.ALSUsignals.constraint_mode(0);
        rst = 0;
        red_op_A = 0;
        red_op_B = 0;
        bypass_A = 0;
        bypass_B = 0;

        repeat (100) begin
                assert (constraintVals.randomize());
                cin = constraintVals.cin;
                direction = constraintVals.direction;
                serial_in = constraintVals.serial_in;
                A = constraintVals.A;
                B = constraintVals.B;

                foreach (constraintVals.opcode_sequence[i]) begin
                    opcode = constraintVals.opcode_sequence[i];

                    check_result();
                end
        end
```

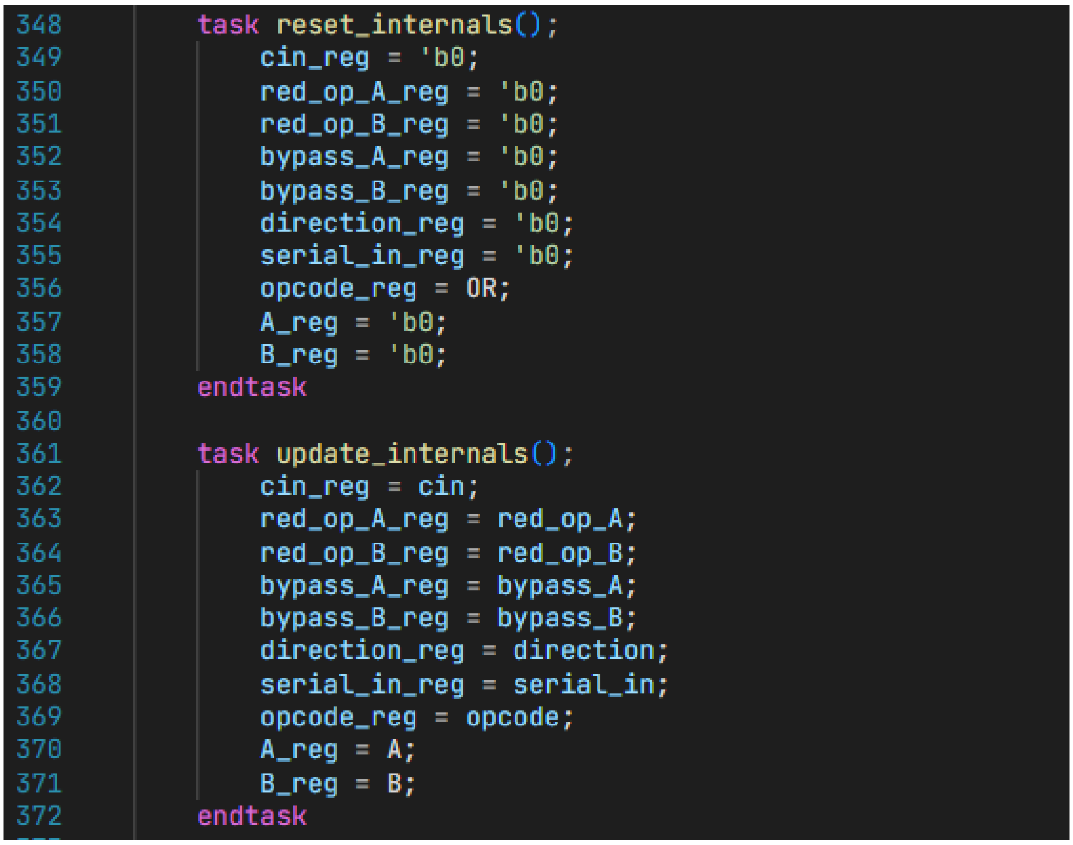

```
    //====== Directed Test case to complete functional coverage(opcode transition)
            transitionPattern = '{OR, XOR, ADD, MULT, SHIFT, ROTATE};

                foreach (transitionPattern[i]) begin
                    opcode = transitionPattern[i];
                    constraintVals.opcode = opcode;
                    check_result();
                end

            $display("Correct Count: %0d",Correct_Count);
            $display("Error Count: %0d", Error_Count);
            $stop;

        end
```

```
    task assert_reset();

        rst = 1;
        check_reset();
    endtask

    task check_reset();
        @(negedge clk);
        if (out == 0 && leds == 0) begin
            @(negedge clk);
            if(out == 0 && leds == 0) begin
                Correct_Count++;
                $display("Reset passed");
            end
            else begin
                Error_Count++;
                $display("Reset failed");
                $stop;
            end
        end
        else begin
                Error_Count++;
                $display("Reset failed");
                $stop;
        end
        reset_internals();
    endtask
```



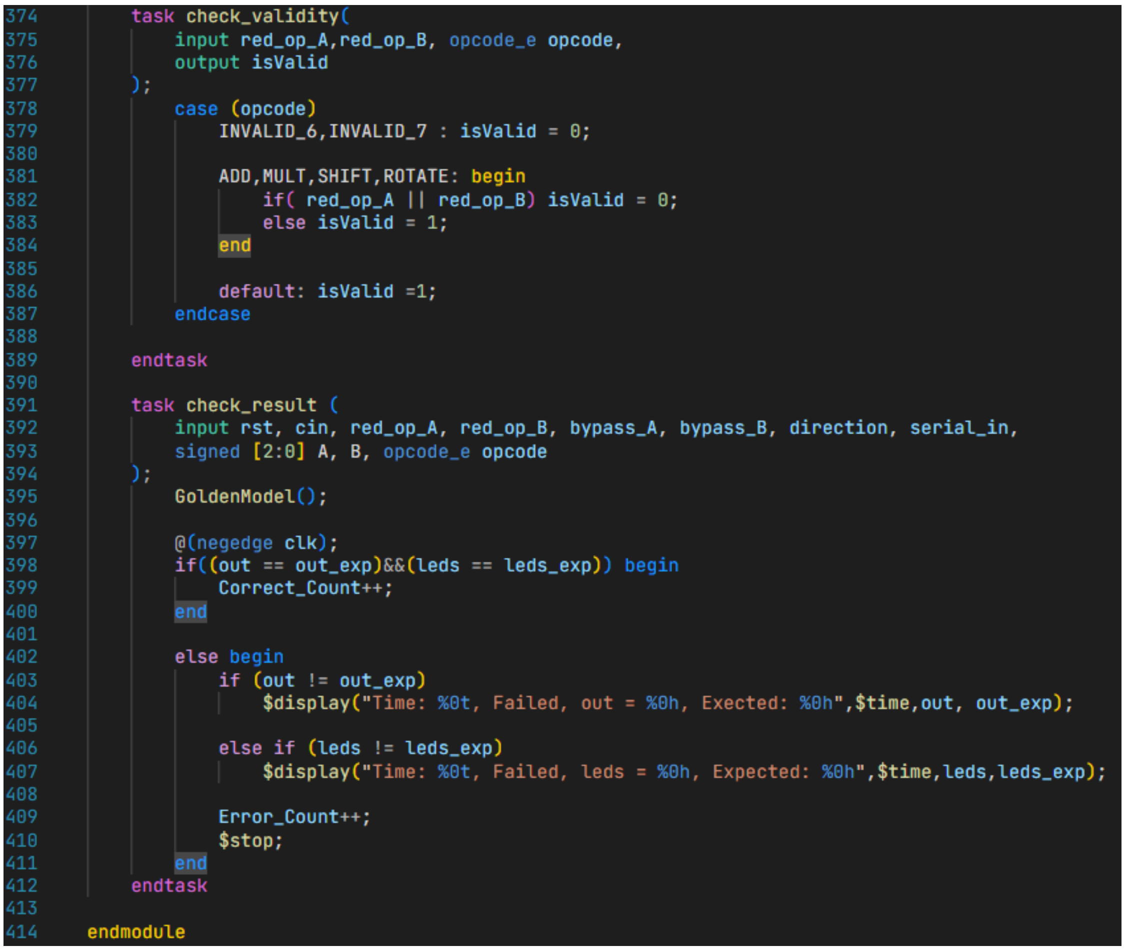
```
task GoldenModel();

    bit valid;
if (rst) begin
    reset_internals();
    out_exp = 'b0;
    leds_exp ='b0;
end

else begin
    check_validity(red_op_A_reg,red_op_B_reg,opcode_reg,valid);

    // leds expected behavior
    if (!valid) begin
        leds_exp = ~leds;
    end
    else begin
        leds_exp = 'b0;
    end

    // out expected behavior
    if(bypass_A_reg) begin
        out_exp = A_reg;
    end
    else if (bypass_B_reg) begin
        out_exp = B_reg;
    end

    else if (!valid) begin
        out_exp = 'b0;
    end

```



```
    else begin
        case (opcode_reg)
            OR: begin
                    if(red_op_A_reg) begin
                        out_exp = |A_reg;
                    end else if (!red_op_A_reg && red_op_B_reg) begin
                        out_exp = |B_reg;
                    end else begin
                        out_exp = A_reg | B_reg;
                end
            end

            XOR: begin
                    if(red_op_A_reg) begin
                        out_exp = ^A_reg;
                    end else if (!red_op_A_reg && red_op_B_reg) begin
                        out_exp = ^B_reg;
                    end else begin
                        out_exp = A_reg ^ B_reg;
                    end
            end

            ADD: begin
                    out_exp = A_reg+B_reg+cin_reg;
            end

            MULT: begin
                    out_exp = A_reg*B_reg;
            end

            SHIFT: begin
                    if (direction_reg) out_exp = {out_exp[4:0], serial_in_reg};
                    else if (!direction_reg) out_exp = {serial_in_reg, out_exp[5:1]};
            end

            ROTATE: begin
                    if (direction_reg) out_exp = {out_exp[4:0], out_exp[5]};
                    else if (!direction_reg) out_exp = {out_exp[0], out_exp[5:1]};
            end
            default: begin
                out_exp = 'b0;
            end
        endcase
    end
end
    update_internals();
endtask
```

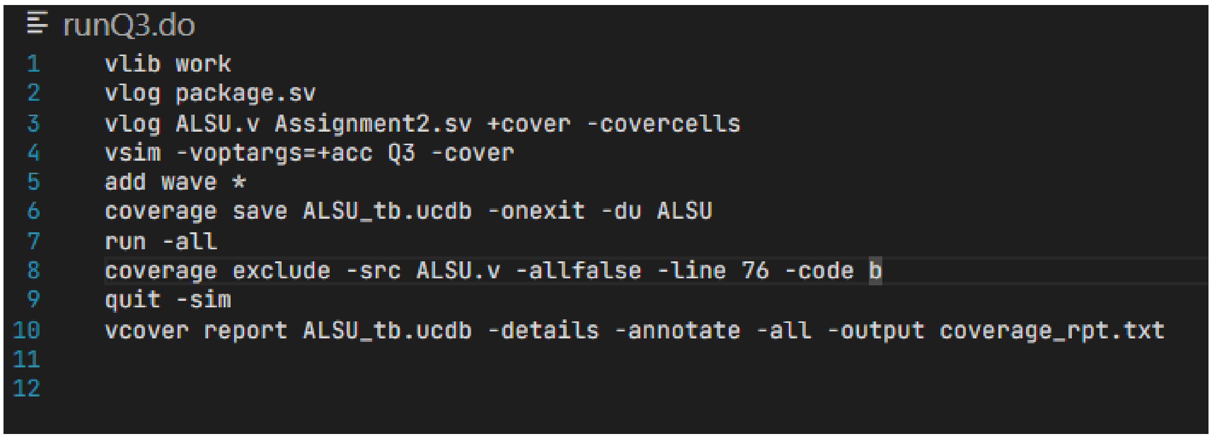


```
    task reset_internals();
        cin_reg = 'b0;
        red_op_A_reg = 'b0;
        red_op_B_reg = 'b0;
        bypass_A_reg = 'b0;
        bypass_B_reg = 'b0;
        direction_reg = 'b0;
        serial_in_reg = 'b0;
        opcode_reg = OR;
        A_reg = 'b0;
        B_reg = 'b0;
    endtask

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



```
    task check_validity(
        input red_op_A,red_op_B, opcode_e opcode,
        output isValid
    );
        case (opcode)
            INVALID_6,INVALID_7 : isValid = 0;

            ADD,MULT,SHIFT,ROTATE: begin
                if( red_op_A || red_op_B) isValid = 0;
                else isValid = 1;
            end

            default: isValid =1;
        endcase

    endtask

    task check_result (
        input rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in,
        signed [2:0] A, B, opcode_e opcode
    );
        GoldenModel();

        @(negedge clk);
        if((out == out_exp)&&(leds == leds_exp)) begin
            Correct_Count++;
        end

        else begin
            if (out != out_exp)
                $display("Time: %0t, Failed, out = %0h, Exected: %0h",$time,out, out_exp);

            else if (leds != leds_exp)
                $display("Time: %0t, Failed, leds = %0h, Expected: %0h",$time,leds,leds_exp);

            Error_Count++;
            $stop;
        end
    endtask

endmodule
```

❖ Do file:

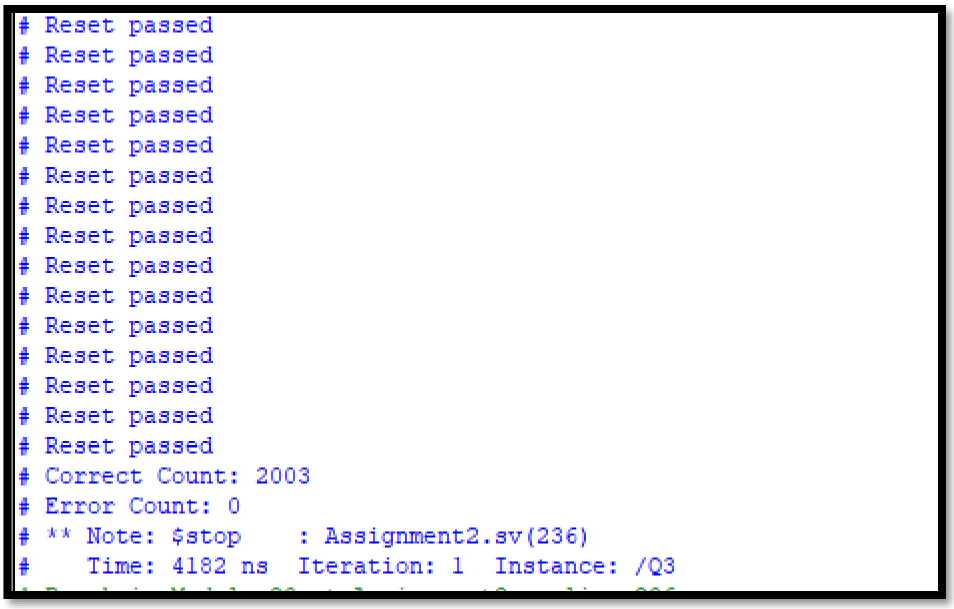

runQ3.do
```
vlib work
vlog package.sv
vlog ALSU.v Assignment2.sv +cover -covercells
vsim -voptargs=+acc Q3 -cover
add wave *
coverage save ALSU_tb.ucdb -onexit -du ALSU
run -all
coverage exclude -src ALSU.v -allfalse -line 76 -code b
quit -sim
vcover report ALSU_tb.ucdb -details -annotate -all -output coverage_rpt.txt
```

❖ Bugs found:

Bug Description	Wrong code	Correction
Internal wire (cin_reg) was defined to be signed, which causes sign extension in the addition processes.	`reg signed [1:0] cin_reg;`	Cin_reg should be unsigned, and no need to have a width more than one bit
The case statement should use the value of opcode_reg	`case (opcode)`	Opcode → opcode_reg

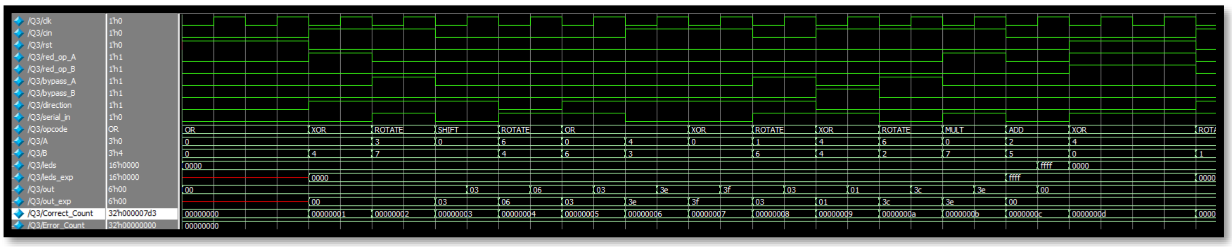
Missing FULL_ADDER parameter check when opcode = add, hence, cin wasn't taken into consideration	`out <= A_reg + B_reg;`	`3'h2:begin`
		`if( FULL_ADDER == "ON") out <= A_reg + B_reg + cin_reg;`
		`else out <= A_reg + B_reg;`
		`end // no full adder?`

❖ Results:

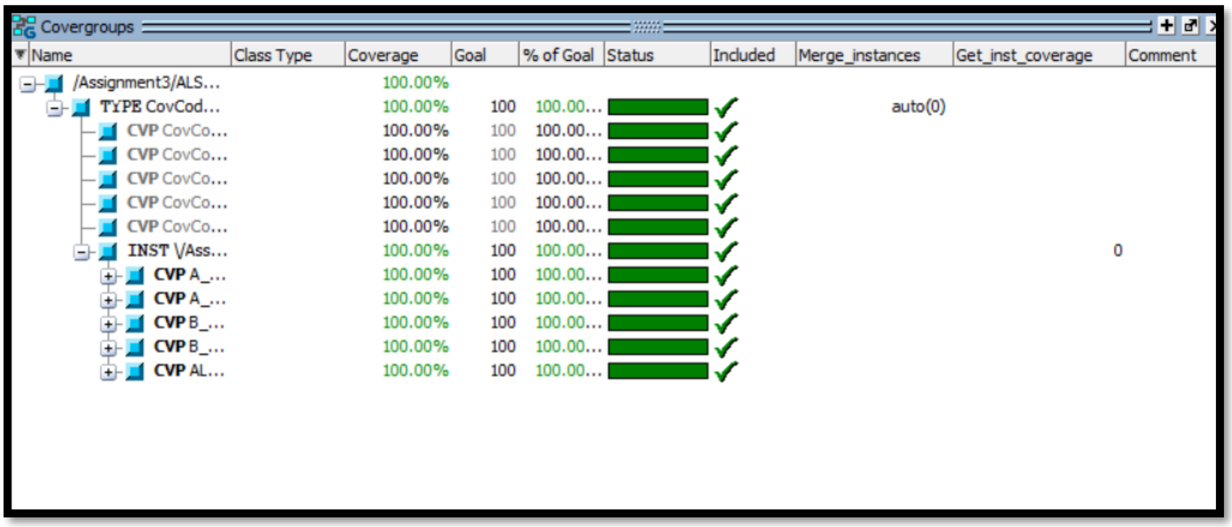


```
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Reset passed
# Correct Count: 2003
# Error Count: 0
# ** Note: $stop    : Assignment2.sv(236)
#    Time: 4182 ns  Iteration: 1  Instance: /Q3
```

Waveform:



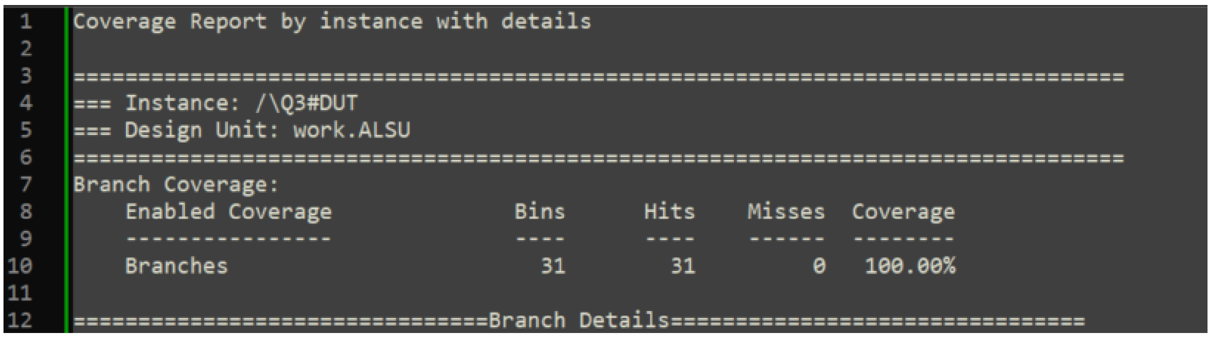
❖ Functional Coverage Report:



Covergroups

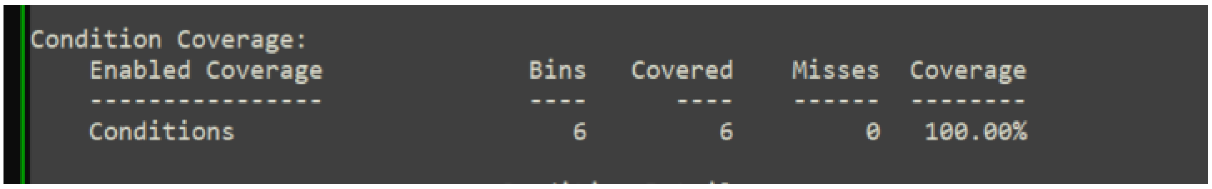
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/Assignment3/ALS...		100.00%							
TYPE CovCod...		100.00%	100	100.00...	🟩	✓	auto(0)		
CVP CovCo...		100.00%	100	100.00...	🟩	✓			
CVP CovCo...		100.00%	100	100.00...	🟩	✓			
CVP CovCo...		100.00%	100	100.00...	🟩	✓			
CVP CovCo...		100.00%	100	100.00...	🟩	✓			
CVP CovCo...		100.00%	100	100.00...	🟩	✓			
INST \/Ass...		100.00%	100	100.00...	🟩	✓		0	
CVP A_...		100.00%	100	100.00...	🟩	✓			
CVP A_...		100.00%	100	100.00...	🟩	✓			
CVP B_...		100.00%	100	100.00...	🟩	✓			
CVP B_...		100.00%	100	100.00...	🟩	✓			
CVP AL...		100.00%	100	100.00...	🟩	✓			

❖ Coverage Report:

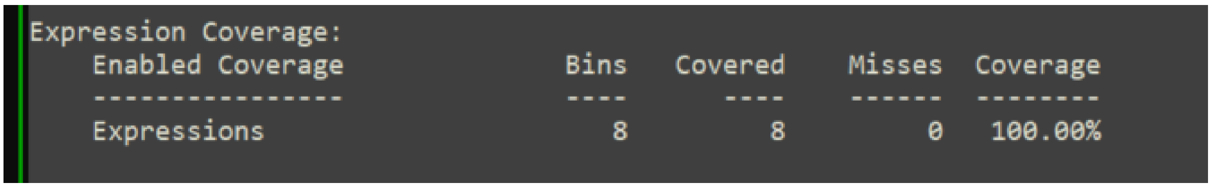


```
Coverage Report by instance with details

=================================================================================
=== Instance: /\Q3#DUT
=== Design Unit: work.ALSU
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        31        31         0   100.00%

================================Branch Details================================
```

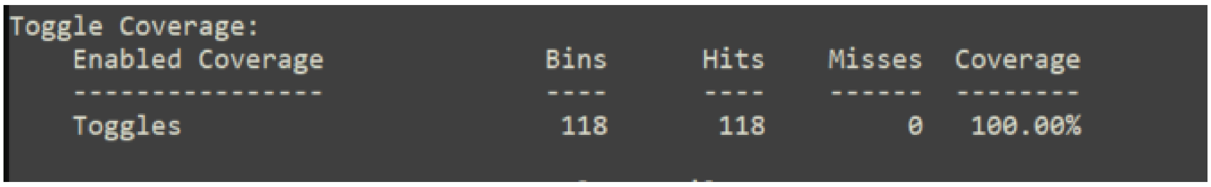
- Excluded **all False case** in the case statement, since invalid opcodes are handled by the flag (**invalid_opcode**)



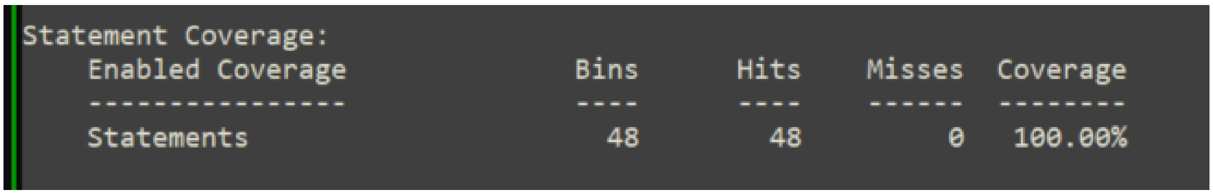
```
Condition Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Conditions                       6          6         0   100.00%
```



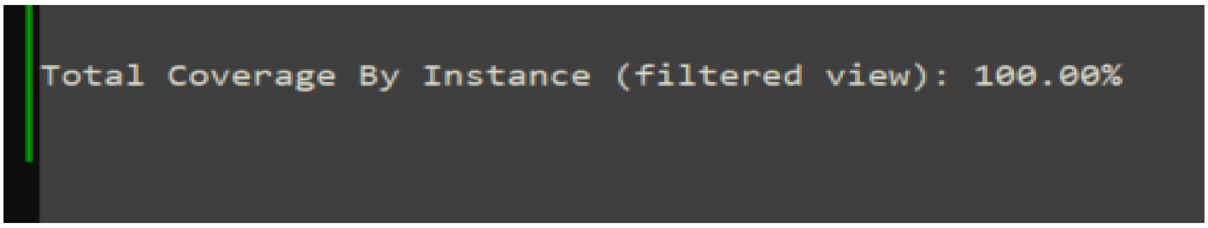
```
Expression Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----       ----    ------  --------
    Expressions                      8          8         0   100.00%
```



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



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



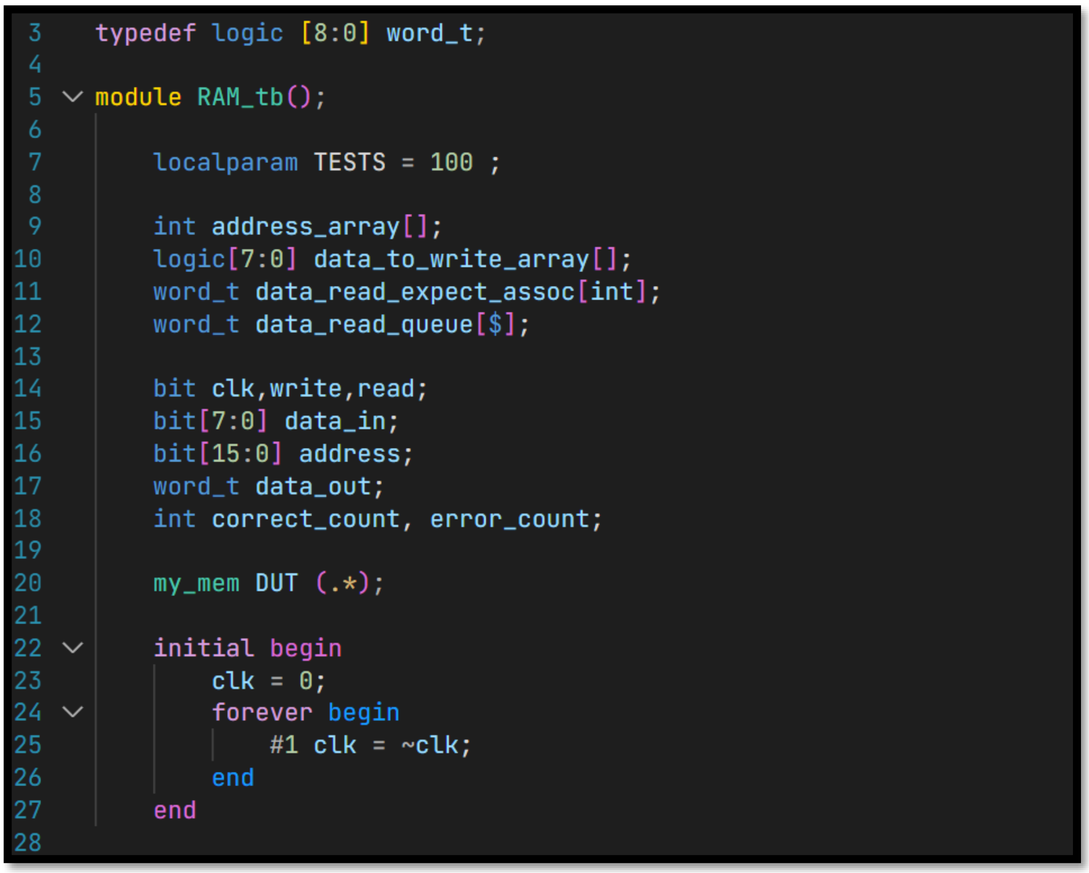
```
Total Coverage By Instance (filtered view): 100.00%
```

Q4) RAM

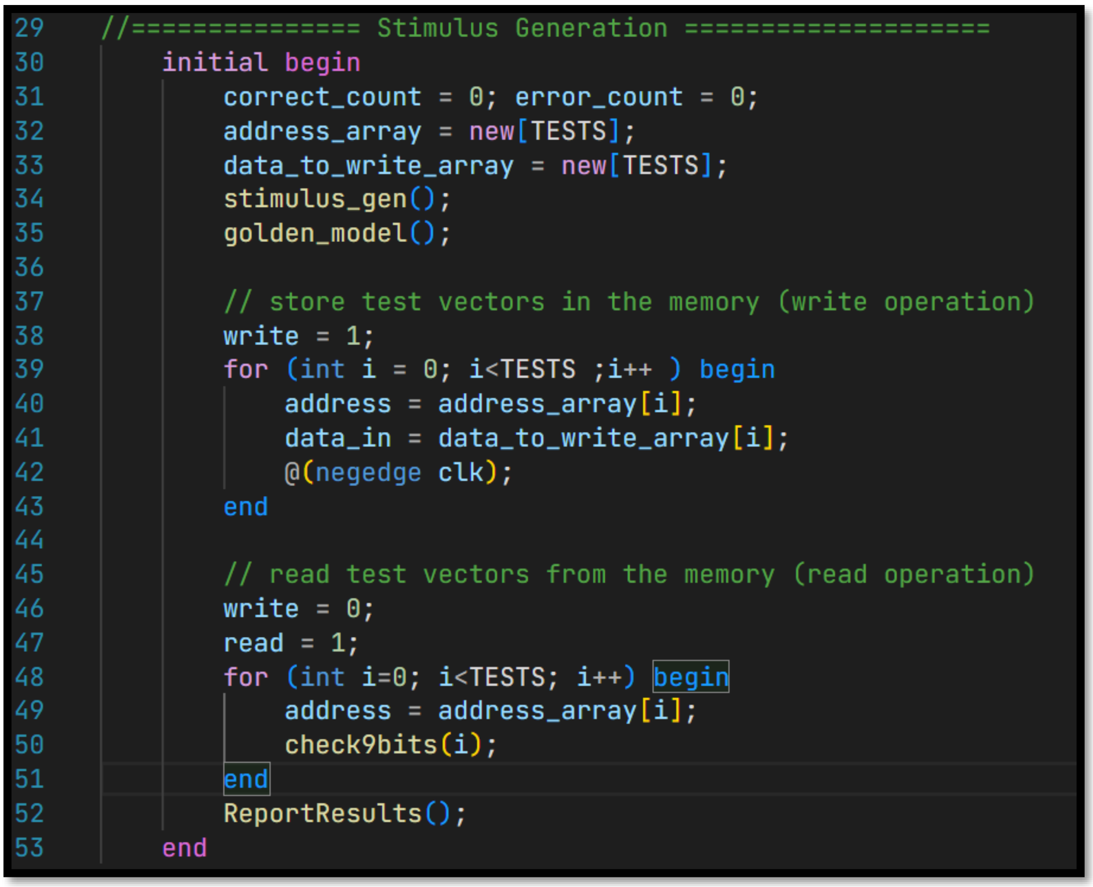
❖ Verification Plan:

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RAM1	Check write functionality by generating 100 random values to be written on 100 different memory locations.	Random during the simulation	-	Comparing results to a refrence golden model
RAM2	Check read functionality by reading the random 100 test verctors written and compare them to the excpected values.	Random during the simulation	-	Comparing results to a refrence golden model

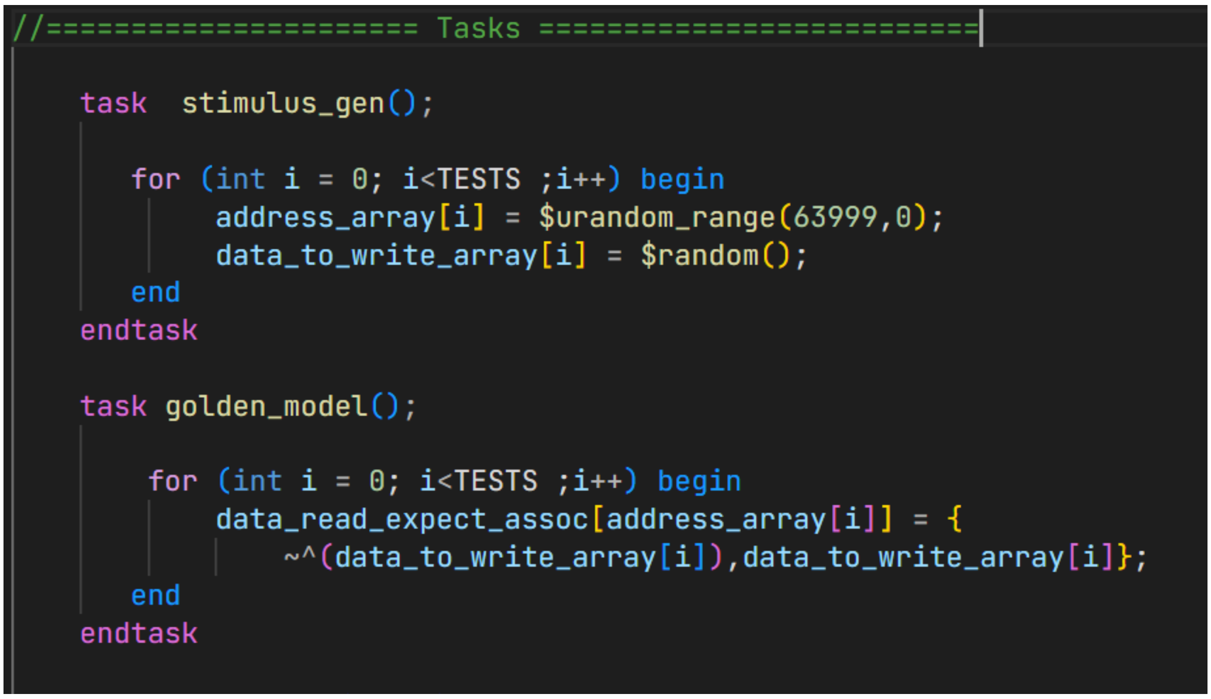
❖ TestBench:



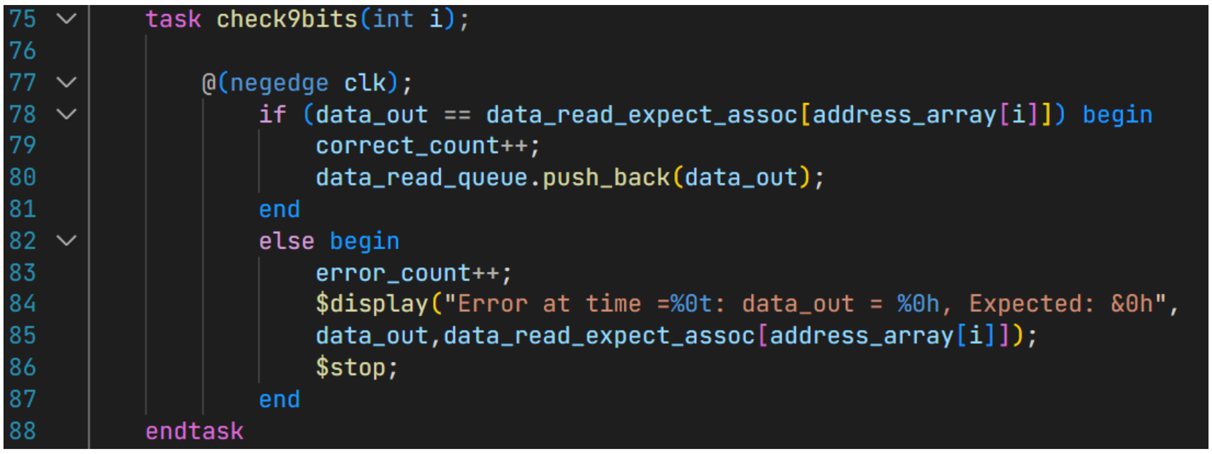
```
typedef logic [8:0] word_t;

module RAM_tb();

    localparam TESTS = 100 ;

    int address_array[];
    logic[7:0] data_to_write_array[];
    word_t data_read_expect_assoc[int];
    word_t data_read_queue[$];

    bit clk,write,read;
    bit[7:0] data_in;
    bit[15:0] address;
    word_t data_out;
    int correct_count, error_count;

    my_mem DUT (.*);

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

```



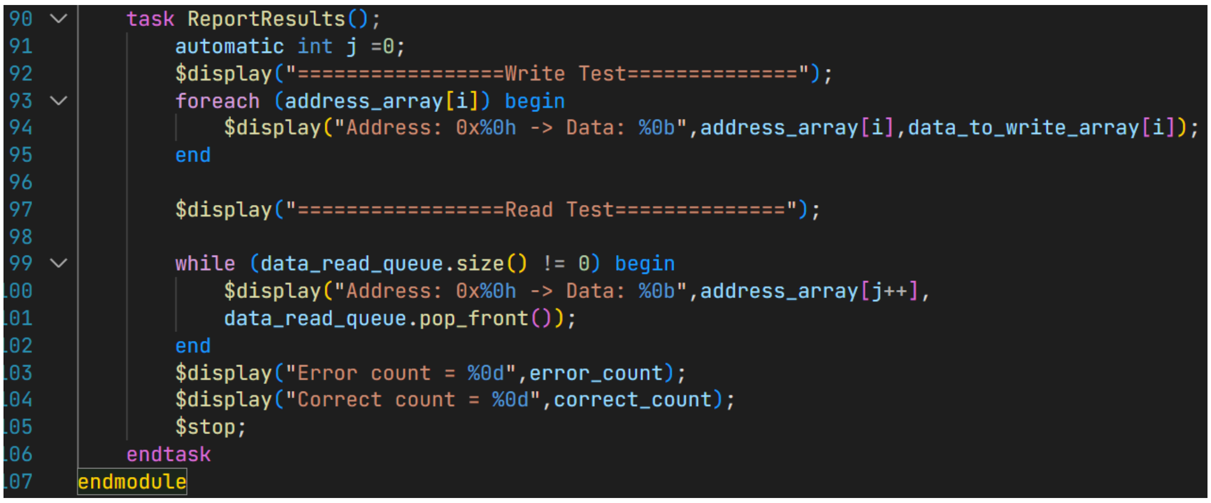
```
//=============== Stimulus Generation ====================
    initial begin
        correct_count = 0; error_count = 0;
        address_array = new[TESTS];
        data_to_write_array = new[TESTS];
        stimulus_gen();
        golden_model();

        // store test vectors in the memory (write operation)
        write = 1;
        for (int i = 0; i<TESTS ;i++ ) begin
            address = address_array[i];
            data_in = data_to_write_array[i];
            @(negedge clk);
        end

        // read test vectors from the memory (read operation)
        write = 0;
        read = 1;
        for (int i=0; i<TESTS; i++) begin
            address = address_array[i];
            check9bits(i);
        end
        ReportResults();
    end
```



```
//===================== Tasks ========================

    task  stimulus_gen();

      for (int i = 0; i<TESTS ;i++) begin
           address_array[i] = $urandom_range(63999,0);
           data_to_write_array[i] = $random();
      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]),data_to_write_array[i]};
      end
    endtask
```

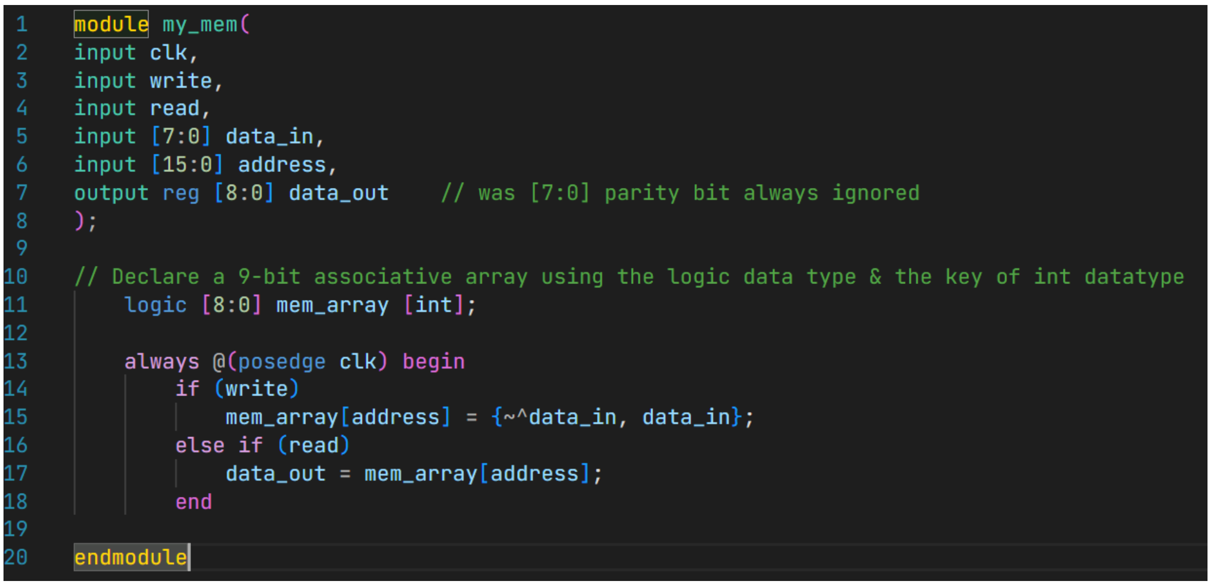


```
    task check9bits(int i);

        @(negedge clk);
            if (data_out == data_read_expect_assoc[address_array[i]]) begin
                correct_count++;
                data_read_queue.push_back(data_out);
            end
            else begin
                error_count++;
                $display("Error at time =%0t: data_out = %0h, Expected: &0h",
                data_out,data_read_expect_assoc[address_array[i]]);
                $stop;
            end
    endtask
```



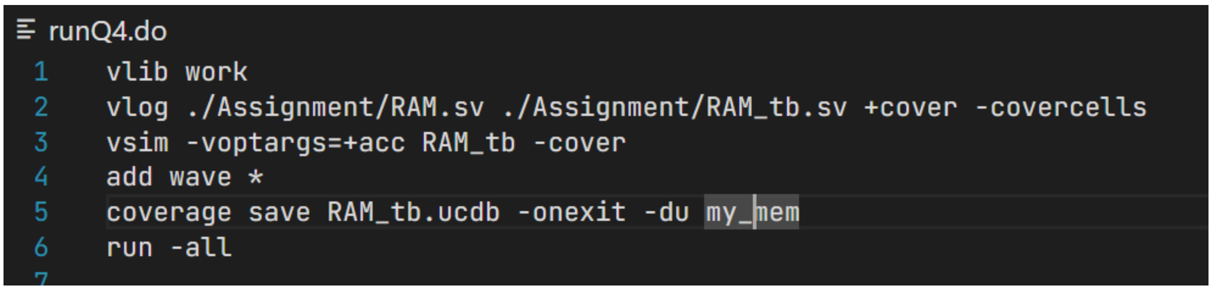
```
    task ReportResults();
        automatic int j =0;
        $display("================Write Test=============");
        foreach (address_array[i]) begin
            $display("Address: 0x%0h -> Data: %0b",address_array[i],data_to_write_array[i]);
        end

        $display("================Read Test=============");

        while (data_read_queue.size() != 0) begin
            $display("Address: 0x%0h -> Data: %0b",address_array[j++],
            data_read_queue.pop_front());
        end
        $display("Error count = %0d",error_count);
        $display("Correct count = %0d",correct_count);
        $stop;
    endtask
endmodule
```

❖ Fixed Design:



```
module my_mem(
input clk,
input write,
input read,
input [7:0] data_in,
input [15:0] address,
output reg [8:0] data_out    // was [7:0] parity bit always ignored
);

// Declare a 9-bit associative array using the logic data type & the key of int datatype
    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
```

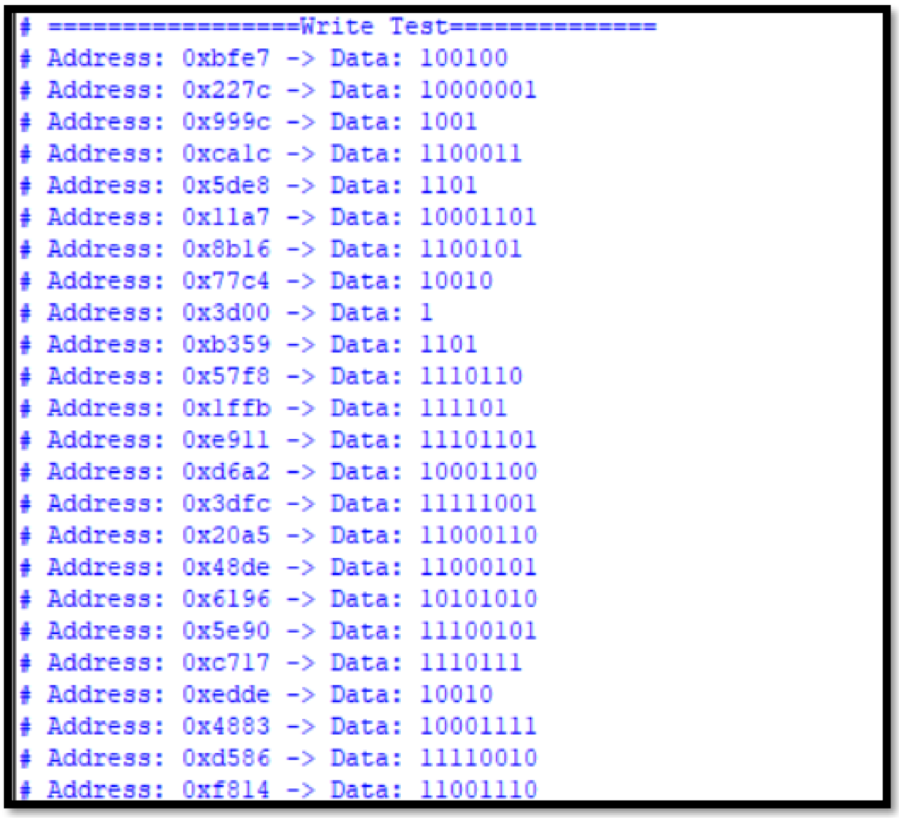
❖ Do file:



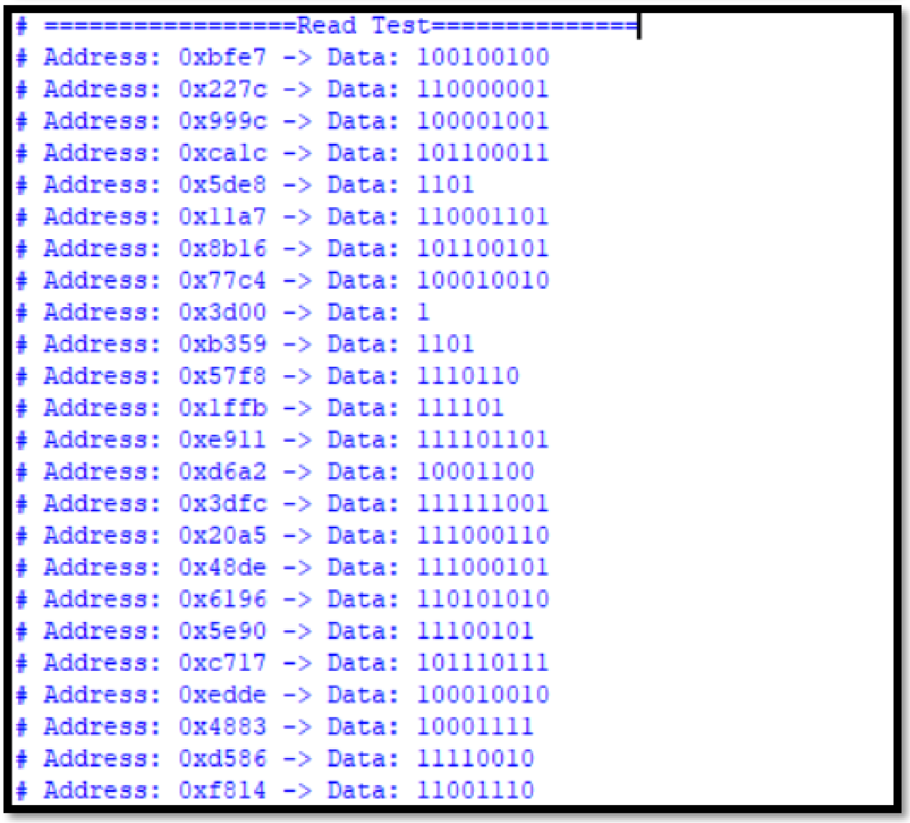
runQ4.do

```
vlib work
vlog ./Assignment/RAM.sv ./Assignment/RAM_tb.sv +cover -covercells
vsim -voptargs=+acc RAM_tb -cover
add wave *
coverage save RAM_tb.ucdb -onexit -du my_mem
run -all
```

❖ Results:

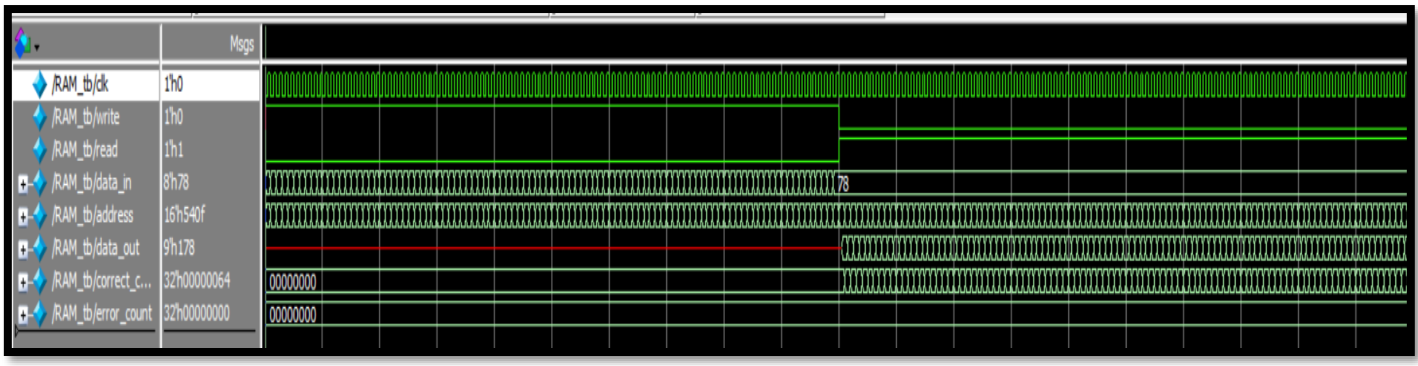


```
# =================Write Test==============
# Address: 0xbfe7 -> Data: 100100
# Address: 0x227c -> Data: 10000001
# Address: 0x999c -> Data: 1001
# Address: 0xca1c -> Data: 1100011
# Address: 0x5de8 -> Data: 1101
# Address: 0x11a7 -> Data: 10001101
# Address: 0x8b16 -> Data: 1100101
# Address: 0x77c4 -> Data: 10010
# Address: 0x3d00 -> Data: 1
# Address: 0xb359 -> Data: 1101
# Address: 0x57f8 -> Data: 1110110
# Address: 0x1ffb -> Data: 111101
# Address: 0xe911 -> Data: 11101101
# Address: 0xd6a2 -> Data: 10001100
# Address: 0x3dfc -> Data: 11111001
# Address: 0x20a5 -> Data: 11000110
# Address: 0x48de -> Data: 11000101
# Address: 0x6196 -> Data: 10101010
# Address: 0x5e90 -> Data: 11100101
# Address: 0xc717 -> Data: 1110111
# Address: 0xedde -> Data: 10010
# Address: 0x4883 -> Data: 10001111
# Address: 0xd586 -> Data: 11110010
# Address: 0xf814 -> Data: 11001110
```



```
# =================Read Test==============
# Address: 0xbfe7 -> Data: 100100100
# Address: 0x227c -> Data: 110000001
# Address: 0x999c -> Data: 100001001
# Address: 0xca1c -> Data: 101100011
# Address: 0x5de8 -> Data: 1101
# Address: 0x11a7 -> Data: 110001101
# Address: 0x8b16 -> Data: 101100101
# Address: 0x77c4 -> Data: 100010010
# Address: 0x3d00 -> Data: 1
# Address: 0xb359 -> Data: 1101
# Address: 0x57f8 -> Data: 1110110
# Address: 0x1ffb -> Data: 111101
# Address: 0xe911 -> Data: 111101101
# Address: 0xd6a2 -> Data: 10001100
# Address: 0x3dfc -> Data: 111111001
# Address: 0x20a5 -> Data: 111000110
# Address: 0x48de -> Data: 111000101
# Address: 0x6196 -> Data: 110101010
# Address: 0x5e90 -> Data: 11100101
# Address: 0xc717 -> Data: 101110111
# Address: 0xedde -> Data: 100010010
# Address: 0x4883 -> Data: 10001111
# Address: 0xd586 -> Data: 11110010
# Address: 0xf814 -> Data: 11001110
```

<u>Waveform</u>**:**



	Msgs
/RAM_tb/clk	1'h0
/RAM_tb/write	1'h0
/RAM_tb/read	1'h1
/RAM_tb/data_in	8'h78
/RAM_tb/address	16'h540f
/RAM_tb/data_out	9'h178
/RAM_tb/correct_c...	32'h00000064
/RAM_tb/error_count	32'h00000000