P1)

1. Testbench code

```
import testing_pkg::*;
module ALU_tb();
    Transaction obj = new ;
    byte operand1 , operand2 ;
    bit clk , rst ;
    opcode_e opcode ;
    byte out ;
    byte expected_out ;


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

    integer correct_count , error_count ;

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

    initial begin
        rst = 0 ;
        correct_count = 0 ;
        error_count = 0 ;
        check_result () ;
        repeat (30000) begin
            assert (obj.randomize()) ;
            operand1 = obj.operand1 ;
            operand2 = obj.operand2 ;
            opcode   = obj.opcode   ;
            golden_model () ;
            check_result () ;
        end

        check_result() ;

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

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

    task check_result ();
        @(negedge clk) ;
        if (expected_out == out) begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
    endtask

    task assert_reset();
        rst = 1 ;
        expected_out = 0 ;
        @(negedge clk) ;
        if (expected_out == out) begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
        rst = 0 ;
    endtask
endmodule
```

2. Package code

```
package testing_pkg;
    typedef enum logic [1:0] { ADD , SUB , MULT , DIV } opcode_e;

    class Transaction;
        rand opcode_e opcode ;
        rand byte operand1 ;
        rand byte operand2 ;
        bit clk ;

        constraint operand1_constraint {
            operand1 dist {127:/25 , 0:/25 , -128:/25 , [1:126]:/15 , [-127:-1]:/10} ;
        }

        constraint operand2_constraint {
            operand2 dist {127:/25 , 0:/25 , -128:/25 , [1:126]:/15 , [-127:-1]:/10} ;
        }

        covergroup covcode @(posedge clk) ;
            cv1: coverpoint opcode {
                bins opbin1  = {ADD,SUB} ;
                bins opbin2  = (ADD => SUB)  ;
                bins op_ADD  = {ADD} ;
                bins op_SUB  = {SUB} ;
                bins op_MULT = {MULT} ;
                bins opbin3  = default ;
                illegal_bins invalid = {DIV} ;
            }

            cv2: coverpoint operand1 {
                bins max_pos = {127}   ;
                bins ZERO    = {0}     ;
                bins max_neg = {-128}  ;
                bins other   = default ;
            }

            cv3: coverpoint operand2 {
                bins max_pos = {127}   ;
                bins ZERO    = {0}     ;
                bins max_neg = {-128}  ;
                bins other   = default ;
            }

            cv4 : cross cv1 ,cv2 , cv3 {
                bins max_pos1_max_pos2_ADD  = binsof (cv1.op_ADD) && binsof (cv2.max_pos) && binsof (cv3.max_pos) ;
                bins max_pos1_max_neg2_ADD  = binsof (cv1.op_ADD) && binsof (cv2.max_pos) && binsof (cv3.max_neg) ;
                bins max_pos1_zero2_ADD     = binsof (cv1.op_ADD) && binsof (cv2.max_pos) && binsof (cv3.ZERO) ;
                bins max_neg1_max_pos2_ADD  = binsof (cv1.op_ADD) && binsof (cv2.max_neg) && binsof (cv3.max_pos) ;
                bins max_neg1_max_neg2_ADD  = binsof (cv1.op_ADD) && binsof (cv2.max_neg) && binsof (cv3.max_neg) ;
                bins max_neg1_zero2_ADD     = binsof (cv1.op_ADD) && binsof (cv2.max_neg) && binsof (cv3.ZERO) ;
                bins zero1_max_pos2_ADD     = binsof (cv1.op_ADD) && binsof (cv2.ZERO)    && binsof (cv3.max_pos) ;
                bins zero1_max_neg2_ADD     = binsof (cv1.op_ADD) && binsof (cv2.ZERO)    && binsof (cv3.max_neg) ;
                bins zero1_zero2_ADD        = binsof (cv1.op_ADD) && binsof (cv2.ZERO)    && binsof (cv3.ZERO) ;

                bins max_pos1_max_pos2_SUB  = binsof (cv1.op_SUB) && binsof (cv2.max_pos) && binsof (cv3.max_pos) ;
                bins max_pos1_max_neg2_SUB  = binsof (cv1.op_SUB) && binsof (cv2.max_pos) && binsof (cv3.max_neg) ;
                bins max_pos1_zero2_SUB     = binsof (cv1.op_SUB) && binsof (cv2.max_pos) && binsof (cv3.ZERO) ;
                bins max_neg1_max_pos2_SUB  = binsof (cv1.op_SUB) && binsof (cv2.max_neg) && binsof (cv3.max_pos) ;
                bins max_neg1_max_neg2_SUB  = binsof (cv1.op_SUB) && binsof (cv2.max_neg) && binsof (cv3.max_neg) ;
                bins max_neg1_zero2_SUB     = binsof (cv1.op_SUB) && binsof (cv2.max_neg) && binsof (cv3.ZERO) ;
                bins zero1_max_pos2_SUB     = binsof (cv1.op_SUB) && binsof (cv2.ZERO)    && binsof (cv3.max_pos) ;
                bins zero1_max_neg2_SUB     = binsof (cv1.op_SUB) && binsof (cv2.ZERO)    && binsof (cv3.max_neg) ;
                bins zero1_zero2_SUB        = binsof (cv1.op_SUB) && binsof (cv2.ZERO)    && binsof (cv3.ZERO) ;

                bins max_pos1_max_pos2_MULT  = binsof (cv1.op_MULT) && binsof (cv2.max_pos) && binsof (cv3.max_pos) ;
                bins max_pos1_max_neg2_MULT  = binsof (cv1.op_MULT) && binsof (cv2.max_pos) && binsof (cv3.max_neg) ;
                bins max_pos1_zero2_MULT     = binsof (cv1.op_MULT) && binsof (cv2.max_pos) && binsof (cv3.ZERO) ;
                bins max_neg1_max_pos2_MULT  = binsof (cv1.op_MULT) && binsof (cv2.max_neg) && binsof (cv3.max_pos) ;
                bins max_neg1_max_neg2_MULT  = binsof (cv1.op_MULT) && binsof (cv2.max_neg) && binsof (cv3.max_neg) ;
                bins max_neg1_zero2_MULT     = binsof (cv1.op_MULT) && binsof (cv2.max_neg) && binsof (cv3.ZERO) ;
                bins zero1_max_pos2_MULT     = binsof (cv1.op_MULT) && binsof (cv2.ZERO)    && binsof (cv3.max_pos) ;
                bins zero1_max_neg2_MULT     = binsof (cv1.op_MULT) && binsof (cv2.ZERO)    && binsof (cv3.max_neg) ;
                bins zero1_zero2_MULT        = binsof (cv1.op_MULT) && binsof (cv2.ZERO)    && binsof (cv3.ZERO) ;
            }
        endgroup

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

    endclass
endpackage
```

3. Design code

```
import testing_pkg::*;
module alu_seq(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
```

4. Snippet to your verification plan document

A	B	C	D	E
Label	**Design Requirement Description**	**Stimulus Generation**	**Functional Coverage**	**Functionality Check**
TEST_1	When the reset is asserted by task **assert_reset**, the output value should be low	Directed at the start of the simulation, then randomized with constraint that drive the reset to be most of the simulation off		A checker in the testbench to make sure the output is correct
TEST_2	when the reset is deasserted the output of the design should be equal the output of golden model	Randomized under constrains on the A & B values to be 25% of simulation with MAXPOS value , 25% of simulation with MAXNEG value, 25% of simulation with ZERO and remain of simulation with other values	Ensure all values of operand A are covered. Ensure all values of operand B are covered. Ensure all opcode operations are exercised. Ensure all maximum combinations of A, B, and opcode operations are covered.	A checker in the testbench to make sure the output is correct
TEST_3	When the reset is asserted, the output value should be low	Directed at the end of the simulation		A checker in the testbench to make sure the output is correct

5. Do file

```
vlib work
vlog alu.svh package.svh module.svh +cover -covercells
vsim -voptargs=+acc work.ALU_tb -cover
add wave *
coverage save alu_seq.ucdb -onexit -du alu_seq
run -all
```

6. Code Coverage report snippets

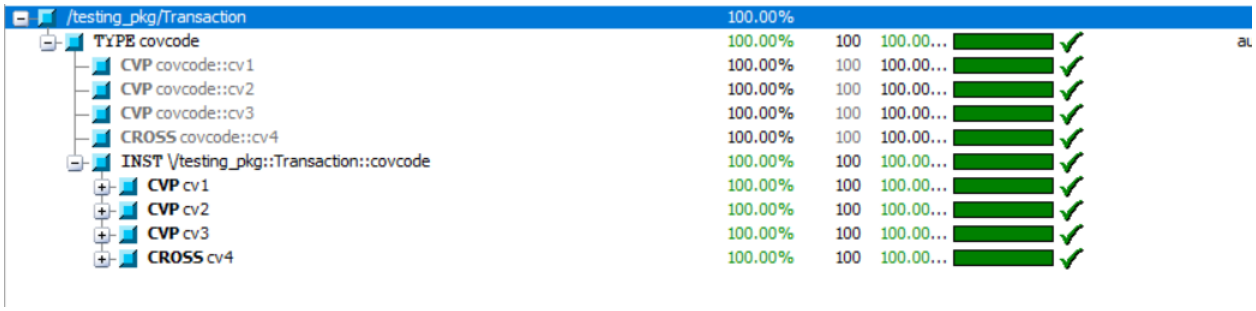
```
TYPE /testing_pkg/Transaction/covcode                100.00%        100          -    Covered
   covered/total bins:                                    56         56          -
   missing/total bins:                                     0         56          -
   % Hit:                                             100.00%        100          -
   Coverpoint cv1                                     100.00%        100          -    Covered
       covered/total bins:                                 5          5          -
       missing/total bins:                                 0          5          -
       % Hit:                                         100.00%        100          -
   Coverpoint cv2                                     100.00%        100          -    Covered
       covered/total bins:                                 3          3          -
       missing/total bins:                                 0          3          -
       % Hit:                                         100.00%        100          -
   Coverpoint cv3                                     100.00%        100          -    Covered
       covered/total bins:                                 3          3          -
       missing/total bins:                                 0          3          -
       % Hit:                                         100.00%        100          -
   Cross cv4                                          100.00%        100          -    Covered
       covered/total bins:                                45         45          -
       missing/total bins:                                 0         45          -
       % Hit:                                         100.00%        100          -
Covergroup instance \/testing_pkg::Transaction::covcode
                                                      100.00%        100          -    Covered
   covered/total bins:                                    56         56          -
   missing/total bins:                                     0         56          -
   % Hit:                                             100.00%        100          -
   Coverpoint cv1                                     100.00%        100          -    Covered
       covered/total bins:                                 5          5          -
       missing/total bins:                                 0          5          -
       % Hit:                                         100.00%        100          -
       illegal_bin invalid                               7568                    -    Occurred
       bin opbin1                                       14787          1          -    Covered
       bin opbin2                                        1747          1          -    Covered
       bin op_ADD                                        7467          1          -    Covered
       bin op_SUB                                        7320          1          -    Covered
       bin op_MULT                                       7646          1          -    Covered
       default bin opbin3                                   0                    -    ZERO
   Coverpoint cv2                                     100.00%        100          -    Covered
       covered/total bins:                                 3          3          -
       missing/total bins:                                 0          3          -
       % Hit:                                         100.00%        100          -
       bin max_pos                                       7409          1          -    Covered
       bin ZERO                                          7510          1          -    Covered
       bin max_neg                                       7511          1          -    Covered
       default bin other                                 7571                    -    Occurred
```

Coverpoint cv3	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
% Hit:	100.00%	100	-	
bin max_pos	7420	1	-	Covered
bin ZERO	7574	1	-	Covered
bin max_neg	7486	1	-	Covered
default bin other	7521		-	Occurred
Cross cv4	100.00%	100	-	Covered
covered/total bins:	45	45	-	
missing/total bins:	0	45	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin max_pos1_max_pos2_ADD	454	1	-	Covered
bin max_pos1_max_neg2_ADD	470	1	-	Covered
bin max_pos1_zero2_ADD	452	1	-	Covered
bin max_neg1_max_pos2_ADD	465	1	-	Covered
bin max_neg1_max_neg2_ADD	503	1	-	Covered
bin max_neg1_zero2_ADD	463	1	-	Covered
bin zero1_max_pos2_ADD	460	1	-	Covered
bin zero1_max_neg2_ADD	471	1	-	Covered
bin zero1_zero2_ADD	490	1	-	Covered
bin max_pos1_max_pos2_SUB	450	1	-	Covered
bin max_pos1_max_neg2_SUB	427	1	-	Covered
bin max_pos1_zero2_SUB	475	1	-	Covered
bin max_neg1_max_pos2_SUB	479	1	-	Covered
bin max_neg1_max_neg2_SUB	503	1	-	Covered
bin max_neg1_zero2_SUB	434	1	-	Covered
bin zero1_max_pos2_SUB	436	1	-	Covered
bin zero1_max_neg2_SUB	455	1	-	Covered
bin zero1_zero2_SUB	461	1	-	Covered
bin max_pos1_max_pos2_MULT	415	1	-	Covered
bin max_pos1_max_neg2_MULT	466	1	-	Covered
bin max_pos1_zero2_MULT	490	1	-	Covered
bin max_neg1_max_pos2_MULT	503	1	-	Covered
bin max_neg1_max_neg2_MULT	465	1	-	Covered
bin max_neg1_zero2_MULT	458	1	-	Covered
bin zero1_max_pos2_MULT	503	1	-	Covered
bin zero1_max_neg2_MULT	465	1	-	Covered
bin zero1_zero2_MULT	493	1	-	Covered
bin <opbin2,max_neg,max_neg>	110	1	-	Covered
bin <opbin1,max_neg,max_neg>	1006	1	-	Covered
bin <opbin2,max_neg,ZERO>	105	1	-	Covered
bin <opbin1,max_neg,ZERO>	897	1	-	Covered
bin <opbin2,max_neg,max_pos>	112	1	-	Covered
bin <opbin1,max_neg,max_pos>	944	1	-	Covered
bin <opbin2,ZERO,max_neg>	109	1	-	Covered
bin <opbin1,ZERO,max_neg>	926	1	-	Covered
bin <opbin2,max_pos,max_neg>	95	1	-	Covered
bin <opbin1,max_pos,max_neg>	897	1	-	Covered
bin <opbin2,ZERO,ZERO>	105	1	-	Covered
bin <opbin1,ZERO,ZERO>	951	1	-	Covered
bin <opbin2,max_pos,ZERO>	99	1	-	Covered
bin <opbin1,max_pos,ZERO>	927	1	-	Covered
bin <opbin2,ZERO,max_pos>	101	1	-	Covered
bin <opbin1,ZERO,max_pos>	896	1	-	Covered

bin <opbin2,max_pos,max_neg>	95	1	-	Covered
bin <opbin1,max_pos,max_neg>	897	1	-	Covered
bin <opbin2,ZERO,ZERO>	105	1	-	Covered
bin <opbin1,ZERO,ZERO>	951	1	-	Covered
bin <opbin2,max_pos,ZERO>	99	1	-	Covered
bin <opbin1,max_pos,ZERO>	927	1	-	Covered
bin <opbin2,ZERO,max_pos>	101	1	-	Covered
bin <opbin1,ZERO,max_pos>	896	1	-	Covered
bin <opbin2,max_pos,max_pos>	120	1	-	Covered
bin <opbin1,max_pos,max_pos>	904	1	-	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

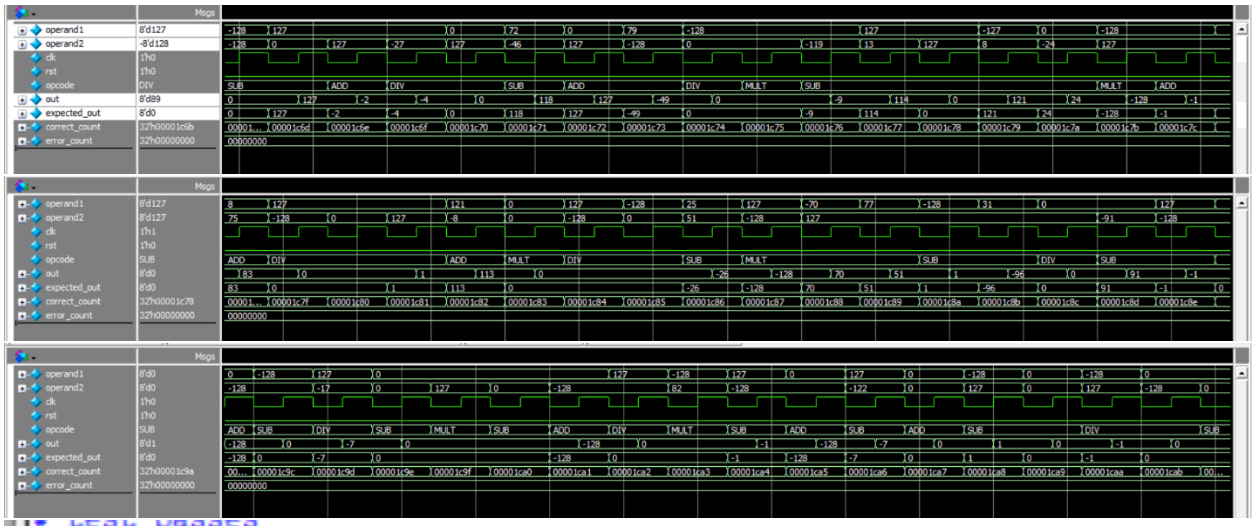


7. Functional Coverage report snippets (Questions 1, 2, and 3 only)

```
# Coverage Report by instance with details
#
# ================================================================================
# === Instance: /\ALU_tb#DUT
# === Design Unit: work.alu_seq
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                         6         6         0    100%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\ALU_tb#DUT
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File alu.svh
# ------------------------------------IF Branch------------------------------------
#     9                                      29935     Count coming in to IF
#     9               1                          2      if (rst)
#
#     11              1                      29933      else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ----------------------------------CASE Branch----------------------------------
#     12                                     29933     Count coming in to CASE
#     13              1                       7509                       ADD: out <= operand1 + operand2;
#
#     14              1                       7515                       SUB: out <= operand1 - operand2;
#
#     15              1                       7464                       MULT:out <= operand1 * operand2;
#
#     16              1                       7445                       DIV: out <= operand1 / operand2;
#
# Branch totals: 4 hits of 4 branches = 100.00%
#
#
```

```
#
# 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           2      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%
#
# End time: 20:22:34 on Sep 09,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

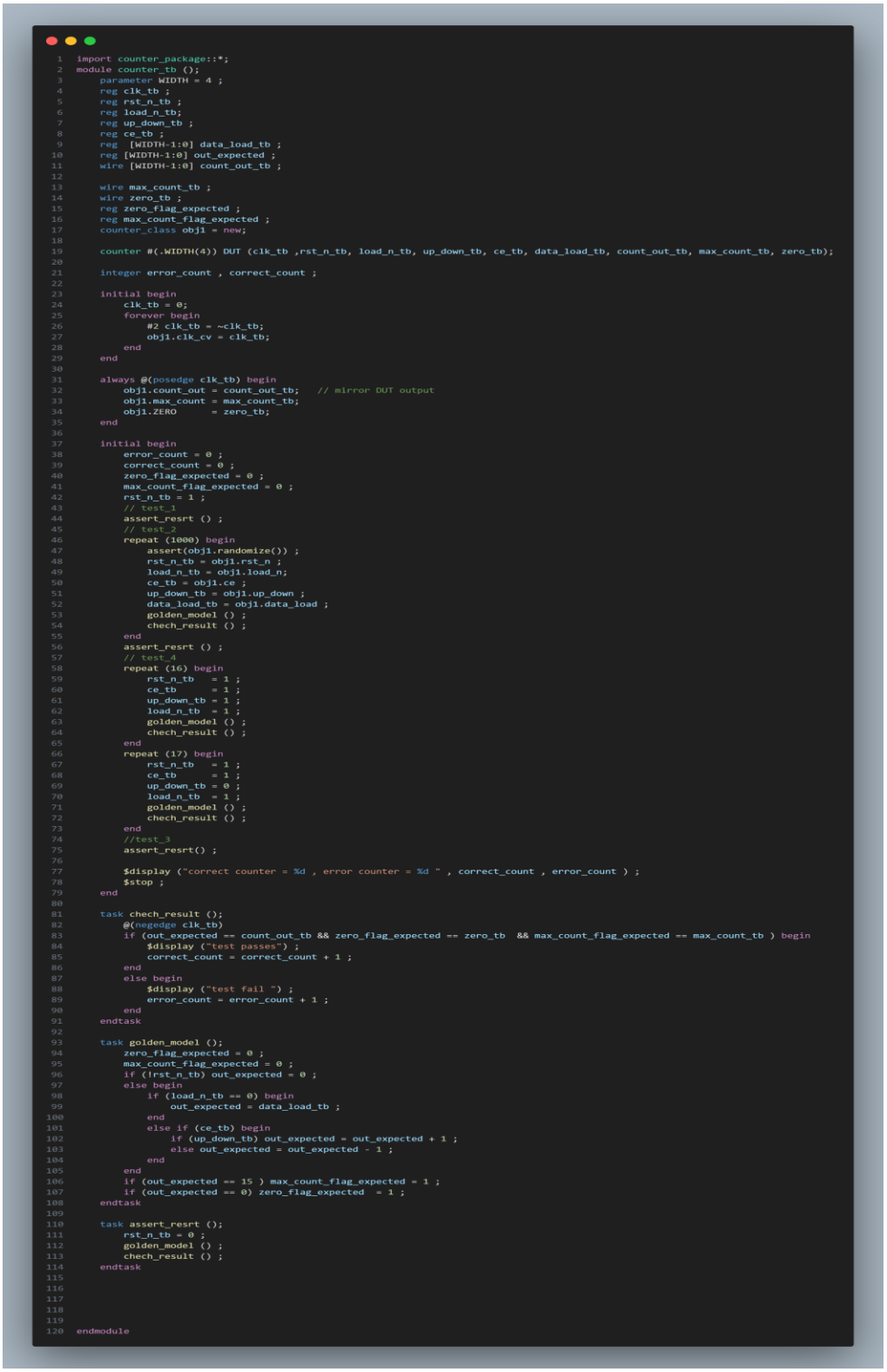
8. Clear and neat QuestaSim waveform snippets showing the functionality of the design



```
# test passes
# test passes
# test passes
# test passes
# test passes
# test passes
# correct counter =       30000 , error counter =           0
# ** Note: $stop    : module.svh(39)
#    Time: 120004 ns  Iteration: 1  Instance: /ALU_tb
```

P2)

1. Testbench code



```
import counter_package::*;
module counter_tb ();
    parameter WIDTH = 4 ;
    reg clk_tb ;
    reg rst_n_tb ;
    reg load_n_tb;
    reg up_down_tb ;
    reg ce_tb ;
    reg  [WIDTH-1:0] data_load_tb ;
    reg [WIDTH-1:0] out_expected ;
    wire [WIDTH-1:0] count_out_tb ;

    wire max_count_tb ;
    wire zero_tb ;
    reg zero_flag_expected ;
    reg max_count_flag_expected ;
    counter_class obj1 = new;

    counter #(.WIDTH(4)) DUT (clk_tb ,rst_n_tb, load_n_tb, up_down_tb, ce_tb, data_load_tb, count_out_tb, max_count_tb, zero_tb);

    integer error_count , correct_count ;

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

    always @(posedge clk_tb) begin
        obj1.count_out = count_out_tb;   // mirror DUT output
        obj1.max_count = max_count_tb;
        obj1.ZERO      = zero_tb;
    end

    initial begin
        error_count = 0 ;
        correct_count = 0 ;
        zero_flag_expected = 0 ;
        max_count_flag_expected = 0 ;
        rst_n_tb = 1 ;
        // test_1
        assert_resrt () ;
        // test_2
        repeat (1000) begin
            assert(obj1.randomize()) ;
            rst_n_tb = obj1.rst_n ;
            load_n_tb = obj1.load_n;
            ce_tb = obj1.ce ;
            up_down_tb = obj1.up_down ;
            data_load_tb = obj1.data_load ;
            golden_model () ;
            chech_result () ;
        end
        assert_resrt () ;
        // test_4
        repeat (16) begin
            rst_n_tb   = 1 ;
            ce_tb      = 1 ;
            up_down_tb = 1 ;
            load_n_tb  = 1 ;
            golden_model () ;
            chech_result () ;
        end
        repeat (17) begin
            rst_n_tb   = 1 ;
            ce_tb      = 1 ;
            up_down_tb = 0 ;
            load_n_tb  = 1 ;
            golden_model () ;
            chech_result () ;
        end
        //test_3
        assert_resrt() ;

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

    task chech_result ();
        @(negedge clk_tb)
        if (out_expected == count_out_tb && zero_flag_expected == zero_tb  && max_count_flag_expected == max_count_tb ) begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
    endtask

    task golden_model ();
        zero_flag_expected = 0 ;
        max_count_flag_expected = 0 ;
        if (!rst_n_tb) out_expected = 0 ;
        else begin
            if (load_n_tb == 0) begin
                out_expected = data_load_tb ;
            end
            else if (ce_tb) begin
                if (up_down_tb) out_expected = out_expected + 1 ;
                else out_expected = out_expected - 1 ;
            end
        end
        if (out_expected == 15 ) max_count_flag_expected = 1 ;
        if (out_expected == 0) zero_flag_expected  = 1 ;
    endtask

    task assert_resrt ();
        rst_n_tb = 0 ;
        golden_model () ;
        chech_result () ;
    endtask




endmodule
```

2. Package code

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

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

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

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

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

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

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

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

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

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

            count_out_cv_5 : coverpoint count_out {
                bins full_count = (0 => 1 => 2 => 3 => 4 => 5 => 6 => 7 => 8 => 9
                        => 10 => 11 => 12 => 13 => 14 => 15);
            }

            count_out_cv_6 : coverpoint count_out {
                bins full_down = (15 => 14 => 13 => 12 => 11 => 10 => 9 => 8 => 7
                      => 6 => 5 => 4 => 3 => 2 => 1 => 0);
            }

            max_count_cv   : coverpoint max_count ;

            zero_cv        : coverpoint ZERO ;

        endgroup

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

    endclass
endpackage
```

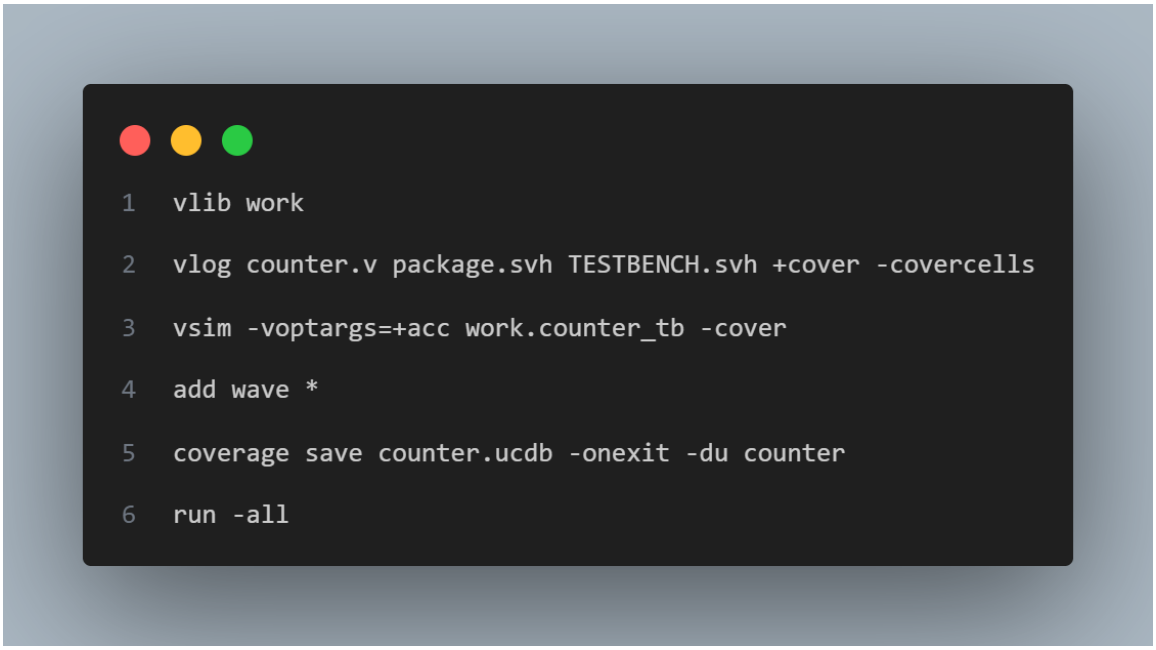
3. Design code

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: Counter Design
//
////////////////////////////////////////////////////////////////////////////////
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
```

4. Snippet to your verification plan document

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
TEST_1	When the reset is asserted by task **assert_reset**, the output value should be low	Directed at the start of the simulation, then randomized with constraint that drive the reset to be most of the simulation off		A checker in the testbench to make sure the output is correct
TEST_2	when the **load_n** is low and reset is deasserted , the output should take the value of **data_load** input	Randomized under constrains on the **load** signal to be off 70% of simulation	cover all values of **load_n** input signal.	A checker in the testbench to make sure the output is correct
TEST_3	when the **load_n** is high & **ce** is high , the output should increment when up_down signal is high and decrement when up_down signal is low	Randomized under constrains on the **ce** signal to be on 70% of simulation & **up_down** signal to be on 70% of simulation	cover all values of **ce** input signal. cover all values of **up_down** input signal.	A checker in the testbench to make sure the output is correct
TEST_3	When the reset is asserted, the output value should be low	Directed at the end of the simulation		A checker in the testbench to make sure the output is correct

5. Do file



```
vlib work
vlog counter.v package.svh TESTBENCH.svh +cover -covercells
vsim -voptargs=+acc work.counter_tb -cover
add wave *
coverage save counter.ucdb -onexit -du counter
run -all
```

6. Code Coverage report snippets

```
# Coverage Report by instance with details
#
# ================================================================================
# === Instance: /\ALU_tb#DUT
# === Design Unit: work.alu_seq
# ================================================================================
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                         6         6         0    100%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /\ALU_tb#DUT
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File alu.svh
# ------------------------------------IF Branch------------------------------------
#     9                                     29935     Count coming in to IF
#     9              1                          2      if (rst)
#
#     11             1                      29933      else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ----------------------------------CASE Branch----------------------------------
#     12                                    29933     Count coming in to CASE
#     13             1                       7509                      ADD: out <= operand1 + operand2;
#
#     14             1                       7515                      SUB: out <= operand1 - operand2;
#
#     15             1                       7464                      MULT:out <= operand1 * operand2;
#
#     16             1                       7445                      DIV: out <= operand1 / operand2;
#
# Branch totals: 4 hits of 4 branches = 100.00%
#
#
```

```
#
# 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           2      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%
#
# End time: 20:22:34 on Sep 09,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

Covergroup	Metric	Goal	Bins	Status
TYPE /counter_package/counter_class/covcode	100.00%	100	-	Covered
covered/total bins:	56	56	-	
missing/total bins:	0	56	-	
% Hit:	100.00%	100	-	
Coverpoint load_data_cv	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_1	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_2	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_3	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_4	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_5	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint count_out_cv_6	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
Coverpoint max_count_cv	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	

Coverpoint zero_cv	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
Covergroup instance \/counter_package::counter_class::covcode				
	100.00%	100	-	Covered
covered/total bins:	56	56	-	
missing/total bins:	0	56	-	
% Hit:	100.00%	100	-	
Coverpoint load_data_cv	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
bin auto[0]	21	1	-	Covered
bin auto[1]	22	1	-	Covered
bin auto[2]	16	1	-	Covered
bin auto[3]	12	1	-	Covered
bin auto[4]	20	1	-	Covered
bin auto[5]	20	1	-	Covered
bin auto[6]	14	1	-	Covered
bin auto[7]	16	1	-	Covered
bin auto[8]	22	1	-	Covered
bin auto[9]	15	1	-	Covered
bin auto[10]	22	1	-	Covered
bin auto[11]	18	1	-	Covered
bin auto[12]	26	1	-	Covered
bin auto[13]	20	1	-	Covered
bin auto[14]	15	1	-	Covered
bin auto[15]	19	1	-	Covered
Coverpoint count out cv 1	100.00%	100	-	Covered
Coverpoint count_out_cv_1	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
bin auto[0]	60	1	-	Covered
bin auto[1]	44	1	-	Covered
bin auto[2]	20	1	-	Covered
bin auto[3]	23	1	-	Covered
bin auto[4]	23	1	-	Covered
bin auto[5]	29	1	-	Covered
bin auto[6]	23	1	-	Covered
bin auto[7]	30	1	-	Covered
bin auto[8]	25	1	-	Covered
bin auto[9]	26	1	-	Covered
bin auto[10]	35	1	-	Covered
bin auto[11]	22	1	-	Covered
bin auto[12]	28	1	-	Covered
bin auto[13]	33	1	-	Covered
bin auto[14]	24	1	-	Covered
bin auto[15]	27	1	-	Covered
Coverpoint count_out_cv_2	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin overflow1	6	1	-	Covered

Coverpoint count_out_cv_3	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
bin auto[0]	27	1	-	Covered
bin auto[1]	18	1	-	Covered
bin auto[2]	14	1	-	Covered
bin auto[3]	1	1	-	Covered
bin auto[4]	12	1	-	Covered
bin auto[5]	12	1	-	Covered
bin auto[6]	7	1	-	Covered
bin auto[7]	6	1	-	Covered
bin auto[8]	11	1	-	Covered
bin auto[9]	7	1	-	Covered
bin auto[10]	13	1	-	Covered
bin auto[11]	13	1	-	Covered
bin auto[12]	12	1	-	Covered
bin auto[13]	11	1	-	Covered
bin auto[14]	11	1	-	Covered
bin auto[15]	22	1	-	Covered
Coverpoint count_out_cv_4	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin overflow2	4	1	-	Covered
Coverpoint count_out_cv_5	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_count	1	1	-	Covered
Coverpoint count_out_cv_6	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_down	1	1	-	Covered
Coverpoint count_out_cv_4	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin overflow2	4	1	-	Covered
Coverpoint count_out_cv_5	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_count	1	1	-	Covered
Coverpoint count_out_cv_6	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin full_down	1	1	-	Covered
Coverpoint max_count_cv	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	959	1	-	Covered
bin auto[1]	77	1	-	Covered
Coverpoint zero_cv	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	910	1	-	Covered
bin auto[1]	126	1	-	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

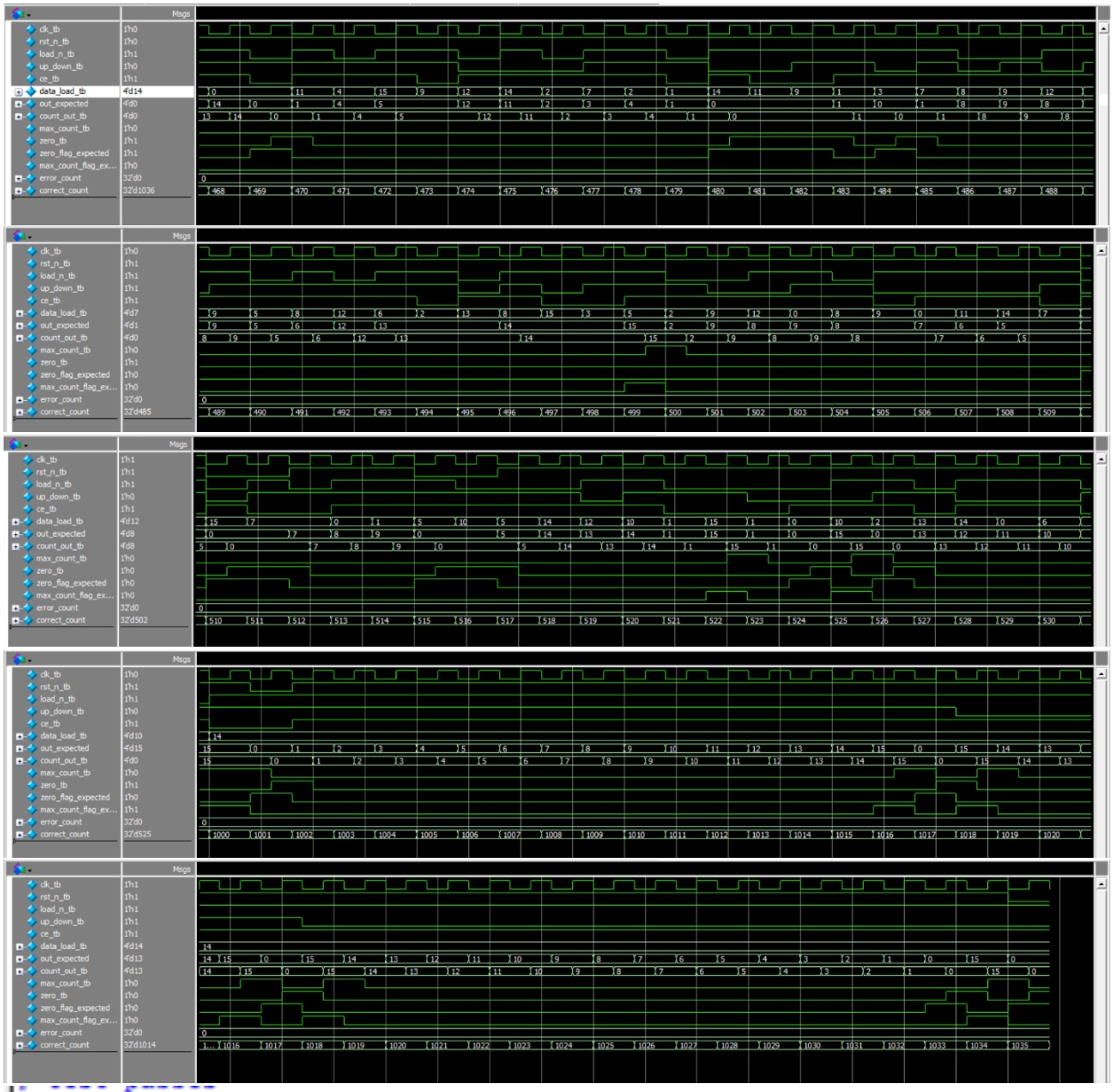
7. Functional Coverage report snippets

```
# ================================================================================
# === Instance: /\counter_tb#DUT
# === Design Unit: work.counter
# ================================================================================
# 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 counter.v
# ------------------------------------IF Branch------------------------------------
#     21                                     1036     Count coming in to IF
#     21              1                        51         if (!rst_n)
#
#     23              1                       298         else if (!load_n)
#
#     25              1                       495         else if (ce)
#
#                                             192     All False Count
# Branch totals: 4 hits of 4 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     26                                      495     Count coming in to IF
#     26              1                       337             if (up_down)
#
#     28              1                       158             else
#
# Branch totals: 2 hits of 2 branches = 100.00%
```

```
#
# ------------------------------------IF Branch------------------------------------
#     26                                      495     Count coming in to IF
#     26              1                       337             if (up_down)
#
#     28              1                       158             else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     32                                      819     Count coming in to IF
#     32              1                        58     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
#
#     32              2                       761     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     33                                      819     Count coming in to IF
#     33              1                        95     assign zero = (count_out == 0)? 1:0;
#
#     33              2                       724     assign zero = (count_out == 0)? 1:0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
#
# Condition Coverage:
#     Enabled Coverage             Bins   Covered    Misses  Coverage
#     ----------------             ----      ----    ------  --------
#     Conditions                      2         2         0   100.00%
```

```
#     Enabled Coverage              Bins    Covered    Misses  Coverage
#     ----------------              ----       ----    ------  --------
#     Conditions                       2          2         0   100.00%
#
# ================================Condition Details================================
#
# Condition Coverage for instance /\counter_tb#DUT  --
#
#   File counter.v
# ----------------Focused Condition View-------------------
# Line       32 Item    1  (count_out == {4{{1}}})
# Condition totals: 1 of 1 input term covered = 100.00%
#
#               Input Term   Covered  Reason for no coverage   Hint
#              -----------  --------  -----------------------  --------------
#   (count_out == {4{{1}}})         Y
#
#      Rows:       Hits  FEC Target                Non-masking condition(s)
#  ---------  ---------  --------------------      -------------------------
#   Row   1:          1  (count_out == {4{{1}}})_0  -
#   Row   2:          1  (count_out == {4{{1}}})_1  -
#
# ----------------Focused Condition View-------------------
# Line       33 Item    1  (count_out == 0)
# Condition totals: 1 of 1 input term covered = 100.00%
#
#         Input Term   Covered  Reason for no coverage   Hint
#        -----------  --------  -----------------------  --------------
#   (count_out == 0)         Y
#
#      Rows:       Hits  FEC Target            Non-masking condition(s)
#  ---------  ---------  --------------------  -------------------------
#   Row   1:          1  (count_out == 0)_0    -
#   Row   2:          1  (count_out == 0)_1    -
#
#
# 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)
#
#
# Total Coverage By Instance (filtered view): 100.00%
#
# End time: 20:40:10 on Sep 09,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

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




```
# test passes
# test passes
# test passes
# test passes
# test passes
# test passes
# correct counter =        1036 , error counter =           0
# ** Note: $stop    : TESTBENCH.svh(78)
#    Time: 4144 ns  Iteration: 1  Instance: /counter_tb
# Break in Module counter_tb at TESTBENCH.svh line 78
```

P3)

1. Testbench code

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

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

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

    ALSU_class obj1 = new ;


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


    integer error_count , correct_count ;

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

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

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

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

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

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

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

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

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

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

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

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

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

        assert_reset () ;

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

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

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

    endtask

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

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

2. Package code

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

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

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

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

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

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

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

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


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

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

        covergroup covcode @(posedge clk_cv);
            A_cp : coverpoint A {
                bins A_data_0             =    {  0  }    ;
                bins A_data_max           =    {MAXPOS}   ;
                bins A_data_min           =    {MAXNEG}   ;
                bins A_data_default       =    default    ;
                bins A_data_walkingones[] = {001,010,100} iff (red_op_A) ;
            }
            B_cp : coverpoint A {
                bins B_data_0             =    {  0  }    ;
                bins B_data_max           =    {MAXPOS}   ;
                bins B_data_min           =    {MAXNEG}   ;
                bins B_data_default       =    default    ;
                bins B_data_walkingones[] = {001,010,100} iff (!red_op_A && red_op_B) ;
            }
            ALU_CP : coverpoint opcode {
                bins Bins_shift[]    = {SHIFT , ROTATE} ;
                bins Bins_arith[]    = { ADD  ,  MULT } ;
                bins Bins_bitwise[]  = {OR , XOR} ;
                bins Bins_invalid    = {INVALID_6 , INVALID_7} ;
                bins trans  = (OR => XOR => ADD => MULT => SHIFT => ROTATE) ;
            }
            ALU_TRANS : coverpoint opcode {
                bins trans  = (OR => XOR => ADD => MULT => SHIFT => ROTATE => INVALID_6 => INVALID_7) ;
            }
        endgroup

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

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

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

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

    endclass
endpackage
```

3. Design code

```
module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction,
leds, out);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
input clk, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input signed [1:0] cin ;
input [2:0] opcode;
input signed [2:0] A, B;
output reg [15:0] leds;
output reg signed [5:0] out;

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

wire invalid_red_op , invalid;

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

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

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

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


        endcase
    end
  end
end

endmodule
```

4. Snippet to your verification plan document

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

5. Do file

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

6. Code Coverage report snippets

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

----------------------------------------------------------------------------------------------------------------
 TYPE /ALSU_package/ALSU_class/covcode                   100.00%        100          -    Covered
    covered/total bins:                                       17         17          -
    missing/total bins:                                        0         17          -
    % Hit:                                                100.00%        100          -
    Coverpoint A_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
    Coverpoint B_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
    Coverpoint ALU_CP                                     100.00%        100          -    Covered
        covered/total bins:                                    8          8          -
        missing/total bins:                                    0          8          -
        % Hit:                                            100.00%        100          -
    Coverpoint ALU_TRANS                                  100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
 Covergroup instance \/ALSU_package::ALSU_class::covcode
                                                          100.00%        100          -    Covered
    covered/total bins:                                       17         17          -
    missing/total bins:                                        0         17          -
    % Hit:                                                100.00%        100          -
    Coverpoint A_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
        bin A_data_0                                        23686          1          -    Covered
        bin A_data_max                                      20774          1          -    Covered
        bin A_data_min                                      21810          1          -    Covered
        bin A_data_walkingones[1]                             666          1          -    Covered
        default bin A_data_default                          74979                     -    Occurred
```

```
    Coverpoint B_cp                                       100.00%        100          -    Covered
        covered/total bins:                                    4          4          -
        missing/total bins:                                    0          4          -
        % Hit:                                            100.00%        100          -
        bin B_data_0                                        23686          1          -    Covered
        bin B_data_max                                      20774          1          -    Covered
        bin B_data_min                                      21810          1          -    Covered
        bin B_data_walkingones[1]                             126          1          -    Covered
        default bin B_data_default                          74979                     -    Occurred
    Coverpoint ALU_CP                                     100.00%        100          -    Covered
        covered/total bins:                                    8          8          -
        missing/total bins:                                    0          8          -
        % Hit:                                            100.00%        100          -
        bin Bins_shift[SHIFT]                               21309          1          -    Covered
        bin Bins_shift[ROTATE]                              21734          1          -    Covered
        bin Bins_arith[ADD]                                 21705          1          -    Covered
        bin Bins_arith[MULT]                                20983          1          -    Covered
        bin Bins_bitwise[OR]                                21609          1          -    Covered
        bin Bins_bitwise[XOR]                               21567          1          -    Covered
        bin Bins_invalid                                    31105          1          -    Covered
        bin trans                                               1          1          -    Covered
    Coverpoint ALU_TRANS                                  100.00%        100          -    Covered
        covered/total bins:                                    1          1          -
        missing/total bins:                                    0          1          -
        % Hit:                                            100.00%        100          -
        bin trans                                               1          1          -    Covered

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

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

7. Functional Coverage report snippets

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

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

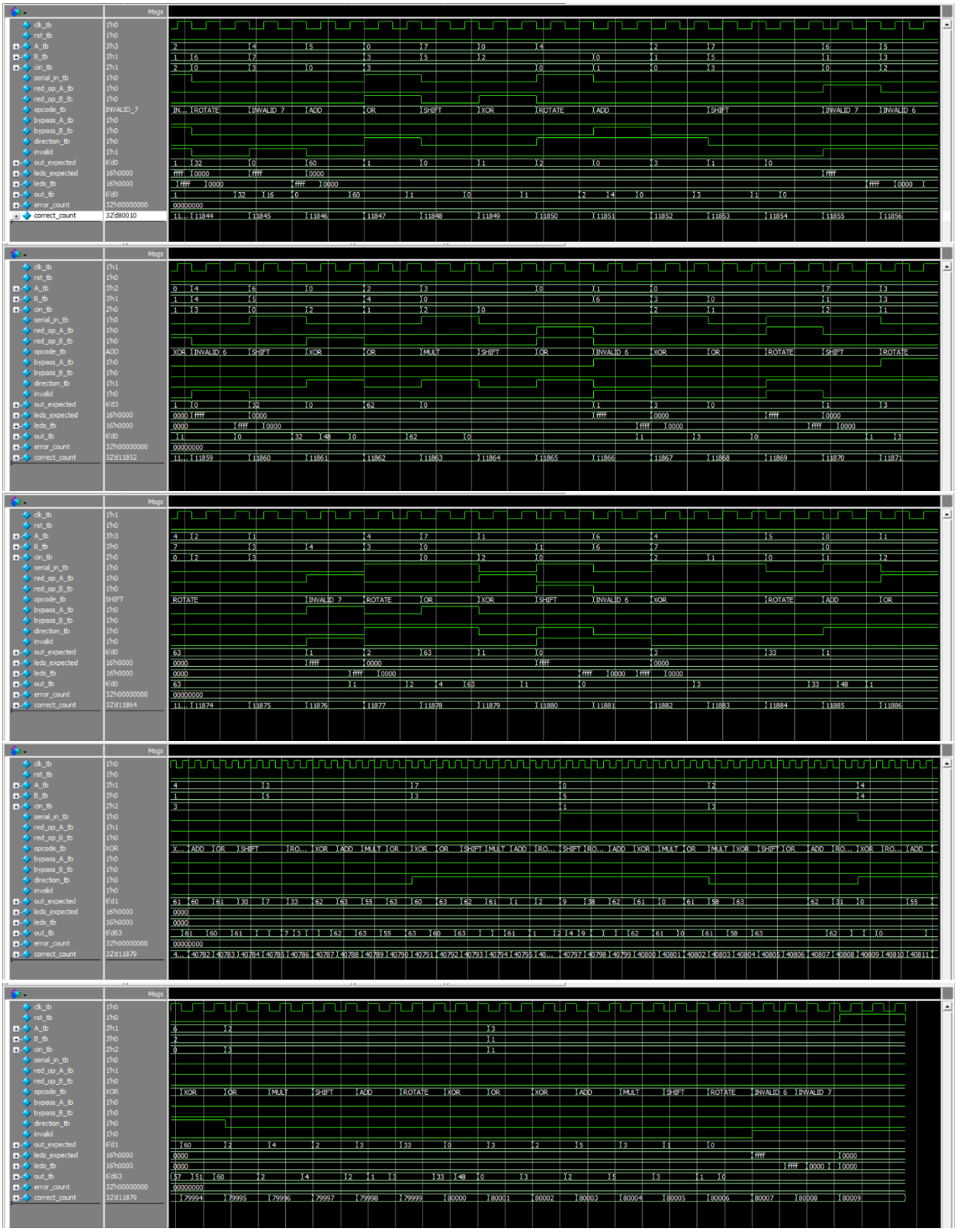
```
#
# ------------------------------------IF Branch------------------------------------
#     89                                    23920     Count coming in to IF
#     89              1                        52                 if (red_op_A_reg && red_op_B_reg)
#
#     91              1         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

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

```
# Statement Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Statements                      48        48         0   100.00%
#
# ================================Statement Details================================
#
# Statement Coverage for instance /\ALSU_tb#DUT  --
#
#     Line         Item                     Count     Source
#     ----         ----                     -----     ------
#   File ALSU.v
#     1                                               module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, led
#
#     2                                               parameter INPUT_PRIORITY = "A";
#
#     3                                               parameter FULL_ADDER = "ON";
#
#     4                                               input clk, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
#
#     5                                               input signed [1:0] cin ;
#
#     6                                               input [2:0] opcode;
#
#     7                                               input signed [2:0] A, B;
#
#     8                                               output reg [15:0] leds;
#
#     9                                               output reg signed [5:0] out;
#
#     10
#
#     11                                              reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
#
#     12                                              reg signed [1:0] cin_reg;
#
#     13                                              reg [2:0] opcode_reg;
#
#     14                                              reg signed [2:0] A_reg, B_reg;
#
#     15
#
#     16                                              wire invalid_red_op , invalid;
```

```
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Toggles                        122       122         0   100.00%
#
# ================================Toggle Details================================
#
# Toggle Coverage for instance /\ALSU_tb#DUT  --
#
#                                          Node      1H->0L      0L->1H                          "Coverage"
#                                          ----------------------------------------------------------------
#                                        A[0-2]           1           1                              100.00
#                                    A_reg[2-0]           1           1                              100.00
#                                        B[0-2]           1           1                              100.00
#                                    B_reg[2-0]           1           1                              100.00
#                                      bypass_A           1           1                              100.00
#                                  bypass_A_reg           1           1                              100.00
#                                      bypass_B           1           1                              100.00
#                                  bypass_B_reg           1           1                              100.00
#                                       cin[0-1]           1           1                              100.00
#                                   cin_reg[1-0]           1           1                              100.00
#                                            clk           1           1                              100.00
#                                      direction           1           1                              100.00
#                                  direction_reg           1           1                              100.00
#                                        invalid           1           1                              100.00
#                                 invalid_opcode           1           1                              100.00
#                                 invalid_red_op           1           1                              100.00
#                                     leds[15-0]           1           1                              100.00
#                                    opcode[0-2]           1           1                              100.00
#                                opcode_reg[2-0]           1           1                              100.00
#                                       out[5-0]           1           1                              100.00
#                                      red_op_A           1           1                              100.00
#                                  red_op_A_reg           1           1                              100.00
#                                      red_op_B           1           1                              100.00
#                                  red_op_B_reg           1           1                              100.00
#                                           rst           1           1                              100.00
#                                     serial_in           1           1                              100.00
#                                 serial_in_reg           1           1                              100.00
#
# Total Node Count     =         61
# Toggled Node Count   =         61
# Untoggled Node Count =          0
#
# Toggle Coverage      =     100.00% (122 of 122 bins)
#
```

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



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

P4)

1. Testbench code

```
module memory_tb();
    localparam TESTS = 100 ;
    reg [16] address_array [] ;
    reg [8] data_to_write_array [] ;
    reg [8] data_read_expect_assoc [int] ;
    reg [9] data_read_queue [$] ;
    reg clk_tb ;
    reg write_tb ;
    reg read_tb ;
    reg [8] DATA_in_tb ;
    reg [16] address_tb ;
    wire [9] DATA_out_tb ;
    reg [9] read_data_stored ;

    reg [9] expected_data_out ;

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

    my_mem DUT (clk_tb , write_tb , read_tb , DATA_in_tb , address_tb , DATA_out_tb ) ;

    int correct_count , error_count ;

    initial begin
        stimuulus_gen () ;
        golden_model  () ;
        write_tb = 1 ;
        read_tb  = 0 ;
        for (int i = 0; i < TESTS; i++) begin
            address_tb = address_array [i] ;
            DATA_in_tb = data_to_write_array [i] ;
            @(negedge clk_tb);
        end
        write_tb = 0 ;
        read_tb  = 1 ;
        for (int i = 0; i < TESTS; i++) begin
            address_tb = address_array [i] ;
            check9Bits (data_read_expect_assoc[address_array[i]]) ;
            print_test_result () ;
            data_read_queue.push_back (DATA_out_tb) ;
        end
        $display("read data stored in the queue");
        for (int i = 0; i < TESTS; i++) begin
            read_data_stored = data_read_queue.pop_front() ;
            check9Bits (data_read_expect_assoc[address_array[i]]) ;
            $display("read data with index %d is %b" , i , read_data_stored);
            $display("read data expected with index %d is %b" , i , expected_data_out);
        end

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

    task stimuulus_gen ();
        address_array       = new[TESTS];
        data_to_write_array = new[TESTS];
        for (int i = 0; i < TESTS; i++) begin
            address_array[i]       = $random;
            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] ;
        end
    endtask

    task check9Bits (reg [8] data_written );
        expected_data_out = {~^data_written , data_written} ;
    endtask

    task print_test_result();
         @(negedge clk_tb);
        if (DATA_out_tb == expected_data_out) begin
            $display ("test passes") ;
            correct_count = correct_count + 1 ;
        end
        else begin
            $display ("test fail ") ;
            error_count = error_count + 1 ;
        end
    endtask
endmodule
```

2. Design code

```
module my_mem(
    input clk,
    input write,
    input read,
    input [7:0] data_in,
    input [15:0] address,
    output reg [8:0] data_out
    );

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

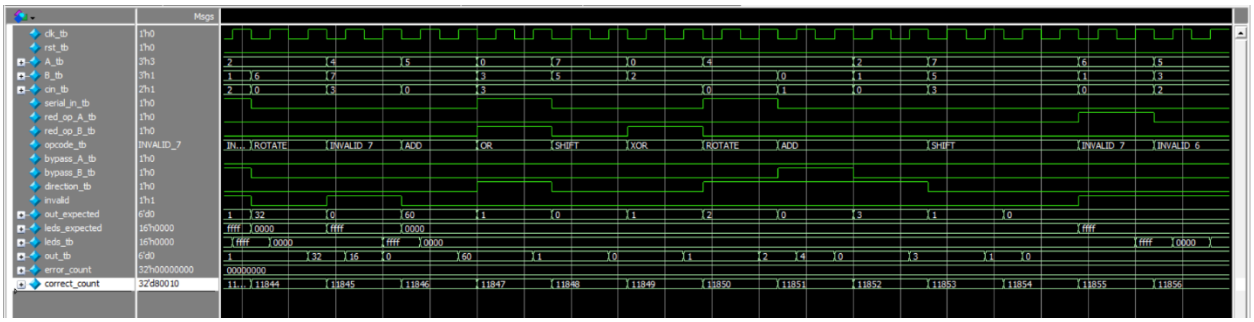
3. Report any bugs detected in the design and fix them
   - Change the width of **DATA_out** from **9 bits** to **8 bits**
4. Snippet to your verification plan document

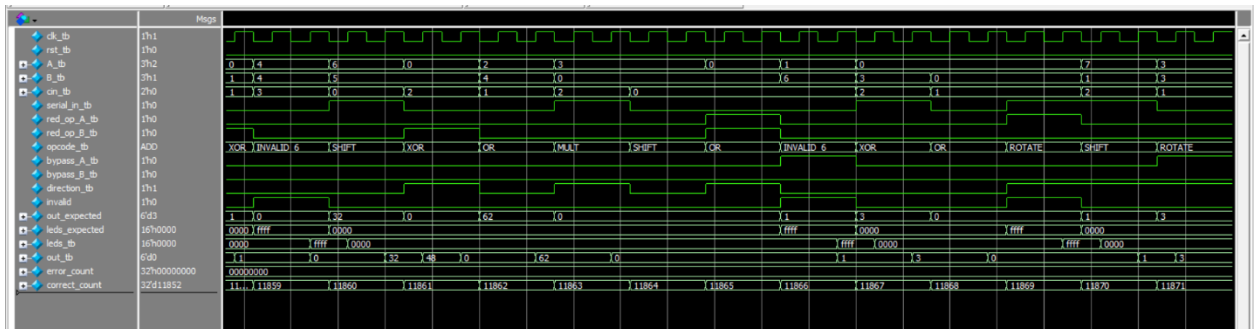
5. Do file

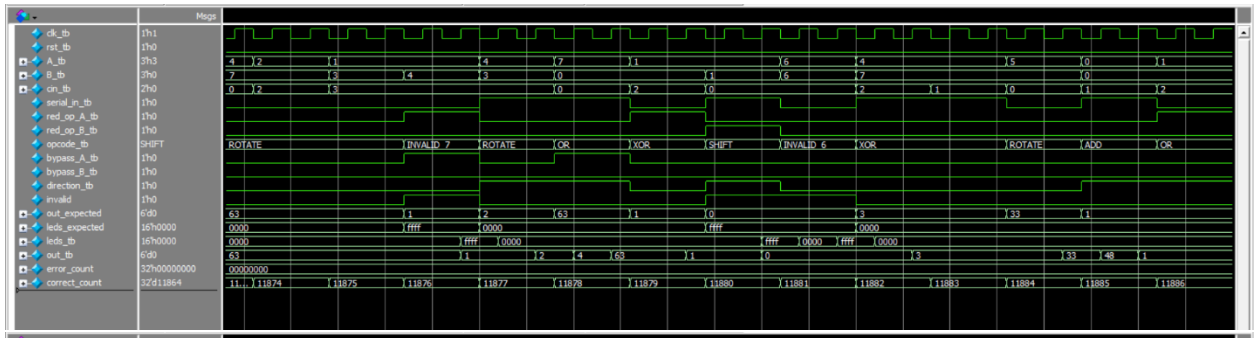
```
vlib work
vlog DESIGN.svh TESTBENCH.svh +cover -covercells
vsim -voptargs=+acc work.memory_tb -cover
add wave -position insertpoint  \
sim:/memory_tb/TESTS \
sim:/memory_tb/clk_tb \
sim:/memory_tb/write_tb \
sim:/memory_tb/read_tb \
sim:/memory_tb/DATA_in_tb \
sim:/memory_tb/address_tb \
sim:/memory_tb/DATA_out_tb \
sim:/memory_tb/read_data_stored \
sim:/memory_tb/expected_data_out \
sim:/memory_tb/correct_count \
sim:/memory_tb/error_count
coverage save my_mem.ucdb -onexit -du ALSU
run -all
```

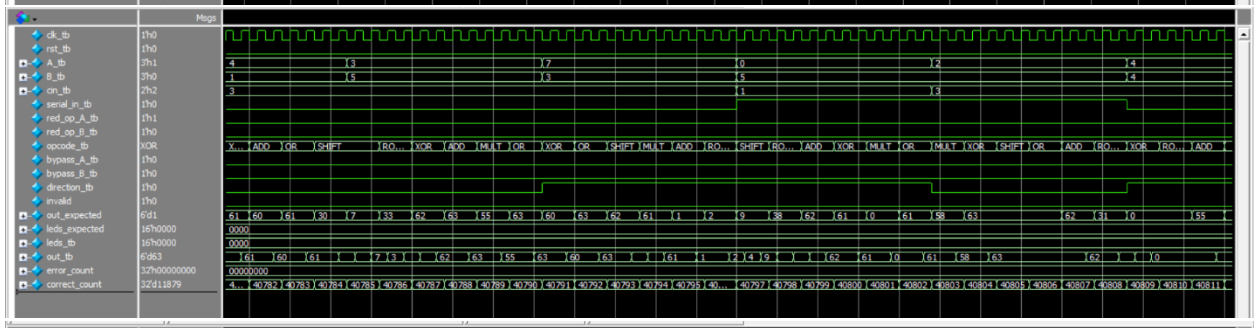
Associated array , **queue** and **dynamic array** cannot added to waveform so I exclude them in do file

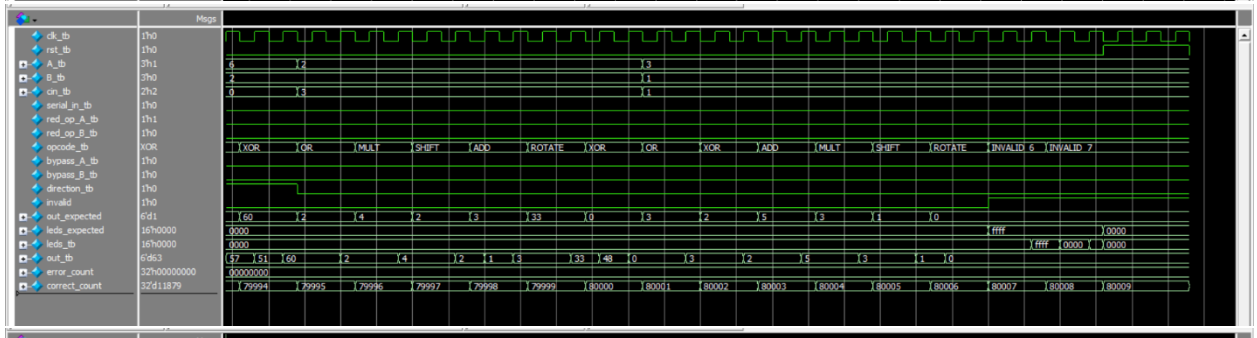
6. Clear and neat QuestaSim waveform snippets showing the functionality of the design

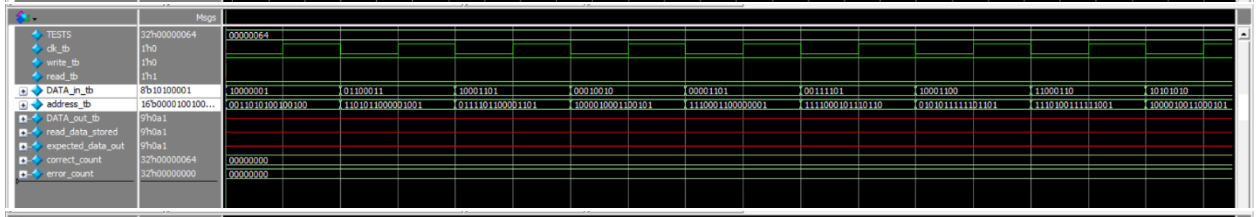


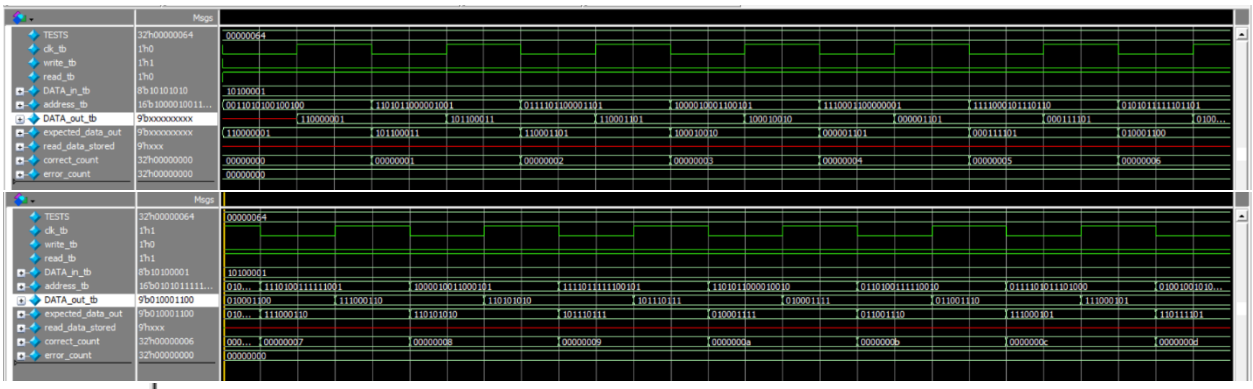













```
# read data stored in the queue
# read data with index           0 is 110000001
# read data expected with index           0 is 110000001
# read data with index           1 is 101100011
# read data expected with index           1 is 101100011
# read data with index           2 is 110001101
# read data expected with index           2 is 110001101
# read data with index           3 is 100010010
# read data expected with index           3 is 100010010
# read data with index           4 is 000001101
# read data expected with index           4 is 000001101
# read data with index           5 is 000111101
# read data expected with index           5 is 000111101
# read data with index           6 is 010001100
# read data expected with index           6 is 010001100
# read data with index           7 is 111000110
# read data expected with index           7 is 111000110
# read data with index           8 is 110101010
# read data expected with index           8 is 110101010
# read data with index           9 is 101110111
# read data expected with index           9 is 101110111
# read data with index          10 is 010001111
# read data expected with index          10 is 010001111
# read data with index          11 is 011001110
# read data expected with index          11 is 011001110
# read data with index          12 is 111000101
# read data expected with index          12 is 111000101
# read data with index          13 is 110111101
# read data expected with index          13 is 110111101
# read data with index          14 is 101100101
# read data expected with index          14 is 101100101
# read data with index          15 is 100001010
# read data expected with index          15 is 100001010
# read data with index          16 is 000100000
# read data expected with index          16 is 000100000
# read data with index          17 is 010011101
# read data expected with index          17 is 010011101
# read data with index          18 is 000010011
# read data expected with index          18 is 000010011
# read data with index          19 is 101010011
# read data expected with index          19 is 101010011
# read data with index          20 is 011010101
# read data expected with index          20 is 011010101
# read data with index          21 is 010101110
# read data expected with index          21 is 010101110
```

```
# read data with index          22 is 111001111
# read data expected with index          22 is 111001111
# read data with index          23 is 100001010
# read data expected with index          23 is 100001010
# read data with index          24 is 100111100
# read data expected with index          24 is 100111100
# read data with index          25 is 010001010
# read data expected with index          25 is 010001010
# read data with index          26 is 111011000
# read data expected with index          26 is 111011000
# read data with index          27 is 010001001
# read data expected with index          27 is 010001001
# read data with index          28 is 010110110
# read data expected with index          28 is 010110110
# read data with index          29 is 010101110
# read data expected with index          29 is 010101110
# read data with index          30 is 000101010
# read data expected with index          30 is 000101010
# read data with index          31 is 101110001
# read data expected with index          31 is 101110001
# read data with index          32 is 001001111
# read data expected with index          32 is 001001111
# read data with index          33 is 100111010
# read data expected with index          33 is 100111010
# read data with index          34 is 000010101
# read data expected with index          34 is 000010101
# read data with index          35 is 011011001
# read data expected with index          35 is 011011001
# read data with index          36 is 001001100
# read data expected with index          36 is 001001100
# read data with index          37 is 010001111
# read data expected with index          37 is 010001111
# read data with index          38 is 110110111
# read data expected with index          38 is 110110111
# read data with index          39 is 101011100
# read data expected with index          39 is 101011100
# read data with index          40 is 010001001
# read data expected with index          40 is 010001001
# read data with index          41 is 011010000
# read data expected with index          41 is 011010000
# read data with index          42 is 001010001
# read data expected with index          42 is 001010001
# read data with index          43 is 100001100
# read data expected with index          43 is 100001100
```

```
# read data with index          44 is 011001000
# read data expected with index          44 is 011001000
# read data with index          45 is 000111101
# read data expected with index          45 is 000111101
# read data with index          46 is 101111110
# read data expected with index          46 is 101111110
# read data with index          47 is 100111001
# read data expected with index          47 is 100111001
# read data with index          48 is 011010011
# read data expected with index          48 is 011010011
# read data with index          49 is 101111000
# read data expected with index          49 is 101111000
# read data with index          50 is 001001001
# read data expected with index          50 is 001001001
# read data with index          51 is 000101010
# read data expected with index          51 is 000101010
# read data with index          52 is 010000110
# read data expected with index          52 is 010000110
# read data with index          53 is 110011100
# read data expected with index          53 is 110011100
# read data with index          54 is 000100110
# read data expected with index          54 is 000100110
# read data with index          55 is 110100011
# read data expected with index          55 is 110100011
# read data with index          56 is 010110011
# read data expected with index          56 is 010110011
# read data with index          57 is 101000100
# read data expected with index          57 is 101000100
# read data with index          58 is 011001011
# read data expected with index          58 is 011001011
# read data with index          59 is 101011010
# read data expected with index          59 is 101011010
# read data with index          60 is 111101101
# read data expected with index          60 is 111101101
# read data with index          61 is 101100101
# read data expected with index          61 is 101100101
# read data with index          62 is 011011111
# read data expected with index          62 is 011011111
# read data with index          63 is 101000100
# read data expected with index          63 is 101000100
# read data with index          64 is 000101010
# read data expected with index          64 is 000101010
# read data with index          65 is 000001110
# read data expected with index          65 is 000001110
# read data with index          66 is 110011010
# read data expected with index          66 is 110011010
```

```
# read data with index          67 is 111000011
# read data expected with index          67 is 111000011
# read data with index          68 is 101001110
# read data expected with index          68 is 101001110
# read data with index          69 is 100001010
# read data expected with index          69 is 100001010
# read data with index          70 is 000111000
# read data expected with index          70 is 000111000
# read data with index          71 is 110111000
# read data expected with index          71 is 110111000
# read data with index          72 is 110010011
# read data expected with index          72 is 110010011
# read data with index          73 is 101011001
# read data expected with index          73 is 101011001
# read data with index          74 is 101001101
# read data expected with index          74 is 101001101
# read data with index          75 is 001101101
# read data expected with index          75 is 001101101
# read data with index          76 is 111001010
# read data expected with index          76 is 111001010
# read data with index          77 is 110010101
# read data expected with index          77 is 110010101
# read data with index          78 is 000000100
# read data expected with index          78 is 000000100
# read data with index          79 is 101101001
# read data expected with index          79 is 101101001
# read data with index          80 is 110001000
# read data expected with index          80 is 110001000
# read data with index          81 is 100101101
# read data expected with index          81 is 100101101
# read data with index          82 is 100101110
# read data expected with index          82 is 100101110
# read data with index          83 is 000011100
# read data expected with index          83 is 000011100
# read data with index          84 is 000101001
# read data expected with index          84 is 000101001
# read data with index          85 is 010000110
# read data expected with index          85 is 010000110
# read data with index          86 is 000111101
# read data expected with index          86 is 000111101
# read data with index          87 is 001110000
# read data expected with index          87 is 001110000
# read data with index          88 is 010111010
# read data expected with index          88 is 010111010
# read data with index          89 is 111111010
# read data expected with index          89 is 111111010
```

```
# read data expected with index          89 is 111111010
# read data with index          90 is 000011010
# read data expected with index          90 is 000011010
# read data with index          91 is 000110111
# read data expected with index          91 is 000110111
# read data with index          92 is 111000000
# read data expected with index          92 is 111000000
# read data with index          93 is 010110110
# read data expected with index          93 is 010110110
# read data with index          94 is 011011100
# read data expected with index          94 is 011011100
# read data with index          95 is 101111000
# read data expected with index          95 is 101111000
# read data with index          96 is 111011011
# read data expected with index          96 is 111011011
# read data with index          97 is 001111001
# read data expected with index          97 is 001111001
# read data with index          98 is 001100001
# read data expected with index          98 is 001100001
# read data with index          99 is 010100001
# read data expected with index          99 is 010100001
# correct counter =         100 , error counter =           0
# ** Note: $stop    : TESTBENCH.svh(55)
#    Time: 800 ns  Iteration: 1  Instance: /memory_tb
# Break in Module memory_tb at TESTBENCH.svh line 55
```