Assignment 3

Digital Design Verification

Contents

1 Q1: ALU_seq

1.1 1. Testbench code

```
//=====================================================
// Testbench with Golden Model and Reset Task
//=====================================================
import alu_pkg::*;  // Import the package

module alu_tb;

  // Declare testbench signals that drive the DUT.
  byte operand1, operand2, dut_out;
  bit clk, rst;
  opcode_e opcode;

  // DUT instantiation.
  alu_seq dut (
    .operand1(operand1),
    .operand2(operand2),
    .clk(clk),
    .rst(rst),
    .opcode(opcode),
    .out(dut_out)
  );

  // Clock generation: 10 time-unit period.
  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end

  //-------------------------------------------------------------
  // Reset Task: Applies reset for two clock cycles.
  //-------------------------------------------------------------
  task reset_task();
    begin
      rst = 1;
      repeat (2) @(posedge clk);
      rst = 0;
    end
  endtask

  //-------------------------------------------------------------
  // Golden Model Function: Computes the expected output.
  //-------------------------------------------------------------
  function byte golden_model(byte a, byte b, opcode_e op);
    byte result;
    begin
      case(op)
        ADD:  result = a + b;
        SUB:  result = a - b;
        MULT: result = a * b;
        DIV:  result = a / b;
      endcase
      return result;
    end
  endfunction

  //-------------------------------------------------------------
  // Instantiate the stimulus/coverage class.
  //-------------------------------------------------------------
  alu_cfg cfg;

  // Error counter.
  int error_count = 0;
  byte expected;

  // Main testbench stimulus.
  initial begin
    // Create the stimulus object.
    cfg = new();
    // Tie the class clock to the testbench clock.
    cfg.clk = clk;

    // Apply reset.
    reset_task();

    // Repeat randomizations (for example, 100 iterations).
    repeat (500) begin
      // Randomize the stimulus class.
      if (!cfg.randomize()) begin
         $display("Randomization failed!");
         continue;
      end

      @(negedge clk);
      // Assign the randomized values to the DUT inputs.
      operand1 = cfg.operand1;
      operand2 = cfg.operand2;
      opcode   = cfg.opcode;

      // Sample coverage within the stimulus object.
      cfg.post_randomize();

      // Allow one clock edge for DUT to latch the new inputs.
      @(posedge clk);

      // Compute expected result using the golden model.
      expected = golden_model(operand1, operand2, opcode);
```

```
97
98        // Wait for DUT to update output.
99        @(posedge clk);
100
101       // Compare DUT output with the golden model.
102       if (dut_out !== expected) begin
103          $display("ERROR␣at␣time␣%0t:␣operand1=%0d,␣operand2=%0d,␣opcode=%s,␣DUT␣out=%0d,␣expected=%0d",
104                   $time, operand1, operand2, opcode.name(), dut_out, expected);
105          error_count++;
106       end else begin
107          $display("PASS␣at␣time␣%0t:␣operand1=%0d,␣operand2=%0d,␣opcode=%s,␣out=%0d",
108                   $time, operand1, operand2, opcode.name(), dut_out);
109       end
110     end
111
112     $display("Test␣completed␣with␣%0d␣errors", error_count);
113
114     @(posedge clk);
115     rst = 1;
116     #20;
117
118     $finish;
119   end
120
121 endmodule
```

1.2 2. Package code

```
1  //=================================================
2  // Package containing stimulus class and coverage
3  //=================================================
4  package alu_pkg;
5
6    // Define an opcode enumeration for the ALU operations.
7    typedef enum logic [1:0] { ADD, SUB, MULT, DIV } opcode_e;
8
9    //-------------------------------------------------------------------
10   // Class to randomize ALU inputs and capture functional coverage
11   //-------------------------------------------------------------------
12   class alu_cfg;
13     // Randomizable inputs:
14     rand byte operand1;
15     rand byte operand2;
16     rand opcode_e opcode;
17
18     // A clock signal used for covergroup sampling. This must be
19     // connected from the testbench.
20     bit clk;
21
22     // Functional coverage group to cover all randomized inputs.
23     covergroup cg_inputs;
24       // Coverpoints for each input:
25       coverpoint operand1;
26       coverpoint operand2;
27       coverpoint opcode;
28     endgroup
29
30     // Constructor: instantiate the covergroup.
31     function new();
32       cg_inputs = new();
33     endfunction
34
35     // This method is called after randomization to sample the covergroup.
36     function void post_randomize();
37       cg_inputs.sample();
38     endfunction
39
40     //=================================================
41     // Constraint to enforce corner cases for operands and
42     // ensure all ALU operations occur.
43     //=================================================
44     constraint operand_corner {
45       // Force operand1 to be 0 or max (8'hFF) in 20% of cases
46       // and any value in between in 30% of cases.
47       operand1 dist {
48         0      :/ 10,
49         -128   :/ 10,
50         [-127:-1] :/ 40,
51         [1:127] :/ 40
52       };
53
54       // Similarly for operand2.
55       operand2 dist {
56         0      :/ 10,
57         -128   :/ 10,
58         [-127:-1] :/ 40,
59         [1:127] :/ 40
60       };
61     }
62
63     constraint opcode_dist {
64       // Ensure that each ALU operation (ADD, SUB, MULT, DIV) occurs equally often.
65       opcode dist { ADD:= 25, SUB:= 25, MULT:= 25, DIV:= 25 };
66     }
67
68   endclass : alu_cfg
69
70 endpackage
```

1.3 3. Design code

```
//=====================================================
// DUT: ALU Sequence Module
//=====================================================
module alu_seq(
  input  byte    operand1,
  input  byte    operand2,
  input          clk,
  input          rst,
  input  alu_pkg::opcode_e opcode,
  output byte    out
);
  // On every positive edge of the clock, process reset or ALU operation.
  always @(posedge clk) begin
    if (rst)
      out <= 0;
    else
      case (opcode)
        alu_pkg::ADD:  out <= operand1 + operand2;
        alu_pkg::SUB:  out <= operand1 - operand2;
        alu_pkg::MULT: out <= operand1 * operand2;
        alu_pkg::DIV:  out <= operand1 / operand2;
        default:       out <= 0;
      endcase
  end
endmodule
```

1.4 4. Bug report

there is a bug i didn't fix this alu does not have the logic when operand2 = 0 and division that not valid but it is give zero as de

1.5 5. Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
ALU_1	When reset is asserted, the output should be reset to 0.	- Directed reset at the start of simulation via reset_task(). - Reset applied for 2 clock cycles.	- Reset transitions are captured through testbench observations. - Reset application is controlled and deterministic.	- The testbench verifies that when rst is high, the output is driven to 0. - Error reporting if behavior deviates from expected.
ALU_2	When opcode is ADD, the ALU should perform addition on operands 1 and 2.	- Randomize opcode using opcode_dist constraint ensuring ADD appears 25% of the time. - Corner-case inputs for operand1 and operand2 under operand_corner constraints (favoring 0, -128, and range values).	- The covergroup tracks opcode and operands to ensure corner cases are exercised. - All possible ADD operations should be covered with emphasis on edge cases.	- After each operation, the testbench calls golden_model() to confirm dut_out == operand1 + operand2. - Errors are flagged and counted if output differs from expected value.
ALU_3	When opcode is SUB, the ALU should perform subtraction on operands 1 and 2.	- Randomize opcode using opcode_dist constraint ensuring SUB appears 25% of the time. - Corner-case inputs follow the operand_corner distribution.	- The covergroup tracks opcode and operands to ensure corner cases of subtraction are exercised. - Special attention to underflow cases.	- Golden model confirms dut_out == operand1 - operand2. - Error detection and reporting for mismatches.
ALU_4	When opcode is MULT, the ALU should perform multiplication on operands 1 and 2.	- Randomize opcode using opcode_dist constraint ensuring MULT appears 25% of the time. - Corner-case inputs follow the operand_corner distribution with special emphasis on 0 values.	- The covergroup tracks opcode and operands to ensure multiplication corner cases are exercised. - Special focus on multiplication by 0, -1, and maximum values.	- Golden model confirms dut_out == operand1 * operand2. - Error reporting for any discrepancies.
ALU_5	When opcode is DIV, the ALU should perform division on operands 1 and 2.	- Randomize opcode using opcode_dist constraint ensuring DIV appears 25% of the time. - Corner-case inputs follow the operand_corner distribution. - Special handling to avoid division by zero.	- The covergroup tracks opcode and operands to ensure division corner cases are exercised. - Special focus on division by 1, -1, and maximum values.	- Golden model confirms dut_out == operand1 / operand2. - Special handling for division by zero cases in both DUT and golden model for consistency.
ALU_6	The ALU should handle all possible combinations of corner cases for operands.	- Operand corner constraints force 0 and -128 values to appear 10% each. - Remaining 80% of values are distributed across negative and positive ranges.	- The covergroup tracks operand values to ensure all byte ranges are covered. - Special tracking of corner case combinations.	- For all operations with corner cases, the golden model provides expected results for comparison. - Special attention to overflow/underflow scenarios.
ALU_7	Operations should complete within one clock cycle.	- Various timing scenarios generated through controlled clock phasing. - Randomization across 500 test cases.	- Coverage of timing requirements through testbench monitoring. - Ensure operations complete in expected timeframe.	- Testbench checks that output is updated after one clock cycle following input changes. - Timing violations would be captured and reported.

Table 1: ALU Verification Plan

1.6 6. Do File

```
vlib work
vlog alu_pkg.sv alu_seq.sv alu_tb.sv +cover -covercells
vsim -voptargs=+acc work.alu_tb -cover
add wave *
coverage save alu_tb.ucdb -onexit
coverage exclude -src alu_seq.sv -line 22
run -all

# to run do file
#-- do run.txt
# to execute coverage report (one for code coverage and other fuctional coverage)
#-- vcover report alu_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=alu_seq
#-- vcover report -details -cvg -output alu_coverage_report.txt alu_tb.ucdb
```

1.7 7. functional Coverage Report

```
Coverage Report by DU with details

=================================================================================
=== Design Unit: work.alu_seq
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         6         6         0   100.00%

================================Branch Details================================
Branch Coverage for Design Unit work.alu_seq

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File alu_seq.sv
-----------------------------------IF Branch-----------------------------------
    14                                       989     Count coming in to IF
    14              1                          2         if (rst)
    16              1                        987         else
Branch totals: 2 hits of 2 branches = 100.00%

---------------------------------CASE Branch-----------------------------------
    17                                       987     Count coming in to CASE
    18              1                        242               alu_pkg::ADD:  out <= operand1 + operand2;
    19              1                        262               alu_pkg::SUB:  out <= operand1 - operand2;
    20              1                        236               alu_pkg::MULT: out <= operand1 * operand2;
    21              1                        247               alu_pkg::DIV:  out <= operand1 / operand2;
Branch totals: 4 hits of 4 branches = 100.00%


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

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

Statement Coverage for Design Unit work.alu_seq --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File alu_seq.sv
    4                                                module alu_seq(
    5                                                  input  byte     operand1,
    6                                                  input  byte     operand2,
    7                                                  input          clk,
    8                                                  input          rst,
    9                                                  input  alu_pkg::opcode_e opcode,
    10                                                 output byte     out
    11                                               );
    12                                                 // On every positive edge of the clock, process reset or ALU operation.
    13              1                        989       always @(posedge clk) begin
    14                                                   if (rst)
    15              1                          2           out <= 0;
    16                                                   else
    17                                                     case (opcode)
    18              1                        242             alu_pkg::ADD:  out <= operand1 + operand2;
    19              1                        262             alu_pkg::SUB:  out <= operand1 - operand2;
    20              1                        236             alu_pkg::MULT: out <= operand1 * operand2;
    21              1                        247             alu_pkg::DIV:  out <= operand1 / operand2;

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

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

Toggle Coverage for Design Unit work.alu_seq

                                    Node         1H->0L         0L->1H                                "Coverage"
                                    ----         ------         ------                                ----------
```

clk	1	1	100.00
opcode[0−1]	1	1	100.00
operand1[0−7]	1	1	100.00
operand2[0−7]	1	1	100.00
out[0−7]	1	1	100.00
rst	1	1	100.00

```
Total Node Count     =         28
Toggled Node Count   =         28
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (56 of 56 bins)


Total Coverage By Design Unit (filtered view): 100.00%
```

1.8 8. code Coverage Report

```
Coverage Report by instance with details

=================================================================================
=== Instance: /alu_pkg
=== Design Unit: work.alu_pkg
=================================================================================

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses     3        na        na        na
            Covergroup Bins   132       132         0   100.00%
```

Covergroup	Metric	Goal	Bins	Status
TYPE /alu_pkg/alu_cfg/cg_inputs	100.00%	100	–	Covered
covered/total bins:	132	132	–	
missing/total bins:	0	132	–	
% Hit:	100.00%	100	–	
Coverpoint operand1	100.00%	100	–	Covered
covered/total bins:	64	64	–	
missing/total bins:	0	64	–	
% Hit:	100.00%	100	–	
bin auto[−128:−125]	16	1	–	Covered
bin auto[−124:−121]	4	1	–	Covered
bin auto[−120:−117]	20	1	–	Covered
bin auto[−116:−113]	16	1	–	Covered
bin auto[−112:−109]	14	1	–	Covered
bin auto[−108:−105]	6	1	–	Covered
bin auto[−104:−101]	18	1	–	Covered
bin auto[−100:−97]	20	1	–	Covered
bin auto[−96:−93]	16	1	–	Covered
bin auto[−92:−89]	10	1	–	Covered
bin auto[−88:−85]	6	1	–	Covered
bin auto[−84:−81]	12	1	–	Covered
bin auto[−80:−77]	12	1	–	Covered
bin auto[−76:−73]	26	1	–	Covered
bin auto[−72:−69]	16	1	–	Covered
bin auto[−68:−65]	18	1	–	Covered
bin auto[−64:−61]	12	1	–	Covered
bin auto[−60:−57]	16	1	–	Covered
bin auto[−56:−53]	16	1	–	Covered
bin auto[−52:−49]	24	1	–	Covered
bin auto[−48:−45]	26	1	–	Covered
bin auto[−44:−41]	20	1	–	Covered
bin auto[−40:−37]	16	1	–	Covered
bin auto[−36:−33]	18	1	–	Covered
bin auto[−32:−29]	20	1	–	Covered
bin auto[−28:−25]	16	1	–	Covered
bin auto[−24:−21]	18	1	–	Covered
bin auto[−20:−17]	4	1	–	Covered
bin auto[−16:−13]	12	1	–	Covered
bin auto[−12:−9]	12	1	–	Covered
bin auto[−8:−5]	16	1	–	Covered
bin auto[−4:−1]	16	1	–	Covered
bin auto[0:3]	14	1	–	Covered
bin auto[4:7]	10	1	–	Covered
bin auto[8:11]	10	1	–	Covered
bin auto[12:15]	12	1	–	Covered
bin auto[16:19]	28	1	–	Covered
bin auto[20:23]	26	1	–	Covered
bin auto[24:27]	8	1	–	Covered
bin auto[28:31]	20	1	–	Covered
bin auto[32:35]	14	1	–	Covered
bin auto[36:39]	16	1	–	Covered
bin auto[40:43]	16	1	–	Covered
bin auto[44:47]	20	1	–	Covered
bin auto[48:51]	6	1	–	Covered
bin auto[52:55]	12	1	–	Covered
bin auto[56:59]	12	1	–	Covered
bin auto[60:63]	16	1	–	Covered
bin auto[64:67]	18	1	–	Covered
bin auto[68:71]	12	1	–	Covered

bin auto[72:75]	18	1	–	Covered
bin auto[76:79]	18	1	–	Covered
bin auto[80:83]	12	1	–	Covered
bin auto[84:87]	12	1	–	Covered
bin auto[88:91]	14	1	–	Covered
bin auto[92:95]	22	1	–	Covered
bin auto[96:99]	22	1	–	Covered
bin auto[100:103]	8	1	–	Covered
bin auto[104:107]	22	1	–	Covered
bin auto[108:111]	12	1	–	Covered
bin auto[112:115]	16	1	–	Covered
bin auto[116:119]	16	1	–	Covered
bin auto[120:123]	22	1	–	Covered
bin auto[124:127]	24	1	–	Covered
Coverpoint operand2	100.00%	100	–	Covered
covered/total bins:	64	64	–	
missing/total bins:	0	64	–	
% Hit:	100.00%	100	–	
bin auto[−128:−125]	6	1	–	Covered
bin auto[−124:−121]	16	1	–	Covered
bin auto[−120:−117]	18	1	–	Covered
bin auto[−116:−113]	18	1	–	Covered
bin auto[−112:−109]	6	1	–	Covered
bin auto[−108:−105]	12	1	–	Covered
bin auto[−104:−101]	26	1	–	Covered
bin auto[−100:−97]	10	1	–	Covered
bin auto[−96:−93]	14	1	–	Covered
bin auto[−92:−89]	26	1	–	Covered
bin auto[−88:−85]	8	1	–	Covered
bin auto[−84:−81]	18	1	–	Covered
bin auto[−80:−77]	10	1	–	Covered
bin auto[−76:−73]	28	1	–	Covered
bin auto[−72:−69]	18	1	–	Covered
bin auto[−68:−65]	4	1	–	Covered
bin auto[−64:−61]	8	1	–	Covered
bin auto[−60:−57]	12	1	–	Covered
bin auto[−56:−53]	18	1	–	Covered
bin auto[−52:−49]	16	1	–	Covered
bin auto[−48:−45]	14	1	–	Covered
bin auto[−44:−41]	10	1	–	Covered
bin auto[−40:−37]	22	1	–	Covered
bin auto[−36:−33]	20	1	–	Covered
bin auto[−32:−29]	14	1	–	Covered
bin auto[−28:−25]	18	1	–	Covered
bin auto[−24:−21]	24	1	–	Covered
bin auto[−20:−17]	10	1	–	Covered
bin auto[−16:−13]	20	1	–	Covered
bin auto[−12:−9]	22	1	–	Covered
bin auto[−8:−5]	16	1	–	Covered
bin auto[−4:−1]	14	1	–	Covered
bin auto[0:3]	10	1	–	Covered
bin auto[4:7]	14	1	–	Covered
bin auto[8:11]	8	1	–	Covered
bin auto[12:15]	14	1	–	Covered
bin auto[16:19]	8	1	–	Covered
bin auto[20:23]	12	1	–	Covered
bin auto[24:27]	6	1	–	Covered
bin auto[28:31]	16	1	–	Covered
bin auto[32:35]	16	1	–	Covered
bin auto[36:39]	30	1	–	Covered
bin auto[40:43]	18	1	–	Covered
bin auto[44:47]	24	1	–	Covered
bin auto[48:51]	14	1	–	Covered
bin auto[52:55]	16	1	–	Covered
bin auto[56:59]	20	1	–	Covered
bin auto[60:63]	12	1	–	Covered
bin auto[64:67]	26	1	–	Covered
bin auto[68:71]	20	1	–	Covered
bin auto[72:75]	14	1	–	Covered
bin auto[76:79]	18	1	–	Covered
bin auto[80:83]	16	1	–	Covered
bin auto[84:87]	22	1	–	Covered
bin auto[88:91]	6	1	–	Covered
bin auto[92:95]	6	1	–	Covered
bin auto[96:99]	26	1	–	Covered
bin auto[100:103]	22	1	–	Covered
bin auto[104:107]	14	1	–	Covered
bin auto[108:111]	24	1	–	Covered
bin auto[112:115]	10	1	–	Covered
bin auto[116:119]	14	1	–	Covered
bin auto[120:123]	10	1	–	Covered
bin auto[124:127]	18	1	–	Covered
Coverpoint opcode	100.00%	100	–	Covered
covered/total bins:	4	4	–	
missing/total bins:	0	4	–	
% Hit:	100.00%	100	–	
bin auto[ADD]	246	1	–	Covered
bin auto[SUB]	262	1	–	Covered
bin auto[MULT]	236	1	–	Covered
bin auto[DIV]	256	1	–	Covered

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /alu_pkg/alu_cfg/cg_inputs	100.00%	100	–	Covered
covered/total bins:	132	132	–	
missing/total bins:	0	132	–	
% Hit:	100.00%	100	–	
Coverpoint operand1	100.00%	100	–	Covered
covered/total bins:	64	64	–	
missing/total bins:	0	64	–	
% Hit:	100.00%	100	–	
bin auto[−128:−125]	16	1	–	Covered
bin auto[−124:−121]	4	1	–	Covered
bin auto[−120:−117]	20	1	–	Covered
bin auto[−116:−113]	16	1	–	Covered
bin auto[−112:−109]	14	1	–	Covered
bin auto[−108:−105]	6	1	–	Covered
bin auto[−104:−101]	18	1	–	Covered
bin auto[−100:−97]	20	1	–	Covered
bin auto[−96:−93]	16	1	–	Covered
bin auto[−92:−89]	10	1	–	Covered
bin auto[−88:−85]	6	1	–	Covered
bin auto[−84:−81]	12	1	–	Covered
bin auto[−80:−77]	12	1	–	Covered
bin auto[−76:−73]	26	1	–	Covered
bin auto[−72:−69]	16	1	–	Covered
bin auto[−68:−65]	18	1	–	Covered
bin auto[−64:−61]	12	1	–	Covered
bin auto[−60:−57]	16	1	–	Covered
bin auto[−56:−53]	16	1	–	Covered
bin auto[−52:−49]	24	1	–	Covered
bin auto[−48:−45]	26	1	–	Covered
bin auto[−44:−41]	20	1	–	Covered
bin auto[−40:−37]	16	1	–	Covered
bin auto[−36:−33]	18	1	–	Covered
bin auto[−32:−29]	20	1	–	Covered
bin auto[−28:−25]	16	1	–	Covered
bin auto[−24:−21]	18	1	–	Covered
bin auto[−20:−17]	4	1	–	Covered
bin auto[−16:−13]	12	1	–	Covered
bin auto[−12:−9]	12	1	–	Covered
bin auto[−8:−5]	16	1	–	Covered
bin auto[−4:−1]	16	1	–	Covered
bin auto[0:3]	14	1	–	Covered
bin auto[4:7]	10	1	–	Covered
bin auto[8:11]	10	1	–	Covered
bin auto[12:15]	12	1	–	Covered
bin auto[16:19]	28	1	–	Covered
bin auto[20:23]	26	1	–	Covered
bin auto[24:27]	8	1	–	Covered
bin auto[28:31]	20	1	–	Covered
bin auto[32:35]	14	1	–	Covered
bin auto[36:39]	16	1	–	Covered
bin auto[40:43]	16	1	–	Covered
bin auto[44:47]	20	1	–	Covered
bin auto[48:51]	6	1	–	Covered
bin auto[52:55]	12	1	–	Covered
bin auto[56:59]	12	1	–	Covered
bin auto[60:63]	16	1	–	Covered
bin auto[64:67]	18	1	–	Covered
bin auto[68:71]	12	1	–	Covered
bin auto[72:75]	18	1	–	Covered
bin auto[76:79]	18	1	–	Covered
bin auto[80:83]	12	1	–	Covered
bin auto[84:87]	12	1	–	Covered
bin auto[88:91]	14	1	–	Covered
bin auto[92:95]	22	1	–	Covered
bin auto[96:99]	22	1	–	Covered
bin auto[100:103]	8	1	–	Covered
bin auto[104:107]	22	1	–	Covered
bin auto[108:111]	12	1	–	Covered
bin auto[112:115]	16	1	–	Covered
bin auto[116:119]	16	1	–	Covered
bin auto[120:123]	22	1	–	Covered
bin auto[124:127]	24	1	–	Covered
Coverpoint operand2	100.00%	100	–	Covered
covered/total bins:	64	64	–	
missing/total bins:	0	64	–	
% Hit:	100.00%	100	–	
bin auto[−128:−125]	6	1	–	Covered
bin auto[−124:−121]	16	1	–	Covered
bin auto[−120:−117]	18	1	–	Covered
bin auto[−116:−113]	18	1	–	Covered
bin auto[−112:−109]	6	1	–	Covered
bin auto[−108:−105]	12	1	–	Covered
bin auto[−104:−101]	26	1	–	Covered
bin auto[−100:−97]	10	1	–	Covered
bin auto[−96:−93]	14	1	–	Covered
bin auto[−92:−89]	26	1	–	Covered
bin auto[−88:−85]	8	1	–	Covered
bin auto[−84:−81]	18	1	–	Covered

bin auto[−80:−77]	10	1	–	Covered
bin auto[−76:−73]	28	1	–	Covered
bin auto[−72:−69]	18	1	–	Covered
bin auto[−68:−65]	4	1	–	Covered
bin auto[−64:−61]	8	1	–	Covered
bin auto[−60:−57]	12	1	–	Covered
bin auto[−56:−53]	18	1	–	Covered
bin auto[−52:−49]	16	1	–	Covered
bin auto[−48:−45]	14	1	–	Covered
bin auto[−44:−41]	10	1	–	Covered
bin auto[−40:−37]	22	1	–	Covered
bin auto[−36:−33]	20	1	–	Covered
bin auto[−32:−29]	14	1	–	Covered
bin auto[−28:−25]	18	1	–	Covered
bin auto[−24:−21]	24	1	–	Covered
bin auto[−20:−17]	10	1	–	Covered
bin auto[−16:−13]	20	1	–	Covered
bin auto[−12:−9]	22	1	–	Covered
bin auto[−8:−5]	16	1	–	Covered
bin auto[−4:−1]	14	1	–	Covered
bin auto[0:3]	10	1	–	Covered
bin auto[4:7]	14	1	–	Covered
bin auto[8:11]	8	1	–	Covered
bin auto[12:15]	14	1	–	Covered
bin auto[16:19]	8	1	–	Covered
bin auto[20:23]	12	1	–	Covered
bin auto[24:27]	6	1	–	Covered
bin auto[28:31]	16	1	–	Covered
bin auto[32:35]	16	1	–	Covered
bin auto[36:39]	30	1	–	Covered
bin auto[40:43]	18	1	–	Covered
bin auto[44:47]	24	1	–	Covered
bin auto[48:51]	14	1	–	Covered
bin auto[52:55]	16	1	–	Covered
bin auto[56:59]	20	1	–	Covered
bin auto[60:63]	12	1	–	Covered
bin auto[64:67]	26	1	–	Covered
bin auto[68:71]	20	1	–	Covered
bin auto[72:75]	14	1	–	Covered
bin auto[76:79]	18	1	–	Covered
bin auto[80:83]	16	1	–	Covered
bin auto[84:87]	22	1	–	Covered
bin auto[88:91]	6	1	–	Covered
bin auto[92:95]	6	1	–	Covered
bin auto[96:99]	26	1	–	Covered
bin auto[100:103]	22	1	–	Covered
bin auto[104:107]	14	1	–	Covered
bin auto[108:111]	24	1	–	Covered
bin auto[112:115]	10	1	–	Covered
bin auto[116:119]	14	1	–	Covered
bin auto[120:123]	10	1	–	Covered
bin auto[124:127]	18	1	–	Covered
Coverpoint opcode	100.00%	100	–	Covered
covered/total bins:	4	4	–	
missing/total bins:	0	4	–	
% Hit:	100.00%	100	–	
bin auto[ADD]	246	1	–	Covered
bin auto[SUB]	262	1	–	Covered
bin auto[MULT]	236	1	–	Covered
bin auto[DIV]	256	1	–	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

1.9 9.Waveform

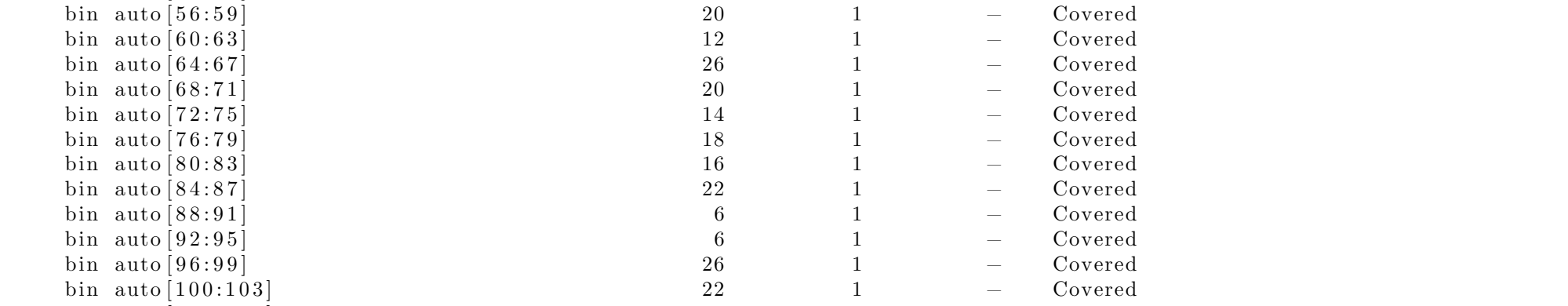

Figure 1: simulation waveform

```
# Saving coverage database on exit...
# End time: 09:38:29 on Apr 01,2025, Elapsed time: 0:01:56
# Errors: 2, Warnings: 0
# vsim -voptargs="+acc" work.alu_tb -coverage
# Start time: 09:38:29 on Apr 01,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.alu_pkg(fast)
# Loading work.alu_tb_sv_unit(fast)
# Loading work.alu_tb(fast)
```

```
# Loading work.alu_seq(fast)
# PASS at time 35: operand1=0, operand2=-38, opcode=MULT, out=0
# PASS at time 55: operand1=-74, operand2=-20, opcode=DIV, out=3
# PASS at time 75: operand1=31, operand2=26, opcode=SUB, out=5
# PASS at time 95: operand1=-67, operand2=-28, opcode=MULT, out=84
# PASS at time 115: operand1=-128, operand2=76, opcode=MULT, out=0
# PASS at time 135: operand1=21, operand2=-24, opcode=DIV, out=0
# PASS at time 155: operand1=-111, operand2=-89, opcode=MULT, out=-105
# PASS at time 175: operand1=-93, operand2=0, opcode=MULT, out=0
# PASS at time 195: operand1=104, operand2=-49, opcode=MULT, out=24
# PASS at time 215: operand1=47, operand2=-87, opcode=MULT, out=7
# PASS at time 235: operand1=124, operand2=29, opcode=ADD, out=-103
# PASS at time 255: operand1=116, operand2=109, opcode=DIV, out=1
# PASS at time 275: operand1=43, operand2=75, opcode=MULT, out=-103
# PASS at time 295: operand1=127, operand2=122, opcode=ADD, out=-7
# PASS at time 315: operand1=80, operand2=47, opcode=DIV, out=1
# PASS at time 335: operand1=86, operand2=-128, opcode=ADD, out=-42
# PASS at time 355: operand1=125, operand2=46, opcode=DIV, out=2
# PASS at time 375: operand1=-74, operand2=106, opcode=MULT, out=92
# PASS at time 395: operand1=101, operand2=0, opcode=MULT, out=0
# PASS at time 415: operand1=94, operand2=-13, opcode=DIV, out=-7
# PASS at time 435: operand1=57, operand2=-11, opcode=SUB, out=68
# PASS at time 455: operand1=-18, operand2=-18, opcode=MULT, out=68
# PASS at time 475: operand1=75, operand2=8, opcode=MULT, out=88
# PASS at time 495: operand1=112, operand2=15, opcode=DIV, out=7
# PASS at time 515: operand1=55, operand2=32, opcode=MULT, out=-32
# PASS at time 535: operand1=-55, operand2=108, opcode=SUB, out=93
# PASS at time 555: operand1=-105, operand2=-94, opcode=DIV, out=1
# PASS at time 575: operand1=-6, operand2=-111, opcode=DIV, out=0
# PASS at time 595: operand1=9, operand2=-116, opcode=DIV, out=0
# PASS at time 615: operand1=-75, operand2=66, opcode=SUB, out=115
# PASS at time 635: operand1=10, operand2=-64, opcode=DIV, out=0
# PASS at time 655: operand1=100, operand2=16, opcode=DIV, out=6
# PASS at time 675: operand1=-24, operand2=-39, opcode=SUB, out=15
# PASS at time 695: operand1=47, operand2=-128, opcode=DIV, out=0
# PASS at time 715: operand1=-104, operand2=-92, opcode=ADD, out=60
# PASS at time 735: operand1=0, operand2=54, opcode=ADD, out=54
# PASS at time 755: operand1=0, operand2=-56, opcode=MULT, out=0
# PASS at time 775: operand1=77, operand2=-114, opcode=DIV, out=0
# PASS at time 795: operand1=-23, operand2=33, opcode=SUB, out=-56
# PASS at time 815: operand1=0, operand2=53, opcode=DIV, out=0
# PASS at time 835: operand1=-84, operand2=74, opcode=DIV, out=-1
# PASS at time 855: operand1=-98, operand2=-101, opcode=ADD, out=57
# PASS at time 875: operand1=0, operand2=4, opcode=ADD, out=4
# PASS at time 895: operand1=-44, operand2=43, opcode=MULT, out=-100
# PASS at time 915: operand1=-127, operand2=-103, opcode=ADD, out=26
# PASS at time 935: operand1=0, operand2=-92, opcode=ADD, out=-92
# PASS at time 955: operand1=83, operand2=-81, opcode=ADD, out=2
# PASS at time 975: operand1=50, operand2=50, opcode=ADD, out=100
# PASS at time 995: operand1=-73, operand2=0, opcode=DIV, out=0
# PASS at time 1015: operand1=50, operand2=0, opcode=DIV, out=0
# PASS at time 1035: operand1=0, operand2=90, opcode=ADD, out=90
# PASS at time 1055: operand1=2, operand2=0, opcode=MULT, out=0
# PASS at time 1075: operand1=-40, operand2=-128, opcode=ADD, out=88
# PASS at time 1095: operand1=6, operand2=9, opcode=MULT, out=54
# PASS at time 1115: operand1=16, operand2=-123, opcode=MULT, out=80
# PASS at time 1135: operand1=2, operand2=51, opcode=SUB, out=-49
# PASS at time 1155: operand1=-128, operand2=36, opcode=SUB, out=92
# PASS at time 1175: operand1=-93, operand2=-128, opcode=SUB, out=35
# PASS at time 1195: operand1=69, operand2=-128, opcode=MULT, out=-128
# PASS at time 1215: operand1=-31, operand2=113, opcode=SUB, out=112
# PASS at time 1235: operand1=0, operand2=32, opcode=MULT, out=0
# PASS at time 1255: operand1=-9, operand2=0, opcode=MULT, out=0
# PASS at time 1275: operand1=0, operand2=0, opcode=SUB, out=0
# PASS at time 1295: operand1=-96, operand2=-16, opcode=ADD, out=-112
# PASS at time 1315: operand1=-128, operand2=10, opcode=DIV, out=-12
# PASS at time 1335: operand1=-35, operand2=-82, opcode=MULT, out=54
# PASS at time 1355: operand1=126, operand2=103, opcode=DIV, out=1
# PASS at time 1375: operand1=36, operand2=-107, opcode=DIV, out=0
# PASS at time 1395: operand1=-33, operand2=-110, opcode=MULT, out=46
# PASS at time 1415: operand1=-128, operand2=-26, opcode=DIV, out=4
# PASS at time 1435: operand1=-20, operand2=-30, opcode=SUB, out=10
# PASS at time 1455: operand1=-34, operand2=-67, opcode=SUB, out=33
# PASS at time 1475: operand1=-106, operand2=-44, opcode=ADD, out=106
# PASS at time 1495: operand1=51, operand2=-38, opcode=MULT, out=110
# PASS at time 1515: operand1=-42, operand2=0, opcode=SUB, out=-42
# PASS at time 1535: operand1=107, operand2=-34, opcode=ADD, out=73
# PASS at time 1555: operand1=-33, operand2=-121, opcode=SUB, out=88
# PASS at time 1575: operand1=46, operand2=-39, opcode=MULT, out=-2
# PASS at time 1595: operand1=68, operand2=51, opcode=SUB, out=17
# PASS at time 1615: operand1=73, operand2=-42, opcode=MULT, out=6
# PASS at time 1635: operand1=23, operand2=19, opcode=ADD, out=42
# PASS at time 1655: operand1=124, operand2=30, opcode=DIV, out=4
# PASS at time 1675: operand1=-128, operand2=0, opcode=MULT, out=0
# PASS at time 1695: operand1=54, operand2=-115, opcode=MULT, out=-66
# PASS at time 1715: operand1=19, operand2=8, opcode=SUB, out=11
# PASS at time 1735: operand1=88, operand2=-67, opcode=MULT, out=-8
# PASS at time 1755: operand1=-54, operand2=93, opcode=DIV, out=0
# PASS at time 1775: operand1=6, operand2=-104, opcode=SUB, out=110
# PASS at time 1795: operand1=-128, operand2=-32, opcode=ADD, out=96
# PASS at time 1815: operand1=0, operand2=-5, opcode=SUB, out=5
# PASS at time 1835: operand1=-29, operand2=-59, opcode=DIV, out=0
```

```
# PASS at time 1855: operand1=-36, operand2=11, opcode=MULT, out=116
# PASS at time 1875: operand1=-21, operand2=-122, opcode=SUB, out=101
# PASS at time 1895: operand1=36, operand2=67, opcode=DIV, out=0
# PASS at time 1915: operand1=-128, operand2=97, opcode=SUB, out=31
# PASS at time 1935: operand1=41, operand2=20, opcode=MULT, out=52
# PASS at time 1955: operand1=-115, operand2=-128, opcode=MULT, out=-128
# PASS at time 1975: operand1=0, operand2=-118, opcode=MULT, out=0
# PASS at time 1995: operand1=-128, operand2=-34, opcode=MULT, out=0
# PASS at time 2015: operand1=113, operand2=-17, opcode=SUB, out=-126
# PASS at time 2035: operand1=-128, operand2=-75, opcode=DIV, out=1
# PASS at time 2055: operand1=4, operand2=40, opcode=DIV, out=0
# PASS at time 2075: operand1=-50, operand2=7, opcode=MULT, out=-94
# PASS at time 2095: operand1=83, operand2=104, opcode=DIV, out=0
# PASS at time 2115: operand1=0, operand2=122, opcode=DIV, out=0
# PASS at time 2135: operand1=123, operand2=9, opcode=DIV, out=13
# PASS at time 2155: operand1=0, operand2=121, opcode=SUB, out=-121
# PASS at time 2175: operand1=85, operand2=-29, opcode=ADD, out=56
# PASS at time 2195: operand1=-21, operand2=19, opcode=MULT, out=113
# PASS at time 2215: operand1=-128, operand2=-40, opcode=ADD, out=88
# PASS at time 2235: operand1=122, operand2=-128, opcode=ADD, out=-6
# PASS at time 2255: operand1=49, operand2=-80, opcode=DIV, out=0
# PASS at time 2275: operand1=0, operand2=33, opcode=DIV, out=0
# PASS at time 2295: operand1=-64, operand2=104, opcode=SUB, out=88
# PASS at time 2315: operand1=-105, operand2=-43, opcode=MULT, out=-93
# PASS at time 2335: operand1=31, operand2=-27, opcode=DIV, out=-1
# PASS at time 2355: operand1=-67, operand2=-84, opcode=ADD, out=105
# PASS at time 2375: operand1=-128, operand2=-128, opcode=MULT, out=0
# PASS at time 2395: operand1=-35, operand2=36, opcode=SUB, out=-71
# PASS at time 2415: operand1=43, operand2=-24, opcode=MULT, out=-8
# PASS at time 2435: operand1=-128, operand2=-116, opcode=SUB, out=-12
# PASS at time 2455: operand1=-128, operand2=-114, opcode=DIV, out=1
# PASS at time 2475: operand1=82, operand2=-128, opcode=ADD, out=-46
# PASS at time 2495: operand1=106, operand2=0, opcode=ADD, out=106
# PASS at time 2515: operand1=-27, operand2=-3, opcode=ADD, out=-30
# PASS at time 2535: operand1=35, operand2=-27, opcode=DIV, out=-1
# PASS at time 2555: operand1=-83, operand2=27, opcode=DIV, out=-3
# PASS at time 2575: operand1=0, operand2=70, opcode=MULT, out=0
# PASS at time 2595: operand1=-128, operand2=25, opcode=DIV, out=-5
# PASS at time 2615: operand1=-4, operand2=-43, opcode=SUB, out=39
# PASS at time 2635: operand1=0, operand2=-128, opcode=ADD, out=-128
# PASS at time 2655: operand1=-38, operand2=5, opcode=DIV, out=-7
# PASS at time 2675: operand1=0, operand2=-87, opcode=DIV, out=0
# PASS at time 2695: operand1=99, operand2=0, opcode=DIV, out=0
# PASS at time 2715: operand1=0, operand2=-86, opcode=DIV, out=0
# PASS at time 2735: operand1=0, operand2=98, opcode=SUB, out=-98
# PASS at time 2755: operand1=60, operand2=85, opcode=MULT, out=-20
# PASS at time 2775: operand1=-51, operand2=104, opcode=ADD, out=53
# PASS at time 2795: operand1=-128, operand2=118, opcode=DIV, out=-1
# PASS at time 2815: operand1=113, operand2=75, opcode=SUB, out=38
# PASS at time 2835: operand1=-128, operand2=-75, opcode=ADD, out=53
# PASS at time 2855: operand1=1, operand2=-63, opcode=DIV, out=0
# PASS at time 2875: operand1=-128, operand2=-23, opcode=SUB, out=-105
# PASS at time 2895: operand1=-114, operand2=67, opcode=ADD, out=-47
# PASS at time 2915: operand1=-22, operand2=-46, opcode=MULT, out=-12
# PASS at time 2935: operand1=-61, operand2=-128, opcode=ADD, out=67
# PASS at time 2955: operand1=-128, operand2=25, opcode=DIV, out=-5
# PASS at time 2975: operand1=21, operand2=69, opcode=ADD, out=90
# PASS at time 2995: operand1=-33, operand2=-54, opcode=MULT, out=-10
# PASS at time 3015: operand1=83, operand2=0, opcode=ADD, out=83
# PASS at time 3035: operand1=59, operand2=0, opcode=DIV, out=0
# PASS at time 3055: operand1=-11, operand2=-99, opcode=MULT, out=65
# PASS at time 3075: operand1=0, operand2=87, opcode=ADD, out=87
# PASS at time 3095: operand1=22, operand2=127, opcode=DIV, out=0
# PASS at time 3115: operand1=-98, operand2=-123, opcode=ADD, out=35
# PASS at time 3135: operand1=-70, operand2=51, opcode=SUB, out=-121
# PASS at time 3155: operand1=-94, operand2=-114, opcode=DIV, out=0
# PASS at time 3175: operand1=31, operand2=-128, opcode=DIV, out=0
# PASS at time 3195: operand1=122, operand2=53, opcode=SUB, out=69
# PASS at time 3215: operand1=52, operand2=0, opcode=DIV, out=0
# PASS at time 3235: operand1=108, operand2=-116, opcode=DIV, out=0
# PASS at time 3255: operand1=27, operand2=-128, opcode=SUB, out=-101
# PASS at time 3275: operand1=33, operand2=119, opcode=MULT, out=87
# PASS at time 3295: operand1=47, operand2=-22, opcode=DIV, out=-2
# PASS at time 3315: operand1=-52, operand2=-95, opcode=SUB, out=43
# PASS at time 3335: operand1=-115, operand2=67, opcode=SUB, out=74
# PASS at time 3355: operand1=86, operand2=0, opcode=SUB, out=86
# PASS at time 3375: operand1=72, operand2=21, opcode=ADD, out=93
# PASS at time 3395: operand1=-63, operand2=-128, opcode=DIV, out=0
# PASS at time 3415: operand1=117, operand2=-100, opcode=SUB, out=-39
# PASS at time 3435: operand1=-22, operand2=-22, opcode=ADD, out=-44
# PASS at time 3455: operand1=90, operand2=34, opcode=DIV, out=2
# PASS at time 3475: operand1=105, operand2=121, opcode=SUB, out=-16
# PASS at time 3495: operand1=101, operand2=36, opcode=MULT, out=52
# PASS at time 3515: operand1=-81, operand2=-107, opcode=DIV, out=0
# PASS at time 3535: operand1=83, operand2=-128, opcode=MULT, out=-128
# PASS at time 3555: operand1=0, operand2=-72, opcode=SUB, out=72
# PASS at time 3575: operand1=-125, operand2=102, opcode=DIV, out=-1
# PASS at time 3595: operand1=-110, operand2=67, opcode=SUB, out=79
# PASS at time 3615: operand1=32, operand2=-88, opcode=MULT, out=0
# PASS at time 3635: operand1=114, operand2=127, opcode=MULT, out=-114
# PASS at time 3655: operand1=-105, operand2=65, opcode=ADD, out=-40
# PASS at time 3675: operand1=-117, operand2=119, opcode=ADD, out=2
```

```
# PASS at time 3695: operand1=-112, operand2=101, opcode=SUB, out=43
# PASS at time 3715: operand1=-128, operand2=40, opcode=ADD, out=-88
# PASS at time 3735: operand1=40, operand2=-80, opcode=ADD, out=-40
# PASS at time 3755: operand1=-67, operand2=-128, opcode=DIV, out=0
# PASS at time 3775: operand1=-76, operand2=114, opcode=SUB, out=66
# PASS at time 3795: operand1=-92, operand2=0, opcode=DIV, out=0
# PASS at time 3815: operand1=0, operand2=-114, opcode=ADD, out=-114
# PASS at time 3835: operand1=18, operand2=-86, opcode=DIV, out=0
# PASS at time 3855: operand1=-120, operand2=60, opcode=SUB, out=76
# PASS at time 3875: operand1=-66, operand2=104, opcode=SUB, out=86
# PASS at time 3895: operand1=69, operand2=37, opcode=ADD, out=106
# PASS at time 3915: operand1=-47, operand2=31, opcode=DIV, out=-1
# PASS at time 3935: operand1=-22, operand2=-118, opcode=DIV, out=0
# PASS at time 3955: operand1=124, operand2=-16, opcode=DIV, out=-7
# PASS at time 3975: operand1=0, operand2=-20, opcode=DIV, out=0
# PASS at time 3995: operand1=-6, operand2=-41, opcode=ADD, out=-47
# PASS at time 4015: operand1=-46, operand2=116, opcode=MULT, out=40
# PASS at time 4035: operand1=-124, operand2=40, opcode=MULT, out=-96
# PASS at time 4055: operand1=0, operand2=32, opcode=ADD, out=32
# PASS at time 4075: operand1=-5, operand2=29, opcode=SUB, out=-34
# PASS at time 4095: operand1=-124, operand2=-27, opcode=ADD, out=105
# PASS at time 4115: operand1=-96, operand2=-77, opcode=DIV, out=1
# PASS at time 4135: operand1=118, operand2=-11, opcode=MULT, out=-18
# PASS at time 4155: operand1=0, operand2=-93, opcode=ADD, out=-93
# PASS at time 4175: operand1=-77, operand2=97, opcode=DIV, out=0
# PASS at time 4195: operand1=0, operand2=-126, opcode=SUB, out=126
# PASS at time 4215: operand1=-34, operand2=15, opcode=SUB, out=-49
# PASS at time 4235: operand1=84, operand2=-74, opcode=MULT, out=-72
# PASS at time 4255: operand1=117, operand2=-95, opcode=SUB, out=-44
# PASS at time 4275: operand1=51, operand2=111, opcode=DIV, out=0
# PASS at time 4295: operand1=-102, operand2=-24, opcode=SUB, out=-78
# PASS at time 4315: operand1=98, operand2=-101, opcode=MULT, out=86
# PASS at time 4335: operand1=69, operand2=42, opcode=MULT, out=82
# PASS at time 4355: operand1=-30, operand2=-66, opcode=SUB, out=36
# PASS at time 4375: operand1=60, operand2=-74, opcode=SUB, out=-122
# PASS at time 4395: operand1=0, operand2=48, opcode=DIV, out=0
# PASS at time 4415: operand1=-26, operand2=0, opcode=DIV, out=0
# PASS at time 4435: operand1=0, operand2=-128, opcode=ADD, out=-128
# PASS at time 4455: operand1=-128, operand2=-76, opcode=MULT, out=0
# PASS at time 4475: operand1=0, operand2=118, opcode=DIV, out=0
# PASS at time 4495: operand1=28, operand2=-71, opcode=DIV, out=0
# PASS at time 4515: operand1=0, operand2=32, opcode=ADD, out=32
# PASS at time 4535: operand1=87, operand2=-25, opcode=SUB, out=112
# PASS at time 4555: operand1=122, operand2=-71, opcode=ADD, out=51
# PASS at time 4575: operand1=9, operand2=-96, opcode=ADD, out=-87
# PASS at time 4595: operand1=-10, operand2=-11, opcode=ADD, out=-21
# PASS at time 4615: operand1=-21, operand2=-24, opcode=DIV, out=0
# PASS at time 4635: operand1=38, operand2=44, opcode=DIV, out=0
# PASS at time 4655: operand1=-11, operand2=-124, opcode=MULT, out=84
# PASS at time 4675: operand1=-16, operand2=-13, opcode=DIV, out=1
# PASS at time 4695: operand1=-114, operand2=90, opcode=MULT, out=-20
# PASS at time 4715: operand1=0, operand2=-24, opcode=ADD, out=-24
# PASS at time 4735: operand1=-112, operand2=-128, opcode=DIV, out=0
# PASS at time 4755: operand1=74, operand2=0, opcode=ADD, out=74
# PASS at time 4775: operand1=13, operand2=0, opcode=MULT, out=0
# PASS at time 4795: operand1=28, operand2=70, opcode=SUB, out=-42
# PASS at time 4815: operand1=-63, operand2=-63, opcode=MULT, out=-127
# PASS at time 4835: operand1=40, operand2=66, opcode=DIV, out=0
# PASS at time 4855: operand1=-47, operand2=-60, opcode=SUB, out=13
# PASS at time 4875: operand1=34, operand2=-19, opcode=DIV, out=-1
# PASS at time 4895: operand1=-89, operand2=-23, opcode=SUB, out=-66
# PASS at time 4915: operand1=39, operand2=-54, opcode=ADD, out=-15
# PASS at time 4935: operand1=0, operand2=91, opcode=ADD, out=91
# PASS at time 4955: operand1=-128, operand2=-43, opcode=SUB, out=-85
# PASS at time 4975: operand1=-106, operand2=36, opcode=MULT, out=24
# PASS at time 4995: operand1=-6, operand2=66, opcode=ADD, out=60
# PASS at time 5015: operand1=86, operand2=46, opcode=MULT, out=116
# PASS at time 5035: operand1=-128, operand2=98, opcode=SUB, out=30
# PASS at time 5055: operand1=-5, operand2=81, opcode=MULT, out=107
# PASS at time 5075: operand1=80, operand2=119, opcode=MULT, out=48
# PASS at time 5095: operand1=0, operand2=-38, opcode=MULT, out=0
# PASS at time 5115: operand1=-44, operand2=-73, opcode=MULT, out=-116
# PASS at time 5135: operand1=-128, operand2=0, opcode=MULT, out=0
# PASS at time 5155: operand1=0, operand2=0, opcode=SUB, out=0
# PASS at time 5175: operand1=124, operand2=58, opcode=DIV, out=2
# PASS at time 5195: operand1=15, operand2=124, opcode=DIV, out=0
# PASS at time 5215: operand1=-128, operand2=80, opcode=ADD, out=-48
# PASS at time 5235: operand1=-128, operand2=67, opcode=ADD, out=-61
# PASS at time 5255: operand1=-73, operand2=-81, opcode=MULT, out=25
# PASS at time 5275: operand1=-22, operand2=57, opcode=MULT, out=26
# PASS at time 5295: operand1=-58, operand2=4, opcode=SUB, out=-62
# PASS at time 5315: operand1=-118, operand2=-113, opcode=MULT, out=22
# PASS at time 5335: operand1=-98, operand2=-11, opcode=ADD, out=-109
# PASS at time 5355: operand1=87, operand2=-2, opcode=SUB, out=89
# PASS at time 5375: operand1=116, operand2=-128, opcode=ADD, out=-12
# PASS at time 5395: operand1=95, operand2=70, opcode=SUB, out=25
# PASS at time 5415: operand1=34, operand2=-51, opcode=ADD, out=-17
# PASS at time 5435: operand1=-75, operand2=99, opcode=DIV, out=0
# PASS at time 5455: operand1=0, operand2=-128, opcode=MULT, out=0
# PASS at time 5475: operand1=-63, operand2=95, opcode=DIV, out=0
# PASS at time 5495: operand1=78, operand2=-87, opcode=DIV, out=0
# PASS at time 5515: operand1=62, operand2=-27, opcode=ADD, out=35
```

```
# PASS at time 5535: operand1=-67, operand2=12, opcode=MULT, out=-36
# PASS at time 5555: operand1=-128, operand2=9, opcode=MULT, out=-128
# PASS at time 5575: operand1=-52, operand2=-128, opcode=DIV, out=0
# PASS at time 5595: operand1=-128, operand2=-97, opcode=ADD, out=31
# PASS at time 5615: operand1=44, operand2=-47, opcode=SUB, out=91
# PASS at time 5635: operand1=93, operand2=-2, opcode=SUB, out=95
# PASS at time 5655: operand1=0, operand2=124, opcode=MULT, out=0
# PASS at time 5675: operand1=19, operand2=123, opcode=DIV, out=0
# PASS at time 5695: operand1=-51, operand2=96, opcode=SUB, out=109
# PASS at time 5715: operand1=30, operand2=-23, opcode=DIV, out=-1
# PASS at time 5735: operand1=0, operand2=37, opcode=MULT, out=0
# PASS at time 5755: operand1=125, operand2=-73, opcode=ADD, out=52
# PASS at time 5775: operand1=30, operand2=125, opcode=SUB, out=-95
# PASS at time 5795: operand1=-116, operand2=-123, opcode=ADD, out=17
# PASS at time 5815: operand1=-25, operand2=-10, opcode=MULT, out=-6
# PASS at time 5835: operand1=-52, operand2=-84, opcode=ADD, out=120
# PASS at time 5855: operand1=30, operand2=69, opcode=DIV, out=0
# PASS at time 5875: operand1=89, operand2=-128, opcode=SUB, out=-39
# PASS at time 5895: operand1=2, operand2=-27, opcode=ADD, out=-25
# PASS at time 5915: operand1=-117, operand2=99, opcode=ADD, out=-18
# PASS at time 5935: operand1=-52, operand2=0, opcode=DIV, out=0
# PASS at time 5955: operand1=-22, operand2=59, opcode=SUB, out=-81
# PASS at time 5975: operand1=-112, operand2=-31, opcode=DIV, out=3
# PASS at time 5995: operand1=-73, operand2=-94, opcode=DIV, out=0
# PASS at time 6015: operand1=-85, operand2=-40, opcode=SUB, out=-45
# PASS at time 6035: operand1=0, operand2=-10, opcode=SUB, out=10
# PASS at time 6055: operand1=125, operand2=13, opcode=SUB, out=112
# PASS at time 6075: operand1=44, operand2=-69, opcode=SUB, out=113
# PASS at time 6095: operand1=0, operand2=29, opcode=ADD, out=29
# PASS at time 6115: operand1=-128, operand2=65, opcode=SUB, out=63
# PASS at time 6135: operand1=-5, operand2=53, opcode=ADD, out=48
# PASS at time 6155: operand1=-60, operand2=-83, opcode=SUB, out=23
# PASS at time 6175: operand1=-128, operand2=-49, opcode=ADD, out=79
# PASS at time 6195: operand1=-79, operand2=-128, opcode=SUB, out=49
# PASS at time 6215: operand1=127, operand2=38, opcode=MULT, out=-38
# PASS at time 6235: operand1=17, operand2=80, opcode=SUB, out=-63
# PASS at time 6255: operand1=100, operand2=0, opcode=ADD, out=100
# PASS at time 6275: operand1=26, operand2=-78, opcode=ADD, out=-52
# PASS at time 6295: operand1=-128, operand2=102, opcode=DIV, out=-1
# PASS at time 6315: operand1=0, operand2=-8, opcode=SUB, out=8
# PASS at time 6335: operand1=-112, operand2=-112, opcode=SUB, out=0
# PASS at time 6355: operand1=-66, operand2=-110, opcode=SUB, out=44
# PASS at time 6375: operand1=-5, operand2=-51, opcode=MULT, out=-1
# PASS at time 6395: operand1=42, operand2=75, opcode=MULT, out=78
# PASS at time 6415: operand1=-126, operand2=-1, opcode=ADD, out=-127
# PASS at time 6435: operand1=13, operand2=-55, opcode=SUB, out=68
# PASS at time 6455: operand1=-128, operand2=-55, opcode=ADD, out=73
# PASS at time 6475: operand1=-32, operand2=-128, opcode=SUB, out=96
# PASS at time 6495: operand1=-40, operand2=45, opcode=ADD, out=5
# PASS at time 6515: operand1=63, operand2=-78, opcode=MULT, out=-50
# PASS at time 6535: operand1=-23, operand2=11, opcode=DIV, out=-2
# PASS at time 6555: operand1=-115, operand2=27, opcode=MULT, out=-33
# PASS at time 6575: operand1=92, operand2=20, opcode=SUB, out=72
# PASS at time 6595: operand1=3, operand2=-77, opcode=ADD, out=-74
# PASS at time 6615: operand1=-50, operand2=-128, opcode=ADD, out=78
# PASS at time 6635: operand1=-128, operand2=0, opcode=MULT, out=0
# PASS at time 6655: operand1=-38, operand2=-37, opcode=DIV, out=1
# PASS at time 6675: operand1=95, operand2=0, opcode=SUB, out=95
# PASS at time 6695: operand1=0, operand2=0, opcode=MULT, out=0
# PASS at time 6715: operand1=-36, operand2=0, opcode=SUB, out=-36
# PASS at time 6735: operand1=58, operand2=102, opcode=DIV, out=0
# PASS at time 6755: operand1=12, operand2=-67, opcode=SUB, out=79
# PASS at time 6775: operand1=69, operand2=0, opcode=ADD, out=69
# PASS at time 6795: operand1=-14, operand2=36, opcode=ADD, out=22
# PASS at time 6815: operand1=0, operand2=8, opcode=SUB, out=-8
# PASS at time 6835: operand1=-128, operand2=115, opcode=ADD, out=-13
# PASS at time 6855: operand1=-25, operand2=101, opcode=SUB, out=-126
# PASS at time 6875: operand1=0, operand2=-6, opcode=SUB, out=6
# PASS at time 6895: operand1=0, operand2=-15, opcode=DIV, out=0
# PASS at time 6915: operand1=110, operand2=43, opcode=DIV, out=2
# PASS at time 6935: operand1=-122, operand2=8, opcode=SUB, out=126
# PASS at time 6955: operand1=100, operand2=83, opcode=ADD, out=-73
# PASS at time 6975: operand1=36, operand2=-10, opcode=SUB, out=46
# PASS at time 6995: operand1=-66, operand2=-126, opcode=ADD, out=64
# PASS at time 7015: operand1=-98, operand2=55, opcode=DIV, out=-1
# PASS at time 7035: operand1=-128, operand2=-100, opcode=DIV, out=1
# PASS at time 7055: operand1=81, operand2=0, opcode=DIV, out=0
# PASS at time 7075: operand1=85, operand2=-128, opcode=ADD, out=-43
# PASS at time 7095: operand1=122, operand2=-11, opcode=SUB, out=-123
# PASS at time 7115: operand1=-109, operand2=-128, opcode=SUB, out=19
# PASS at time 7135: operand1=-105, operand2=-70, opcode=MULT, out=-74
# PASS at time 7155: operand1=-128, operand2=-77, opcode=ADD, out=51
# PASS at time 7175: operand1=109, operand2=-53, opcode=SUB, out=-94
# PASS at time 7195: operand1=-61, operand2=-99, opcode=ADD, out=96
# PASS at time 7215: operand1=-38, operand2=-128, opcode=DIV, out=0
# PASS at time 7235: operand1=-104, operand2=0, opcode=DIV, out=0
# PASS at time 7255: operand1=15, operand2=-49, opcode=DIV, out=0
# PASS at time 7275: operand1=-128, operand2=-81, opcode=ADD, out=47
# PASS at time 7295: operand1=98, operand2=-53, opcode=SUB, out=-105
# PASS at time 7315: operand1=65, operand2=78, opcode=ADD, out=-113
# PASS at time 7335: operand1=-128, operand2=98, opcode=SUB, out=30
# PASS at time 7355: operand1=31, operand2=125, opcode=ADD, out=-100
```

```
# PASS at time 7375: operand1=-67, operand2=27, opcode=DIV, out=-2
# PASS at time 7395: operand1=0, operand2=-49, opcode=ADD, out=-49
# PASS at time 7415: operand1=91, operand2=99, opcode=MULT, out=49
# PASS at time 7435: operand1=0, operand2=0, opcode=ADD, out=0
# PASS at time 7455: operand1=-128, operand2=-128, opcode=ADD, out=0
# PASS at time 7475: operand1=112, operand2=-123, opcode=SUB, out=-21
# PASS at time 7495: operand1=-116, operand2=83, opcode=MULT, out=100
# PASS at time 7515: operand1=-6, operand2=-9, opcode=SUB, out=3
# PASS at time 7535: operand1=115, operand2=-106, opcode=ADD, out=9
# PASS at time 7555: operand1=-48, operand2=15, opcode=MULT, out=48
# PASS at time 7575: operand1=-16, operand2=122, opcode=ADD, out=106
# PASS at time 7595: operand1=-128, operand2=82, opcode=SUB, out=46
# PASS at time 7615: operand1=-25, operand2=98, opcode=MULT, out=110
# PASS at time 7635: operand1=97, operand2=-105, opcode=SUB, out=-54
# PASS at time 7655: operand1=0, operand2=49, opcode=MULT, out=0
# PASS at time 7675: operand1=78, operand2=-8, opcode=DIV, out=-9
# PASS at time 7695: operand1=0, operand2=87, opcode=DIV, out=0
# PASS at time 7715: operand1=-128, operand2=29, opcode=MULT, out=-128
# PASS at time 7735: operand1=-37, operand2=-128, opcode=DIV, out=0
# PASS at time 7755: operand1=-125, operand2=87, opcode=MULT, out=-123
# PASS at time 7775: operand1=39, operand2=0, opcode=MULT, out=0
# PASS at time 7795: operand1=76, operand2=30, opcode=ADD, out=106
# PASS at time 7815: operand1=-105, operand2=0, opcode=MULT, out=0
# PASS at time 7835: operand1=40, operand2=-128, opcode=ADD, out=-88
# PASS at time 7855: operand1=81, operand2=77, opcode=ADD, out=-98
# PASS at time 7875: operand1=39, operand2=77, opcode=ADD, out=116
# PASS at time 7895: operand1=52, operand2=-74, opcode=DIV, out=0
# PASS at time 7915: operand1=-45, operand2=122, opcode=MULT, out=-114
# PASS at time 7935: operand1=-13, operand2=119, opcode=SUB, out=124
# PASS at time 7955: operand1=115, operand2=-128, opcode=MULT, out=-128
# PASS at time 7975: operand1=-128, operand2=24, opcode=DIV, out=-5
# PASS at time 7995: operand1=101, operand2=-67, opcode=MULT, out=-111
# PASS at time 8015: operand1=-73, operand2=0, opcode=MULT, out=0
# PASS at time 8035: operand1=-27, operand2=-51, opcode=DIV, out=0
# PASS at time 8055: operand1=-78, operand2=7, opcode=MULT, out=-34
# PASS at time 8075: operand1=0, operand2=109, opcode=MULT, out=0
# PASS at time 8095: operand1=0, operand2=89, opcode=DIV, out=0
# PASS at time 8115: operand1=-128, operand2=100, opcode=ADD, out=-28
# PASS at time 8135: operand1=29, operand2=89, opcode=DIV, out=0
# PASS at time 8155: operand1=-128, operand2=-83, opcode=DIV, out=1
# PASS at time 8175: operand1=-9, operand2=111, opcode=SUB, out=-120
# PASS at time 8195: operand1=5, operand2=-95, opcode=SUB, out=100
# PASS at time 8215: operand1=-32, operand2=-62, opcode=ADD, out=-94
# PASS at time 8235: operand1=10, operand2=4, opcode=MULT, out=40
# PASS at time 8255: operand1=112, operand2=65, opcode=DIV, out=1
# PASS at time 8275: operand1=100, operand2=119, opcode=ADD, out=-37
# PASS at time 8295: operand1=-93, operand2=92, opcode=MULT, out=-108
# PASS at time 8315: operand1=52, operand2=-2, opcode=ADD, out=50
# PASS at time 8335: operand1=-128, operand2=-118, opcode=SUB, out=-10
# PASS at time 8355: operand1=52, operand2=39, opcode=ADD, out=91
# PASS at time 8375: operand1=-31, operand2=85, opcode=SUB, out=-116
# PASS at time 8395: operand1=-32, operand2=82, opcode=ADD, out=50
# PASS at time 8415: operand1=-128, operand2=0, opcode=MULT, out=0
# PASS at time 8435: operand1=96, operand2=117, opcode=SUB, out=-21
# PASS at time 8455: operand1=101, operand2=20, opcode=ADD, out=121
# PASS at time 8475: operand1=108, operand2=49, opcode=MULT, out=-84
# PASS at time 8495: operand1=0, operand2=-1, opcode=SUB, out=1
# PASS at time 8515: operand1=-84, operand2=-19, opcode=MULT, out=60
# PASS at time 8535: operand1=75, operand2=15, opcode=ADD, out=90
# PASS at time 8555: operand1=-88, operand2=-42, opcode=DIV, out=2
# PASS at time 8575: operand1=-17, operand2=-20, opcode=DIV, out=0
# PASS at time 8595: operand1=98, operand2=-81, opcode=SUB, out=-77
# PASS at time 8615: operand1=0, operand2=96, opcode=MULT, out=0
# PASS at time 8635: operand1=115, operand2=-26, opcode=MULT, out=82
# PASS at time 8655: operand1=83, operand2=-81, opcode=SUB, out=-92
# PASS at time 8675: operand1=83, operand2=-26, opcode=DIV, out=-3
# PASS at time 8695: operand1=2, operand2=-25, opcode=SUB, out=27
# PASS at time 8715: operand1=-70, operand2=45, opcode=ADD, out=-25
# PASS at time 8735: operand1=0, operand2=75, opcode=ADD, out=75
# PASS at time 8755: operand1=-76, operand2=-88, opcode=ADD, out=92
# PASS at time 8775: operand1=15, operand2=-111, opcode=SUB, out=126
# PASS at time 8795: operand1=-128, operand2=10, opcode=SUB, out=118
# PASS at time 8815: operand1=-107, operand2=7, opcode=DIV, out=-15
# PASS at time 8835: operand1=0, operand2=34, opcode=SUB, out=-34
# PASS at time 8855: operand1=0, operand2=-103, opcode=ADD, out=-103
# PASS at time 8875: operand1=-85, operand2=-8, opcode=ADD, out=-93
# PASS at time 8895: operand1=21, operand2=-7, opcode=ADD, out=14
# PASS at time 8915: operand1=22, operand2=-128, opcode=ADD, out=-106
# PASS at time 8935: operand1=-128, operand2=52, opcode=DIV, out=-2
# PASS at time 8955: operand1=-68, operand2=65, opcode=SUB, out=123
# PASS at time 8975: operand1=6, operand2=-70, opcode=SUB, out=76
# PASS at time 8995: operand1=-128, operand2=-121, opcode=SUB, out=-7
# PASS at time 9015: operand1=84, operand2=-34, opcode=MULT, out=-40
# PASS at time 9035: operand1=0, operand2=70, opcode=DIV, out=0
# PASS at time 9055: operand1=55, operand2=88, opcode=ADD, out=-113
# PASS at time 9075: operand1=108, operand2=-23, opcode=ADD, out=85
# PASS at time 9095: operand1=14, operand2=82, opcode=SUB, out=-68
# PASS at time 9115: operand1=-78, operand2=59, opcode=DIV, out=-1
# PASS at time 9135: operand1=-13, operand2=105, opcode=MULT, out=-85
# PASS at time 9155: operand1=-118, operand2=43, opcode=SUB, out=95
# PASS at time 9175: operand1=85, operand2=0, opcode=MULT, out=0
# PASS at time 9195: operand1=89, operand2=-86, opcode=DIV, out=-1
```

```
# PASS at time 9215: operand1=25, operand2=70, opcode=ADD, out=95
# PASS at time 9235: operand1=-127, operand2=93, opcode=MULT, out=-35
# PASS at time 9255: operand1=-128, operand2=-76, opcode=DIV, out=1
# PASS at time 9275: operand1=-32, operand2=-69, opcode=SUB, out=37
# PASS at time 9295: operand1=13, operand2=-65, opcode=SUB, out=78
# PASS at time 9315: operand1=48, operand2=-119, opcode=MULT, out=-80
# PASS at time 9335: operand1=95, operand2=-77, opcode=SUB, out=-84
# PASS at time 9355: operand1=55, operand2=125, opcode=ADD, out=-76
# PASS at time 9375: operand1=16, operand2=2, opcode=SUB, out=14
# PASS at time 9395: operand1=-49, operand2=-51, opcode=DIV, out=0
# PASS at time 9415: operand1=-128, operand2=70, opcode=ADD, out=-58
# PASS at time 9435: operand1=-107, operand2=-22, opcode=DIV, out=4
# PASS at time 9455: operand1=-112, operand2=-75, opcode=MULT, out=-48
# PASS at time 9475: operand1=93, operand2=-21, opcode=ADD, out=72
# PASS at time 9495: operand1=-128, operand2=-19, opcode=ADD, out=109
# PASS at time 9515: operand1=-47, operand2=5, opcode=SUB, out=-52
# PASS at time 9535: operand1=-128, operand2=-115, opcode=MULT, out=-128
# PASS at time 9555: operand1=0, operand2=-113, opcode=MULT, out=0
# PASS at time 9575: operand1=12, operand2=-128, opcode=SUB, out=-116
# PASS at time 9595: operand1=-40, operand2=0, opcode=SUB, out=-40
# PASS at time 9615: operand1=18, operand2=-45, opcode=MULT, out=-42
# PASS at time 9635: operand1=-35, operand2=70, opcode=SUB, out=-105
# PASS at time 9655: operand1=74, operand2=77, opcode=ADD, out=-105
# PASS at time 9675: operand1=0, operand2=-101, opcode=SUB, out=101
# PASS at time 9695: operand1=-30, operand2=54, opcode=MULT, out=-84
# PASS at time 9715: operand1=-77, operand2=72, opcode=DIV, out=-1
# PASS at time 9735: operand1=-128, operand2=31, opcode=SUB, out=97
# PASS at time 9755: operand1=74, operand2=1, opcode=ADD, out=75
# PASS at time 9775: operand1=0, operand2=127, opcode=SUB, out=-127
# PASS at time 9795: operand1=33, operand2=35, opcode=ADD, out=68
# PASS at time 9815: operand1=1, operand2=-28, opcode=MULT, out=-28
# PASS at time 9835: operand1=-19, operand2=-21, opcode=DIV, out=0
# PASS at time 9855: operand1=-95, operand2=-128, opcode=DIV, out=0
# PASS at time 9875: operand1=0, operand2=103, opcode=SUB, out=-103
# PASS at time 9895: operand1=17, operand2=101, opcode=SUB, out=-84
# PASS at time 9915: operand1=-90, operand2=-21, opcode=SUB, out=-69
# PASS at time 9935: operand1=58, operand2=0, opcode=DIV, out=0
# PASS at time 9955: operand1=-50, operand2=-79, opcode=MULT, out=110
# PASS at time 9975: operand1=0, operand2=69, opcode=SUB, out=-69
# PASS at time 9995: operand1=88, operand2=-41, opcode=DIV, out=-2
# PASS at time 10015: operand1=-117, operand2=-128, opcode=DIV, out=0
# Test completed with 0 errors
# ** Note: $finish    : alu_tb.sv(118)
#    Time: 10045 ns  Iteration: 0  Instance: /alu_tb
```

2 Q2: Counter

2.1 1. Testbench code

```
1  `timescale 1ns/1ps
2
3  //===============================================================
4  // Import the counter package containing parameters and classes
5  //===============================================================
6  import counter_pkg::*;  // Bring in our parameter + class
7
8  module counter_tb;
9
10   //===============================================================
11   // Local parameters: Override the default WIDTH if needed
12   //===============================================================
13   localparam int TB_WIDTH = WIDTH; // or you can set TB_WIDTH = 4, 6, or 8.
14
15   //===============================================================
16   // Testbench signals: Declare all signals used to interface with DUT
17   //===============================================================
18   bit clk;       // Clock signal
19   bit rst_n;     // Active-low reset
20   bit load_n;    // Load enable (active-low)
21   bit up_down;   // Direction control: 1 for up, 0 for down
22   bit ce;        // Count enable
23   logic [TB_WIDTH-1:0] data_load; // Data to load into counter
24   wire [TB_WIDTH-1:0] count_out;  // Counter output
25   wire max_count; // Flag indicating max count reached
26   wire zero;      // Flag indicating count is zero
27
28   //===============================================================
29   // Instantiate the DUT (Device Under Test) with overridden WIDTH
30   //===============================================================
31   counter #(.WIDTH(TB_WIDTH)) dut (
32     .clk      (clk),
33     .rst_n    (rst_n),
34     .load_n   (load_n),
35     .up_down  (up_down),
36     .ce       (ce),
37     .data_load(data_load),
38     .count_out(count_out),
39     .max_count(max_count),
40     .zero     (zero)
41   );
42
43   //===============================================================
44   // Create an instance of the random config class
45   //===============================================================
46   counter_cfg cfg;
```

```
47
48    //================================================================
49    // Reference model state (golden model for verification)
50    //================================================================
51    logic [TB_WIDTH-1:0] golden_count; // Reference count value
52    wire golden_max; // Reference max count flag
53    wire golden_zero; // Reference zero flag
54
55    //================================================================
56    // Clock generator: Generates a 10 ns period clock signal
57    //================================================================
58    initial begin
59      clk = 0;
60      forever begin
61          #5 clk = ~clk; // Toggle clock every 5ns
62          cfg.clk = clk;  // Update clock in config
63      end
64    end
65
66    //================================================================
67    // Task: synchronous reset assertion
68    //================================================================
69    task assert_reset();
70      begin
71        rst_n = 0;   // Activate reset (active-low)
72        @(posedge clk); // wait 1 clock cycle
73        @(posedge clk); // wait another clock cycle
74        rst_n = 1;   // Deactivate reset
75      end
76    endtask
77
78    //================================================================
79    // Golden model logic: Simulate expected counter behavior
80    //================================================================
81    always @(posedge clk) begin
82      if (!rst_n) begin
83        golden_count <= '0; // Reset golden count to zero
84      end else if (!load_n) begin
85        golden_count <= data_load; // Load value into golden count
86      end else if (ce) begin
87        if (up_down)
88          golden_count <= golden_count + 1; // Increment
89        else
90          golden_count <= golden_count - 1; // Decrement
91      end
92    end
93
94    // Calculate golden model outputs
95    assign golden_max  = (golden_count == {TB_WIDTH{1'b1}});
96    assign golden_zero = (golden_count == 0);
97
98    //================================================================
99    // Task: Compare DUT outputs with golden model
100   //================================================================
101   task check_result();
102     if (count_out !== golden_count) begin
103       $error("[CHECK_RESULT] Mismatch! DUT=%0d, GOLDEN=%0d", count_out, golden_count);
104     end
105     if (max_count !== golden_max) begin
106       $error("[CHECK_RESULT] max_count mismatch! DUT=%b, GOLDEN=%b", max_count, golden_max);
107     end
108     if (zero !== golden_zero) begin
109       $error("[CHECK_RESULT] zero mismatch! DUT=%b, GOLDEN=%b", zero, golden_zero);
110     end
111   endtask
112
113   //================================================================
114   // Main verification process: Drives the testbench flow
115   //================================================================
116   initial begin
117     // Create config object for randomization
118     cfg = new();
119
120     // 1) Assert reset
121     assert_reset();
122
123     // 2) Run random tests
124     for (int i = 0; i < 500; i++) begin
125       if (!cfg.randomize()) begin
126         $error("Randomization failed!");
127         $finish;
128       end
129
130       // Drive signals from random config
131       rst_n    = cfg.rst_n;
132       load_n   = cfg.load_n;
133       up_down  = cfg.up_down;
134       ce       = cfg.ce;
135       data_load= cfg.data_load;
136
137       @(posedge clk); // Wait for clock edge
138
139       // Compare DUT with golden model
140       check_result();
141
142       cfg.count_out = count_out;
143     end
144
145     $display("All done. End of simulation.");
146     $finish; // End simulation
```

```
  end

endmodule
```

2.2 2. Package code

```
package counter_pkg;

    //============================
    // 1) Declare the parameter
    //============================
    parameter int WIDTH = 4;

    //=====================================================
    // 2) Create a class for constrained-random stimulus
    //=====================================================
    class counter_cfg;

        //===================================
        // DUT signals we want to randomize
        //===================================
        bit             clk;            // clock signal
        rand bit        rst_n;          // Active-low reset
        rand bit        load_n;         // Active-low load
        rand bit        up_down;        // 1 => increment, 0 => decrement
        rand bit        ce;             // clock enable
        rand logic [WIDTH-1:0] data_load;

        logic [WIDTH-1:0] count_out;

        //====================
        // Coverage group
        //====================
        covergroup cg @(posedge clk);
          cp1: coverpoint rst_n;
          cp2: coverpoint load_n;
          cp3: coverpoint up_down;
          cp4: coverpoint ce;
          cp5: coverpoint data_load;

            // 1. Coverpoint for load data when load is asserted (active low) and reset is deasserted.
             cp_load_data: coverpoint data_load iff ((load_n == 0) && (rst_n == 1));

             // 2. Coverpoint for count out when reset is deasserted, enable is active and up_down is high.
             //    a. Auto-generate bins for all count values.
             // 3. And a transition bin to check for overflow (from maximum value to zero).
             cp_count_up_auto: coverpoint count_out iff ((rst_n == 1) && (ce == 1) && (up_down == 1));
          cp_count_up: coverpoint count_out iff ((rst_n == 1) && (ce == 1) && (up_down == 1)) {
             // Transition bin: detect when the count goes from the maximum value to zero.
             // (Using the transition operator "=>".)
             bins overflow = ( (2**WIDTH - 1) => 0 );
          }

             // 4. Coverpoint for count out when reset is deasserted, enable is active and up_down is low.
             //    a. Auto-generate bins for all count values.
             // 5. And a transition bin to check for underflow (from zero to maximum value).
             cp_count_down_auto: coverpoint count_out iff ((rst_n == 1) && (ce == 1) && (up_down == 0));
             cp_count_down: coverpoint count_out iff ((rst_n == 1) && (ce == 1) && (up_down == 0)) {
            // Transition bin: detect when the count goes from zero to the maximum value.
            bins underflow = ( 0 => (2**WIDTH - 1) );
             }

             endgroup


        //====================
        // Constructor
        //====================
        function new();
          cg = new();
        endfunction

        //================================================
        // 3) Constraints to meet the 70%/30% guidelines
        // use distribution for probability
        //================================================
        constraint reset_deactivated_most {
          // Reset low 30% of time, high 70%
          rst_n dist { 1 := 70, 0 := 30 };
        }

        constraint load_active_70 {
          // load_n=0 is "active" => 70% of time
          load_n dist { 0 := 70, 1 := 30 };
        }

        constraint enable_active_70 {
          // ce=1 => 70% of time
          ce dist { 1 := 70, 0 := 30 };
        }

        constraint up_down_dist {
          // Distribution constraint: 50% chance of 0, 50% chance of 1
          up_down dist { 0 :/ 50, 1 :/ 50 };
        }

        constraint up_down_data_load_c {
              if (up_down) {
            // up_down=1 => pick data_load mostly in lower half
            data_load dist {
```

```
              [0 : (1<<(WIDTH-1))-1] := 80,
              [(1<<(WIDTH-1)) : (1<<WIDTH)-1] := 20
            };
              } else {
            // up_down=0 => pick data_load mostly in upper half
            data_load dist {
              [0 : (1<<(WIDTH-1))-1] := 20,
              [(1<<(WIDTH-1)) : (1<<WIDTH)-1] := 80
            };
              }
        }

    endclass

endpackage
```

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

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

endmodule
```

2.4 4. Bug Fixes

missing (begin and end) for "else if (ce)"

2.5 5. Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
COUNTER_1	When reset (`rst_n`) is asserted, the output `count_out` should be zero.	Directed at the start of the simulation, followed by randomization with a constraint to keep reset active for 30% of the time.	Coverage point to track how many times reset is asserted and confirm `count_out == 0`.	A checker in the testbench ensures `count_out` is 0 when `rst_n == 0`.
COUNTER_2	When load (`load_n`) is asserted, `count_out` should take the value of `data_load`.	Randomization of `load_n` with a 70% probability of being active (low) and randomized `data_load` values.	Coverage point to track how many times `load_n` is asserted and the range of `data_load` values.	A checker in the testbench verifies `count_out == data_load` when `load_n == 0`.
COUNTER_3	When `ce` is enabled, the counter should increment or decrement based on `up_down`.	Randomization with 70% chance for `ce` being high, and a 50-50 distribution for `up_down`.	Coverage point to capture the toggling of `up_down` and transitions of `count_out`.	A checker compares `count_out` with the expected increment or decrement, verified against the golden model.
COUNTER_4	`max_count` should be asserted when `count_out` reaches its maximum value.	Random tests allowing the counter to reach its maximum possible value (`{WIDTH{1'b1}}`).	Coverage point to check how often `max_count` is triggered.	A checker validates `max_count == 1` when `count_out == max value`.
COUNTER_5	`zero` should be asserted when `count_out` is zero.	Directed reset tests and decrement tests pushing `count_out` to zero.	Coverage point for how often `zero` is asserted.	A checker verifies `zero == 1` only when `count_out == 0`.

Table 2: Verification Plan for Counter Design

2.6 6. Do File

```
vlib work
vlog counter_pkg.sv counter.v counter_tb.sv +cover -covercells
vsim -voptargs=+acc work.counter_tb -cover
add wave *
```

```
coverage save counter_tb.ucdb -onexit
run -all

# to run do file
#-- do run.txt
# to execute coverage report (one for code coverage and other fuctional coverage)
#-- vcover report counter_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=counter
#-- vcover report -details -cvg -output counter_coverage_report.txt counter_tb.ucdb
```

2.7 7. code Coverage Report

```
Coverage Report by DU with details

=================================================================================
=== Design Unit: work.counter
=================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        10        10         0   100.00%

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

Branch Coverage for Design Unit work.counter

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File counter.v
------------------------------------IF Branch------------------------------------
    21                                       501     Count coming in to IF
    21              1                        158         if (!rst_n)
    23              1                        242         else if (!load_n)
    25              1                         68         else if (ce) begin
                                              33     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    26                                        68     Count coming in to IF
    26              1                         32             if (up_down)
    28              1                         36             else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                       390     Count coming in to IF
    33              1                         39     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
    33              2                        351     assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    34                                       390     Count coming in to IF
    34              1                        115     assign zero = (count_out == 0)? 1:0;
    34              2                        275     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%

================================Condition Details================================

Condition Coverage for Design Unit work.counter --

  File counter.v
----------------Focused Condition View----------------
Line       33 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       34 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  -
```

Statement Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
Statements	7	7	0	100.00%

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

Statement Coverage for Design Unit work.counter —

Line	Item	Count	Source
File counter.v			
8			module counter (clk ,rst_n, load_n, up_down, ce, data_load, count_out, max_count, z
9			parameter WIDTH = 4;
10			input clk;
11			input rst_n;
12			input load_n;
13			input up_down;
14			input ce;
15			input [WIDTH-1:0] data_load;
16			output reg [WIDTH-1:0] count_out;
17			output max_count;
18			output zero;
19			
20	1	501	always @(posedge clk) begin
21			if (!rst_n)
22	1	158	count_out <= 0;
23			else if (!load_n)
24	1	242	count_out <= data_load;
25			else if (ce) begin
26			if (up_down)
27	1	32	count_out <= count_out + 1;
28			else
29	1	36	count_out <= count_out - 1;
30			end
31			end
32			
33	1	391	assign max_count = (count_out == {WIDTH{1'b1}})? 1:0;
34	1	391	assign zero = (count_out == 0)? 1:0;

Toggle Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
Toggles	30	30	0	100.00%

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

Toggle Coverage for Design Unit work.counter

Node	1H->0L	0L->1H	"Coverage"
ce	1	1	100.00
clk	1	1	100.00
count_out[0-3]	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 Design Unit (filtered view): 100.00%

2.8 8. functional coverage report

Coverage Report by instance with details

==== Instance: /counter_pkg
==== Design Unit: work.counter_pkg

Covergroup Coverage:

Covergroups	1	na	na	100.00%
Coverpoints/Crosses	10	na	na	na
Covergroup Bins	74	74	0	100.00%

Covergroup	Metric	Goal	Bins	Status
TYPE /counter_pkg/counter_cfg/cg	100.00%	100	-	Covered
covered/total bins:	74	74	-	
missing/total bins:	0	74	-	
% Hit:	100.00%	100	-	
Coverpoint cp1	100.00%	100	-	Covered

covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp5	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_load_data	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_up_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_up	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_down_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_down	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
Covergroup instance \/counter_pkg::counter_cfg::cg	100.00%	100	–	Covered
covered/total bins:	74	74	–	
missing/total bins:	0	74	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	159	1	–	Covered
bin auto[1]	343	1	–	Covered
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	353	1	–	Covered
bin auto[1]	149	1	–	Covered
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	273	1	–	Covered
bin auto[1]	229	1	–	Covered
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	146	1	–	Covered
bin auto[1]	356	1	–	Covered
Coverpoint cp5	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	35	1	–	Covered
bin auto[1]	25	1	–	Covered
bin auto[2]	28	1	–	Covered
bin auto[3]	34	1	–	Covered
bin auto[4]	24	1	–	Covered
bin auto[5]	30	1	–	Covered
bin auto[6]	27	1	–	Covered
bin auto[7]	24	1	–	Covered
bin auto[8]	25	1	–	Covered
bin auto[9]	42	1	–	Covered
bin auto[10]	27	1	–	Covered
bin auto[11]	41	1	–	Covered
bin auto[12]	29	1	–	Covered
bin auto[13]	42	1	–	Covered
bin auto[14]	25	1	–	Covered
bin auto[15]	42	1	–	Covered
Coverpoint cp_load_data	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	

bin auto[0]	17	1	–	Covered
bin auto[1]	9	1	–	Covered
bin auto[2]	15	1	–	Covered
bin auto[3]	20	1	–	Covered
bin auto[4]	10	1	–	Covered
bin auto[5]	14	1	–	Covered
bin auto[6]	15	1	–	Covered
bin auto[7]	11	1	–	Covered
bin auto[8]	10	1	–	Covered
bin auto[9]	18	1	–	Covered
bin auto[10]	12	1	–	Covered
bin auto[11]	20	1	–	Covered
bin auto[12]	17	1	–	Covered
bin auto[13]	15	1	–	Covered
bin auto[14]	14	1	–	Covered
bin auto[15]	25	1	–	Covered
Coverpoint cp_count_up_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	42	1	–	Covered
bin auto[1]	2	1	–	Covered
bin auto[2]	6	1	–	Covered
bin auto[3]	4	1	–	Covered
bin auto[4]	2	1	–	Covered
bin auto[5]	5	1	–	Covered
bin auto[6]	1	1	–	Covered
bin auto[7]	4	1	–	Covered
bin auto[8]	1	1	–	Covered
bin auto[9]	5	1	–	Covered
bin auto[10]	4	1	–	Covered
bin auto[11]	9	1	–	Covered
bin auto[12]	6	1	–	Covered
bin auto[13]	2	1	–	Covered
bin auto[14]	5	1	–	Covered
bin auto[15]	9	1	–	Covered
Coverpoint cp_count_up	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
bin overflow	6	1	–	Covered
Coverpoint cp_count_down_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	48	1	–	Covered
bin auto[1]	10	1	–	Covered
bin auto[2]	6	1	–	Covered
bin auto[3]	7	1	–	Covered
bin auto[4]	3	1	–	Covered
bin auto[5]	6	1	–	Covered
bin auto[6]	5	1	–	Covered
bin auto[7]	3	1	–	Covered
bin auto[8]	4	1	–	Covered
bin auto[9]	7	1	–	Covered
bin auto[10]	5	1	–	Covered
bin auto[11]	9	1	–	Covered
bin auto[12]	4	1	–	Covered
bin auto[13]	11	1	–	Covered
bin auto[14]	4	1	–	Covered
bin auto[15]	9	1	–	Covered
Coverpoint cp_count_down	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
bin underflow	3	1	–	Covered

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /counter_pkg/counter_cfg/cg	100.00%	100	–	Covered
covered/total bins:	74	74	–	
missing/total bins:	0	74	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	

Coverpoint cp5	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_load_data	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_up_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_up	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_down_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
Coverpoint cp_count_down	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
Covergroup instance \/counter_pkg::counter_cfg::cg				
	100.00%	100	–	Covered
covered/total bins:	74	74	–	
missing/total bins:	0	74	–	
% Hit:	100.00%	100	–	
Coverpoint cp1	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	159	1	–	Covered
bin auto[1]	343	1	–	Covered
Coverpoint cp2	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	353	1	–	Covered
bin auto[1]	149	1	–	Covered
Coverpoint cp3	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	273	1	–	Covered
bin auto[1]	229	1	–	Covered
Coverpoint cp4	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	146	1	–	Covered
bin auto[1]	356	1	–	Covered
Coverpoint cp5	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	35	1	–	Covered
bin auto[1]	25	1	–	Covered
bin auto[2]	28	1	–	Covered
bin auto[3]	34	1	–	Covered
bin auto[4]	24	1	–	Covered
bin auto[5]	30	1	–	Covered
bin auto[6]	27	1	–	Covered
bin auto[7]	24	1	–	Covered
bin auto[8]	25	1	–	Covered
bin auto[9]	42	1	–	Covered
bin auto[10]	27	1	–	Covered
bin auto[11]	41	1	–	Covered
bin auto[12]	29	1	–	Covered
bin auto[13]	42	1	–	Covered
bin auto[14]	25	1	–	Covered
bin auto[15]	42	1	–	Covered
Coverpoint cp_load_data	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	17	1	–	Covered
bin auto[1]	9	1	–	Covered
bin auto[2]	15	1	–	Covered
bin auto[3]	20	1	–	Covered
bin auto[4]	10	1	–	Covered
bin auto[5]	14	1	–	Covered
bin auto[6]	15	1	–	Covered
bin auto[7]	11	1	–	Covered
bin auto[8]	10	1	–	Covered
bin auto[9]	18	1	–	Covered
bin auto[10]	12	1	–	Covered
bin auto[11]	20	1	–	Covered
bin auto[12]	17	1	–	Covered
bin auto[13]	15	1	–	Covered
bin auto[14]	14	1	–	Covered

bin auto[15]	25	1	–	Covered
Coverpoint cp_count_up_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	42	1	–	Covered
bin auto[1]	2	1	–	Covered
bin auto[2]	6	1	–	Covered
bin auto[3]	4	1	–	Covered
bin auto[4]	2	1	–	Covered
bin auto[5]	5	1	–	Covered
bin auto[6]	1	1	–	Covered
bin auto[7]	4	1	–	Covered
bin auto[8]	1	1	–	Covered
bin auto[9]	5	1	–	Covered
bin auto[10]	4	1	–	Covered
bin auto[11]	9	1	–	Covered
bin auto[12]	6	1	–	Covered
bin auto[13]	2	1	–	Covered
bin auto[14]	5	1	–	Covered
bin auto[15]	9	1	–	Covered
Coverpoint cp_count_up	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
bin overflow	6	1	–	Covered
Coverpoint cp_count_down_auto	100.00%	100	–	Covered
covered/total bins:	16	16	–	
missing/total bins:	0	16	–	
% Hit:	100.00%	100	–	
bin auto[0]	48	1	–	Covered
bin auto[1]	10	1	–	Covered
bin auto[2]	6	1	–	Covered
bin auto[3]	7	1	–	Covered
bin auto[4]	3	1	–	Covered
bin auto[5]	6	1	–	Covered
bin auto[6]	5	1	–	Covered
bin auto[7]	3	1	–	Covered
bin auto[8]	4	1	–	Covered
bin auto[9]	7	1	–	Covered
bin auto[10]	5	1	–	Covered
bin auto[11]	9	1	–	Covered
bin auto[12]	4	1	–	Covered
bin auto[13]	11	1	–	Covered
bin auto[14]	4	1	–	Covered
bin auto[15]	9	1	–	Covered
Coverpoint cp_count_down	100.00%	100	–	Covered
covered/total bins:	1	1	–	
missing/total bins:	0	1	–	
% Hit:	100.00%	100	–	
bin underflow	3	1	–	Covered

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

2.9 9.Waveform

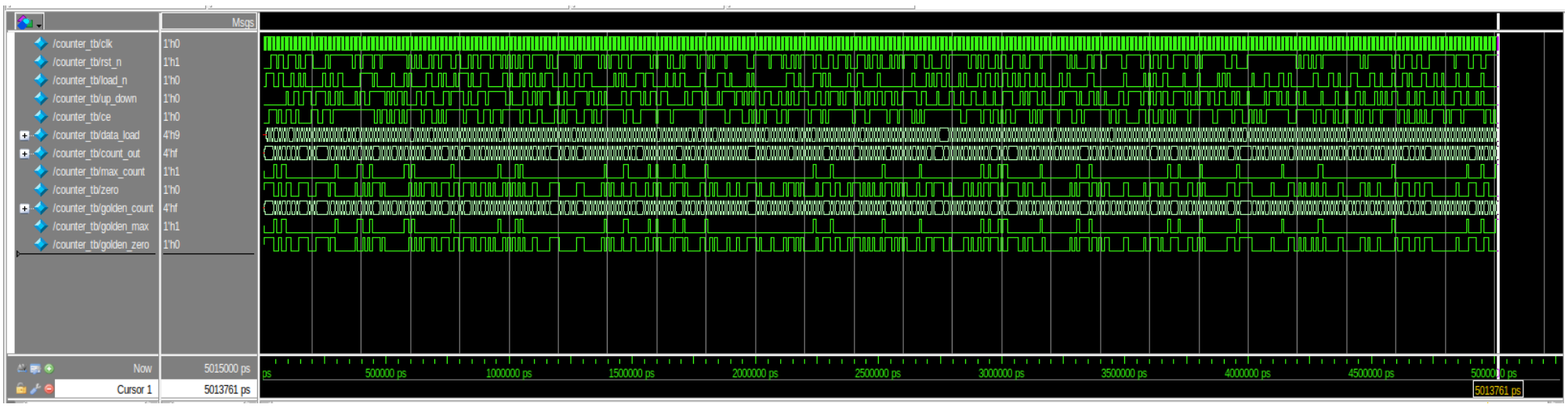

Figure 2: simulation waveform

```
# -- Compiling package counter_tb_sv_unit
# -- Importing package counter_pkg
# -- Compiling module counter_tb
#
# Top level modules:
#       counter_tb
# End time: 10:42:00 on Apr 01,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# Saving coverage database on exit...
# End time: 10:42:03 on Apr 01,2025, Elapsed time: 0:01:07
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.counter_tb -coverage
# Start time: 10:42:03 on Apr 01,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.counter_pkg(fast)
# Loading work.counter_tb_sv_unit(fast)
# Loading work.counter_tb(fast)
# Loading work.counter(fast)
# All done. End of simulation.
# ** Note: $finish    : counter_tb.sv(146)
#    Time: 5015 ns  Iteration: 1  Instance: /counter_tb
# 1
# Break in Module counter_tb at counter_tb.sv line 146
```

Figure 3: Transcript : all test cases passed

3 Q3: ALSU

3.1 1. Testbench code

```
`timescale 1ns/1ps

import ALSU_pkg::*;

module ALSU_tb;

  // ----------------------------------
  // Testbench signals
  // ----------------------------------
  logic clk;
  logic rst;
  logic cin;
  logic red_op_A;
  logic red_op_B;
  logic bypass_A;
  logic bypass_B;
  logic direction;
  logic serial_in;
  logic signed [2:0] A;
  logic signed [2:0] B;
  logic [2:0]        opcode;
  wire [15:0]        leds;
  wire signed [5:0]  out;

  // ----------------------------------
  // DUT instantiation
  // ----------------------------------
  ALSU #(
    .INPUT_PRIORITY("A"),
    .FULL_ADDER("ON")
  ) dut (
    .clk        (clk),
    .rst        (rst),
    .cin        (cin),
    .red_op_A   (red_op_A),
    .red_op_B   (red_op_B),
    .bypass_A   (bypass_A),
    .bypass_B   (bypass_B),
    .direction  (direction),
    .serial_in  (serial_in),
    .A          (A),
    .B          (B),
    .opcode     (opcode),
    .leds       (leds),
    .out        (out)
  );

  // ----------------------------------
  // Clock & reset generation
  // ----------------------------------
  initial begin
        clk = 0;
        forever begin
           #5 clk = ~clk;
        end
   end


  // ----------------------------------------
  // Create an object for random stimulus
  // ----------------------------------------
  alsu_rand_class rand_stim;

// ------------
  // reset task
  // ------------
  task do_reset();
    rst = 1;
    #10;
    // Check result against a golden model
      golden_model(
        rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
```

```
        direction, opcode
      );
    #10;
    rst = 0;
  endtask

  // ----------------------------------
  // Golden model for reference
  // ----------------------------------
task golden_model(
    input logic rst,
    input logic signed [2:0] A, B,
    input logic cin, serial_in, red_op_A, red_op_B,
    input logic bypass_A, bypass_B, direction,
    input logic [2:0] opcode
);

    logic signed [5:0] expected_out;
    logic [15:0] expected_leds;

    logic invalid_red_op, invalid_opcode, invalid;

    // Invalid condition handling
    invalid_red_op = (red_op_A | red_op_B) & (opcode[1] | opcode[2]);
    invalid_opcode = opcode[1] & opcode[2];
    invalid = invalid_red_op | invalid_opcode;

    if(rst) begin
       expected_out = 0;
       expected_leds = 0;
    end else begin
      if (invalid)
        expected_leds = ~expected_leds;
      else
        expected_leds = 0;
    end


    if (bypass_A && bypass_B)
    expected_out = ("A" == "A") ? A : B;  // INPUT_PRIORITY is "A"
    else if (bypass_A)
    expected_out = A;
    else if (bypass_B)
    expected_out = B;
    else if (invalid)
    expected_out = 0;
    else begin
    case (opcode)
        3'h0: begin // OR or Reduction OR
            if (red_op_A && red_op_B)
                expected_out = ("A" == "A") ? |A : |B;
            else if (red_op_A)
                expected_out = |A;
            else if (red_op_B)
                expected_out = |B;
            else
                expected_out = A | B;
        end
        3'h1: begin // XOR or Reduction XOR
            if (red_op_A && red_op_B)
                expected_out = ("A" == "A") ? ^A : ^B;
            else if (red_op_A)
                expected_out = ^A;
            else if (red_op_B)
                expected_out = ^B;
            else
                expected_out = A ^ B;
        end
        3'h2: expected_out = A + B; // ADD
        3'h3: expected_out = A * B; // MUL
        3'h4: begin // SHIFT
            if (direction)
                expected_out = {expected_out[4:0], serial_in};
            else
                expected_out = {serial_in, expected_out[5:1]};
        end
        3'h5: begin // ROTATE
            if (direction)
                expected_out = {expected_out[4:0], expected_out[5]};
            else
                expected_out = {expected_out[0], expected_out[5:1]};
        end
        default: expected_out = 0;
    endcase
    end

    // Wait another clock so the output is stable
    @(posedge clk);
    #1;

    if ( (out != expected_out) && (leds != expected_leds)) begin
        $error("[ALSU] Mismatch with golden model: opcode=%0b.  out=%0d, expected_out=%0d",
                opcode, out, expected_out);
    end
endtask

  // Variables for simulation control
  integer i, j;

  // Testbench stimulus generation
```

```
  initial begin
    // Initialize all inputs
    rst       = 1;
    cin       = 0;
    red_op_A  = 0;
    red_op_B  = 0;
    bypass_A  = 0;
    bypass_B  = 0;
    direction = 0;
    serial_in = 0;
    opcode    = 0;
    A         = 0;
    B         = 0;

    // Instantiate the random stimulus object
    rand_stim = new();

    // Hold reset for a few clock cycles
    rand_stim.stop();

    do_reset();  // start in reset

    rand_stim.start();

    // ============================================================
    // Phase 1: Full Constrained Randomization (Constraints 1-7 enabled)
    // Disable constraint 8 here
    // ============================================================
    rand_stim.disable_constrain_8();
    $display("Phase 1: Full constrained randomization with constraints 1-7");
    for (i = 0; i < 400; i = i + 1) begin

      if (!rand_stim.randomize()) begin
          $error("Randomization failed in phase 1 at iteration %0d", i);
          $finish;
      end

      @(negedge clk);

      // Drive DUT with randomized values
      cin       = rand_stim.cin;
      red_op_A  = rand_stim.red_op_A;
      red_op_B  = rand_stim.red_op_B;
      bypass_A  = rand_stim.bypass_A;
      bypass_B  = rand_stim.bypass_B;
      direction = rand_stim.direction;
      serial_in = rand_stim.serial_in;
      opcode    = rand_stim.opcode;
      A         = rand_stim.A;
      B         = rand_stim.B;

      // Wait a clock for inputs to be sampled
      @(posedge clk);


     // Check result against a golden model
     golden_model(
        rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
        direction, opcode
     );

      // Sample only when not in reset or bypass mode
      if (!(rst || bypass_A || bypass_B)) begin
        rand_stim.sample();
      end

    end

    $display("ALSU Phase 1 test completed");

    // ============================================================
    // Transition to Phase 2: Opcode Verification
    // ============================================================
    $display("Transitioning to Phase 2: Opcode verification");
    // Force key signals to 0 as required
    rst      = 0;
    bypass_A = 0;
    bypass_B = 0;
    red_op_A = 0;
    red_op_B = 0;

    // Disable all constraints for the random stimulus object
    // Enable constraint 8 (for unique opcode generation)
    rand_stim.disabled_all_constrains_expect_8();

    // Randomize other inputs once (without constraints)
    if (!rand_stim.randomize()) begin
      $error("Randomization without constraints failed in phase 2");
      $finish;
    end
    // Drive constant signals from the randomized object
    cin       = rand_stim.cin;
    direction = rand_stim.direction;
    serial_in = rand_stim.serial_in;
    A         = rand_stim.A;
    B         = rand_stim.B;

      // Sample only when not in reset or bypass mode
      if (!(rst || bypass_A || bypass_B)) begin
        rand_stim.sample();
```

```
    end


  // ============================================================
  // Phase 2: Nested Loop for Opcode Verification
  // ============================================================
  for (j = 0; j < 6; j = j + 1) begin

    if (!rand_stim.randomize()) begin
      $error("Randomization without constraints failed in phase 2");
      $finish;
    end

    @(negedge clk);
    opcode = rand_stim.opcode_array[j];
    rand_stim.opcode = rand_stim.opcode_array[j];

    // Drive the DUT with the current opcode while other inputs remain constant
      // Wait a clock for inputs to be sampled
      @(posedge clk);

      // Check result against a golden model
      golden_model(
         rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
         direction, opcode
      );

    // Sample only when not in reset or bypass mode
    if (!(rst || bypass_A || bypass_B)) begin
      rand_stim.sample();
    end

  end

    // ============================================================
    // End simulation and dump coverage data if needed
    // ============================================================
    $display("Testbench completed.");

    // direct rst value for 0 -> 1
    @(posedge clk);
    rst       = 1;
    #20;

    $finish;
  end

endmodule
```

3.2 2. Package code

```
package ALSU_pkg;
    // ----------------------------------------
    // 1) Define the opcodes (including invalids)
    // ----------------------------------------
    typedef enum logic [2:0] {
        OR_0       = 3'h0,  // 000
        XOR_1      = 3'h1,  // 001
        ADD_2      = 3'h2,  // 010
        MUL_3      = 3'h3,  // 011
        SHIFT_4    = 3'h4,  // 100
        ROTATE_5   = 3'h5,  // 101
        INVALID_6  = 3'h6,  // 110
        INVALID_7  = 3'h7   // 111
    } opcode_e;

    // ----------------------------------------
    // 2) 3-bit signed range is -4 .. +3
    // ----------------------------------------
    localparam logic signed [2:0] MAXNEG = -4;  // 3'b100
    localparam logic signed [2:0] ZERO   =  0;
    localparam logic signed [2:0] MAXPOS =  3;  // 3'b011

    class alsu_rand_class;
        // -------------------------------
        // Randomizable DUT inputs
        // -------------------------------
        bit                     clk;
        rand bit                rst;
        rand bit                cin;
        rand bit                red_op_A;
        rand bit                red_op_B;
        rand bit                bypass_A;
        rand bit                bypass_B;
        rand bit                direction;
        rand bit                serial_in;
        rand opcode_e           opcode;
        rand logic signed [2:0] A;
        rand logic signed [2:0] B;
        rand opcode_e           opcode_array[6];

        bit disable_all = 0;
        bit disable_8 = 0;


        // -------------------------------
        // Constraints from specification
        // -------------------------------

        // (a) Make RESET happen with a low probability
```

```
    constraint c_reset_low_prob {
        if (!disable_all) {
            rst dist { 0 := 95, 1 := 5 };
        }
    }

    // (b) For ADD or MUL, pick corner values of A,B more often
    //     (MAXNEG, ZERO, MAXPOS) than the other possibilities.
    //     Weighted distribution is used here.
    constraint c_adder_mult_corner {
        if (!disable_all) {
            if (opcode inside {ADD_2, MUL_3}) {
              A dist { MAXNEG := 3, ZERO := 3, MAXPOS := 3, [-3:-1] := 1, [1:2] := 1 };
              B dist { MAXNEG := 3, ZERO := 3, MAXPOS := 3, [-3:-1] := 1, [1:2] := 1 };
            }
        }
    }


    // (c) If opcode=OR or XOR and red_op_A=1, then A has exactly one bit set
    //     and B is 0 .
    constraint c_red_opA_onebit {
        if (!disable_all) {
            if ((opcode==OR_0 || opcode==XOR_1) && red_op_A==1) {
              // Force B to be 0 or near 0
              B == 0;
              // A has exactly 1 bit set in its 3 bits:
              A dist {1:=3, 2:=3, MAXNEG:=3 , MAXPOS := 1 , [-3:0] := 1};
            }
        }
    }

    // (d) Similarly, if opcode=OR or XOR and red_op_B=1, then B has exactly one bit set
    //     and A is 0.
    constraint c_red_opB_onebit {
        if (!disable_all) {
            if ((opcode==OR_0 || opcode==XOR_1) && red_op_B==1) {
              A == 0;
              B dist {1:=3, 2:=3, MAXNEG:=3 , MAXPOS := 1 , [-3:0]:= 1 };
            }
        }
    }

    // (e) Invalid cases (opcode=6 or 7, or red_op_X=1 for non-OR/XOR)
    //     should occur *less* frequently.
    //     Weighted distribution on opcode:
    constraint c_opcode_distribution {
        if (!disable_all) {
            opcode dist {
              INVALID_6  := 1,
              INVALID_7  := 1,
              OR_0       := 5,
              XOR_1      := 5,
              ADD_2      := 5,
              MUL_3      := 5,
              SHIFT_4    := 5,
              ROTATE_5   := 5
            };
        }
    }

    // (f) For red_op_A/B, require them to be 0 if opcode in {ADD_2, MUL_3, SHIFT_4, ROTATE_5}
    // except for a small chance to produce the invalid scenario:
    constraint c_red_op_non_orxor {
        if (!disable_all) {
            if (opcode inside {ADD_2, MUL_3, SHIFT_4, ROTATE_5}) {
              (red_op_A == 0) dist {0:=95, 1:=5};
              (red_op_B == 0) dist {0:=95, 1:=5};
            }
        }
    }

    constraint c_red_op_orxor {
        if (!disable_all) {
            if (opcode inside {XOR_1, OR_0}) {
                   {red_op_A, red_op_B} dist {2'b00 :/ 5, 2'b01 :/ 10, 2'b10 :/ 10 ,2'b11 :/ 75};
            }
        }
    }

    // (g) bypass_A and bypass_B should be disabled most of the time
    constraint c_bypass_dist {
        if (!disable_all) {
          bypass_A dist {0:=3,1:=1};
          bypass_B dist {0:=3,1:=1};
        }
    }

    // (h) If SHIFT or ROTATE, do not constrain A,B.
    //     (No explicit constraint needed => they can be anything.)


    // -------------------------------
    // Constraint 8
    // -------------------------------
    constraint c_opcode_unique {
        if (!disable_8) {
          foreach (opcode_array[i]) {
            opcode_array[i] == i;
          }
```

```
150             }
151         }
152 
153         // -------------------------------------
154         // Functional Coverage
155         // -------------------------------------
156         covergroup cg;
157           coverpoint rst;
158           coverpoint cin;
159           red_op_A_cp: coverpoint red_op_A{
160             bins red_op_A_0 = {0};
161             bins red_op_A_1 = {1};
162             bins red_op_A_default = default;
163           }
164           red_op_B_cp: coverpoint red_op_B{
165             bins red_op_B_0 = {0};
166             bins red_op_B_1 = {1};
167             bins red_op_B_default = default;
168           }
169           coverpoint bypass_A;
170           coverpoint bypass_B;
171           coverpoint direction;
172           coverpoint serial_in;
173           A_cp: coverpoint A {
174             bins A_data_0 = {0};
175             bins A_data_max = {MAXPOS};
176             bins A_data_min = {MAXNEG};
177             bins A_data_default = default;
178             bins A_data_walkingones[] = {3'b001, 3'b010, 3'b100};
179           }
180           B_cp: coverpoint B {
181             bins B_data_0 = {0};
182             bins B_data_max = {MAXPOS};
183             bins B_data_min = {MAXNEG};
184             bins B_data_default = default;
185             bins B_data_walkingones[] = {3'b001, 3'b010, 3'b100};
186           }
187           ALU_cp: coverpoint opcode {
188             bins Bins_shift[] = {SHIFT_4, ROTATE_5};
189             bins Bins_arith[] = {ADD_2, MUL_3};
190             bins Bins_bitwise[] = {OR_0, XOR_1};
191             bins Bins_invalid = {INVALID_6, INVALID_7};
192             bins Bins_trans = (OR_0 => XOR_1 => ADD_2 => MUL_3 => SHIFT_4 => ROTATE_5);
193           }
194 
195           cross A_cp , red_op_A_cp{
196            ignore_bins assert_red_op_A = binsof(A_cp.A_data_walkingones) && binsof(red_op_A_cp.red_op_A_1);
197           }
198 
199           cross B_cp , red_op_A_cp , red_op_B_cp{
200            ignore_bins assert_red_op_B = binsof(B_cp.B_data_walkingones) && binsof(red_op_A_cp.red_op_A_0) &&
                 binsof(red_op_B_cp.red_op_B_1) intersect {1};
201           }
202 
203         endgroup
204 
205 
206 
207         // constructor
208         function new();
209           cg = new();
210         endfunction
211 
212         //stop covergroup
213         function void stop();
214             cg.stop();
215         endfunction
216 
217         //start covergroup
218         function void start();
219             cg.start();
220         endfunction
221 
222         //sample covergroup
223         function void sample();
224             cg.sample();
225         endfunction
226 
227         // disabled all constrains expect 8
228         function void disabled_all_constrains_expect_8();
229           disable_all = 1;
230           disable_8 = 0 ;
231         endfunction
232 
233         // disable constrain 8
234         function void disable_constrain_8();
235           disable_all = 0;
236           disable_8 = 1 ;
237         endfunction
238 
239         // enable all constrains
240         function void enable_all_constrains();
241           disable_all = 0;
242           disable_8 = 0 ;
243         endfunction
244 
245 
246     endclass
247 
248 
```

```
endpackage
```

3.3 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, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
input [2:0] opcode;
input signed [2:0] A, B;
output reg [15:0] leds;
output reg signed [5:0] out;

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

wire invalid_red_op, invalid_opcode, invalid;

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

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

//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)
          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 <= 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
        endcase
    end
  end
end

endmodule
```

3.4 4. Bug Fixes

no bugs except cin_reg is one bit not two bits

3.5 5. Verification Plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
ALSU_1	When the reset is asserted, the outputs should be low.	• Directed reset is applied at simulation start via the `do_reset()` task. • Afterwards, the reset signal is randomized with the constraint `c_reset_low_prob` (5% chance of `rst`=1).	• A covergroup in `alsu_rand_class` monitors transitions on `rst`. • Coverage bins ensure that enough reset assertions are observed during randomization.	• The golden model is invoked during reset to verify that both `out` and `leds` are 0. • Any deviation (non-zero outputs when `rst` is high) flags an error.
ALSU_2	In the absence of invalid conditions, when the opcode is ADD, the output should perform addition on ports A and B, incorporating `cin` if `FULL_ADDER` is enabled.	• The random stimulus is generated with the weighted constraint `c_opcode_distribution` to ensure frequent selection of ADD (3'h2). • Inputs A and B are randomized using `c_adder_mult_corner` to emphasize corner cases (values `MAXNEG`, `ZERO`, `MAXPOS`). • Reduction control signals (`red_op_A` and `red_op_B`) are mostly deasserted to avoid invalid conditions.	• The covergroup within `alsu_rand_class` collects data on opcode, A, B, and `cin` along with other control signals. • Specific bins track the occurrence of corner-case operand values and the frequency of the ADD opcode.	• After each randomized transaction, the `golden_model()` task computes the expected output (i.e. `A+B` plus `cin` when relevant). • The testbench compares the DUT output against the golden model; any mismatch in the computed sum flags an error.

Table 3: Verification Plan

3.6 6. Do File

```
vlib work
vlog ALSU.sv ALSU_tb.sv ALSU_pkg.sv +cover -covercells
vsim -voptargs=+acc work.ALSU_tb -cover
add wave *
coverage save ALSU_tb.ucdb -onexit
run -all

# to run do file
#-- do run.txt
#to execute coverage report
#-- vcover report ALSU_tb.ucdb -details -annotate -all -output coverage_rpt.txt -du=ALSU
#-- vcover report -details -cvg -output ALSU_coverage_report.txt ALSU_tb.ucdb
```

3.7 7. functional Coverage Report

```
Coverage Report by DU with details

================================================================================
=== Design Unit: work.ALSU
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        32        32         0   100.00%

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

Branch Coverage for Design Unit work.ALSU

    Line         Item                           Count     Source
    ----         ----                           -----     ------
  File ALSU.sv
----------------------------------IF Branch-----------------------------------
```

```
    24                                  816     Count coming in to IF
    24              1                     4       if(rst) begin
    35              1                   812       end else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    51                                  819     Count coming in to IF
    51              1                     7       if(rst) begin
    53              1                   812       end else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    54                                  812     Count coming in to IF
    54              1                   622           if (invalid)
    56              1                   190           else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    63                                  793     Count coming in to IF
    63              1                     4       if(rst) begin
    66              1                   789       else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    67                                  789     Count coming in to IF
    67              1                    44         if (bypass_A_reg && bypass_B_reg)
    69              1                   167         else if (bypass_A_reg)
    71              1                   138         else if (bypass_B_reg)
    73              1                   326         else if (invalid)
    75              1                   114         else begin
Branch totals: 5 hits of 5 branches = 100.00%

------------------------------------CASE Branch------------------------------------
    76                                  114     Count coming in to CASE
    77              1                    31               3'h0: begin
    87              1                    32               3'h1: begin
    97              1                    10               3'h2: out <= A_reg + B_reg;
    98              1                    17               3'h3: out <= A_reg * B_reg;
    99              1                    15               3'h4: begin
    105             1                     8               3'h5: begin
                                          1     All False Count
Branch totals: 7 hits of 7 branches = 100.00%

------------------------------------IF Branch------------------------------------
    78                                   31     Count coming in to IF
    78              1                     5               if (red_op_A_reg && red_op_B_reg)
    80              1                     7               else if (red_op_A_reg)
    82              1                    10               else if (red_op_B_reg)
    84              1                     9               else
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    88                                   32     Count coming in to IF
    88              1                     5               if (red_op_A_reg && red_op_B_reg)
    90              1                     7               else if (red_op_A_reg)
    92              1                    12               else if (red_op_B_reg)
    94              1                     8               else
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    100                                  15     Count coming in to IF
    100             1                    11               if (direction_reg)
    102             1                     4               else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    106                                   8     Count coming in to IF
    106             1                     5               if (direction_reg)
    108             1                     3               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 Design Unit work.ALSU —

  File ALSU.sv
----------Focused Condition View----------
Line       67 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)
```

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 78 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 88 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 Design Unit work.ALSU —

File ALSU.sv

Focused Expression View

Line 18 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 19 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 20 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 Design Unit work.ALSU —

Line	Item	Count	Source
File ALSU.sv			
1			module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, c
2			parameter INPUT_PRIORITY = "A";
3			parameter FULL_ADDER = "ON";
4			input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
5			input [2:0] opcode;
6			input signed [2:0] A, B;
7			output reg [15:0] leds;
8			output reg signed [5:0] out;
9			
10			reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in
11			reg signed cin_reg;
12			reg [2:0] opcode_reg;
13			reg signed [2:0] A_reg, B_reg;
14			
15			wire invalid_red_op, invalid_opcode, invalid;
16			
17			//Invalid handling
18	1	372	assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg
19	1	357	assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
20	1	185	assign invalid = invalid_red_op | invalid_opcode;
21			
22			//Registering input signals
23	1	816	always @(posedge clk or posedge rst) begin
24			if(rst) begin
25	1	4	cin_reg <= 0;
26	1	4	red_op_B_reg <= 0;
27	1	4	red_op_A_reg <= 0;
28	1	4	bypass_B_reg <= 0;
29	1	4	bypass_A_reg <= 0;
30	1	4	direction_reg <= 0;
31	1	4	serial_in_reg <= 0;
32	1	4	opcode_reg <= 0;
33	1	4	A_reg <= 0;
34	1	4	B_reg <= 0;
35			end else begin
36	1	812	cin_reg <= cin;
37	1	812	red_op_B_reg <= red_op_B;
38	1	812	red_op_A_reg <= red_op_A;
39	1	812	bypass_B_reg <= bypass_B;
40	1	812	bypass_A_reg <= bypass_A;
41	1	812	direction_reg <= direction;
42	1	812	serial_in_reg <= serial_in;
43	1	812	opcode_reg <= opcode;
44	1	812	A_reg <= A;
45	1	812	B_reg <= B;
46			end
47			end
48			
49			//leds output blinking
50	1	819	always @(posedge clk or posedge rst) begin
51			if(rst) begin
52	1	7	leds <= 0;
53			end else begin
54			if (invalid)
55	1	622	leds <= ~leds;
56			else
57	1	190	leds <= 0;
58			end
59			end
60			
61			//ALSU output processing
62	1	793	always @(posedge clk or posedge rst) begin
63			if(rst) begin
64	1	4	out <= 0;
65			end
66			else begin
67			if (bypass_A_reg && bypass_B_reg)
68	1	44	out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;

```
69                                          else if (bypass_A_reg)
70              1                   167       out <= A_reg;
71                                          else if (bypass_B_reg)
72              1                   138       out <= B_reg;
73                                          else if (invalid)
74              1                   326         out <= 0;
75                                          else begin
76                                            case (opcode)
77                                              3'h0: begin
78                                                if (red_op_A_reg && red_op_B_reg)
79              1                     5           out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
80                                                else if (red_op_A_reg)
81              1                     7           out <= |A_reg;
82                                                else if (red_op_B_reg)
83              1                    10           out <= |B_reg;
84                                                else
85              1                     9           out <= A_reg | B_reg;
86                                              end
87                                              3'h1: begin
88                                                if (red_op_A_reg && red_op_B_reg)
89              1                     5           out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
90                                                else if (red_op_A_reg)
91              1                     7           out <= ^A_reg;
92                                                else if (red_op_B_reg)
93              1                    12           out <= ^B_reg;
94                                                else
95              1                     8           out <= A_reg ^ B_reg;
96                                              end
97              1                    10         3'h2: out <= A_reg + B_reg;
98              1                    17         3'h3: out <= A_reg * B_reg;
99                                              3'h4: begin
100                                               if (direction_reg)
101             1                    11           out <= {out[4:0], serial_in_reg};
102                                               else
103             1                     4           out <= {serial_in_reg, out[5:1]};
104                                           end
105                                           3'h5: begin
106                                             if (direction_reg)
107             1                     5         out <= {out[4:0], out[5]};
108                                             else
109             1                     3         out <= {out[0], out[5:1]};
```

Toggle Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
Toggles	118	118	0	100.00%

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

Toggle Coverage for Design Unit work.ALSU

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

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

Toggle Coverage         =     100.00% (118 of 118 bins)


Total Coverage By Design Unit (filtered view): 100.00%
```

3.8 8. code Coverage Report

Coverage Report by instance with details

```
=================================================================================
=== Instance: /ALSU_pkg
=== Design Unit: work.ALSU_pkg
=================================================================================

Covergroup Coverage:
    Covergroups                 1        na        na   100.00%
        Coverpoints/Crosses    13        na        na        na
            Covergroup Bins    60        60         0   100.00%
```

Covergroup	Metric	Goal	Bins	Status
TYPE /ALSU_pkg/alsu_rand_class/cg	100.00%	100	–	Covered
covered/total bins:	60	60	–	
missing/total bins:	0	60	–	
% Hit:	100.00%	100	–	
Coverpoint rst	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cin	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint red_op_A_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint red_op_B_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint bypass_A	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint bypass_B	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint direction	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint serial_in	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint A_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
Coverpoint B_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% 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	–	
Cross #cross__0#	100.00%	100	–	Covered
covered/total bins:	8	8	–	
missing/total bins:	0	8	–	
% Hit:	100.00%	100	–	
Cross #cross__1#	100.00%	100	–	Covered
covered/total bins:	18	18	–	
missing/total bins:	0	18	–	
% Hit:	100.00%	100	–	
Covergroup instance \/ALSU_pkg::alsu_rand_class::cg	100.00%	100	–	Covered
covered/total bins:	60	60	–	
missing/total bins:	0	60	–	
% Hit:	100.00%	100	–	
Coverpoint rst	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	214	1	–	Covered
bin auto[1]	15	1	–	Covered
Coverpoint cin	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	108	1	–	Covered
bin auto[1]	121	1	–	Covered
Coverpoint red_op_A_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	

% Hit:	100.00%	100	–	
bin red_op_A_0	43	1	–	Covered
bin red_op_A_1	186	1	–	Covered
default bin red_op_A_default	0		–	ZERO
Coverpoint red_op_B_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin red_op_B_0	40	1	–	Covered
bin red_op_B_1	189	1	–	Covered
default bin red_op_B_default	0		–	ZERO
Coverpoint bypass_A	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	225	1	–	Covered
bin auto[1]	4	1	–	Covered
Coverpoint bypass_B	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	224	1	–	Covered
bin auto[1]	5	1	–	Covered
Coverpoint direction	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	115	1	–	Covered
bin auto[1]	114	1	–	Covered
Coverpoint serial_in	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	113	1	–	Covered
bin auto[1]	116	1	–	Covered
Coverpoint A_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
bin A_data_0	57	1	–	Covered
bin A_data_max	28	1	–	Covered
bin A_data_min	28	1	–	Covered
bin A_data_walkingones[1]	25	1	–	Covered
bin A_data_walkingones[2]	30	1	–	Covered
default bin A_data_default	61		–	Occurred
Coverpoint B_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
bin B_data_0	45	1	–	Covered
bin B_data_max	35	1	–	Covered
bin B_data_min	31	1	–	Covered
bin B_data_walkingones[1]	28	1	–	Covered
bin B_data_walkingones[2]	28	1	–	Covered
default bin B_data_default	62		–	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_4]	40	1	–	Covered
bin Bins_shift[ROTATE_5]	41	1	–	Covered
bin Bins_arith[ADD_2]	39	1	–	Covered
bin Bins_arith[MUL_3]	46	1	–	Covered
bin Bins_bitwise[OR_0]	25	1	–	Covered
bin Bins_bitwise[XOR_1]	28	1	–	Covered
bin Bins_invalid	10	1	–	Covered
bin Bins_trans	1	1	–	Covered
Cross #cross__0#	100.00%	100	–	Covered
covered/total bins:	8	8	–	
missing/total bins:	0	8	–	
% Hit:	100.00%	100	–	
Auto, Default and User Defined Bins:				
bin <A_data_walkingones[2],red_op_A_0>	4	1	–	Covered
bin <A_data_walkingones[1],red_op_A_0>	2	1	–	Covered
bin <A_data_min,red_op_A_1>	24	1	–	Covered
bin <A_data_min,red_op_A_0>	4	1	–	Covered
bin <A_data_max,red_op_A_1>	26	1	–	Covered
bin <A_data_0,red_op_A_1>	33	1	–	Covered
bin <A_data_max,red_op_A_0>	2	1	–	Covered
bin <A_data_0,red_op_A_0>	24	1	–	Covered
Illegal and Ignore Bins:				
ignore_bin assert_red_op_A	49		–	Occurred
Cross #cross__1#	100.00%	100	–	Covered
covered/total bins:	18	18	–	
missing/total bins:	0	18	–	
% Hit:	100.00%	100	–	
Auto, Default and User Defined Bins:				
bin <B_data_walkingones[2],red_op_A_1,red_op_B_1>	18	1	–	Covered
bin <B_data_walkingones[2],red_op_A_1,red_op_B_0>	3	1	–	Covered
bin <B_data_walkingones[1],red_op_A_1,red_op_B_1>				

	17	1	–	Covered
bin <B_data_walkingones[1],red_op_A_1,red_op_B_0>	2	1	–	Covered
bin <B_data_min,red_op_A_1,red_op_B_1>	22	1	–	Covered
bin <B_data_min,red_op_A_1,red_op_B_0>	2	1	–	Covered
bin <B_data_max,red_op_A_1,red_op_B_1>	28	1	–	Covered
bin <B_data_0,red_op_A_1,red_op_B_1>	28	1	–	Covered
bin <B_data_max,red_op_A_1,red_op_B_0>	1	1	–	Covered
bin <B_data_0,red_op_A_1,red_op_B_0>	14	1	–	Covered
bin <B_data_walkingones[2],red_op_A_0,red_op_B_0>	1	1	–	Covered
bin <B_data_walkingones[1],red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_min,red_op_A_0,red_op_B_1>	5	1	–	Covered
bin <B_data_min,red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_max,red_op_A_0,red_op_B_1>	4	1	–	Covered
bin <B_data_0,red_op_A_0,red_op_B_1>	2	1	–	Covered
bin <B_data_max,red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_0,red_op_A_0,red_op_B_0>	1	1	–	Covered
Illegal and Ignore Bins:				
ignore_bin assert_red_op_B	13		–	Occurred

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /ALSU_pkg/alsu_rand_class/cg	100.00%	100	–	Covered
covered/total bins:	60	60	–	
missing/total bins:	0	60	–	
% Hit:	100.00%	100	–	
Coverpoint rst	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint cin	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint red_op_A_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint red_op_B_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint bypass_A	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint bypass_B	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint direction	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint serial_in	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
Coverpoint A_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
Coverpoint B_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% 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	–	
Cross #cross__0#	100.00%	100	–	Covered
covered/total bins:	8	8	–	
missing/total bins:	0	8	–	
% Hit:	100.00%	100	–	
Cross #cross__1#	100.00%	100	–	Covered
covered/total bins:	18	18	–	
missing/total bins:	0	18	–	
% Hit:	100.00%	100	–	
Covergroup instance \/ALSU_pkg::alsu_rand_class::cg	100.00%	100	–	Covered
covered/total bins:	60	60	–	
missing/total bins:	0	60	–	
% Hit:	100.00%	100	–	
Coverpoint rst	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	

bin auto[0]	214	1	–	Covered
bin auto[1]	15	1	–	Covered
Coverpoint cin	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	108	1	–	Covered
bin auto[1]	121	1	–	Covered
Coverpoint red_op_A_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin red_op_A_0	43	1	–	Covered
bin red_op_A_1	186	1	–	Covered
default bin red_op_A_default	0		–	ZERO
Coverpoint red_op_B_cp	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin red_op_B_0	40	1	–	Covered
bin red_op_B_1	189	1	–	Covered
default bin red_op_B_default	0		–	ZERO
Coverpoint bypass_A	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	225	1	–	Covered
bin auto[1]	4	1	–	Covered
Coverpoint bypass_B	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	224	1	–	Covered
bin auto[1]	5	1	–	Covered
Coverpoint direction	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	115	1	–	Covered
bin auto[1]	114	1	–	Covered
Coverpoint serial_in	100.00%	100	–	Covered
covered/total bins:	2	2	–	
missing/total bins:	0	2	–	
% Hit:	100.00%	100	–	
bin auto[0]	113	1	–	Covered
bin auto[1]	116	1	–	Covered
Coverpoint A_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
bin A_data_0	57	1	–	Covered
bin A_data_max	28	1	–	Covered
bin A_data_min	28	1	–	Covered
bin A_data_walkingones[1]	25	1	–	Covered
bin A_data_walkingones[2]	30	1	–	Covered
default bin A_data_default	61		–	Occurred
Coverpoint B_cp	100.00%	100	–	Covered
covered/total bins:	5	5	–	
missing/total bins:	0	5	–	
% Hit:	100.00%	100	–	
bin B_data_0	45	1	–	Covered
bin B_data_max	35	1	–	Covered
bin B_data_min	31	1	–	Covered
bin B_data_walkingones[1]	28	1	–	Covered
bin B_data_walkingones[2]	28	1	–	Covered
default bin B_data_default	62		–	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_4]	40	1	–	Covered
bin Bins_shift[ROTATE_5]	41	1	–	Covered
bin Bins_arith[ADD_2]	39	1	–	Covered
bin Bins_arith[MUL_3]	46	1	–	Covered
bin Bins_bitwise[OR_0]	25	1	–	Covered
bin Bins_bitwise[XOR_1]	28	1	–	Covered
bin Bins_invalid	10	1	–	Covered
bin Bins_trans	1	1	–	Covered
Cross #cross__0#	100.00%	100	–	Covered
covered/total bins:	8	8	–	
missing/total bins:	0	8	–	
% Hit:	100.00%	100	–	
Auto, Default and User Defined Bins:				
bin <A_data_walkingones[2],red_op_A_0>	4	1	–	Covered
bin <A_data_walkingones[1],red_op_A_0>	2	1	–	Covered
bin <A_data_min,red_op_A_1>	24	1	–	Covered
bin <A_data_min,red_op_A_0>	4	1	–	Covered
bin <A_data_max,red_op_A_1>	26	1	–	Covered
bin <A_data_0,red_op_A_1>	33	1	–	Covered
bin <A_data_max,red_op_A_0>	2	1	–	Covered
bin <A_data_0,red_op_A_0>	24	1	–	Covered
Illegal and Ignore Bins:				

ignore_bin assert_red_op_A	49		–	Occurred
Cross #cross__1#	100.00%	100	–	Covered
covered/total bins:	18	18	–	
missing/total bins:	0	18	–	
% Hit:	100.00%	100	–	
Auto, Default and User Defined Bins:				
bin <B_data_walkingones[2],red_op_A_1,red_op_B_1>	18	1	–	Covered
bin <B_data_walkingones[2],red_op_A_1,red_op_B_0>	3	1	–	Covered
bin <B_data_walkingones[1],red_op_A_1,red_op_B_1>	17	1	–	Covered
bin <B_data_walkingones[1],red_op_A_1,red_op_B_0>	2	1	–	Covered
bin <B_data_min,red_op_A_1,red_op_B_1>	22	1	–	Covered
bin <B_data_min,red_op_A_1,red_op_B_0>	2	1	–	Covered
bin <B_data_max,red_op_A_1,red_op_B_1>	28	1	–	Covered
bin <B_data_0,red_op_A_1,red_op_B_1>	28	1	–	Covered
bin <B_data_max,red_op_A_1,red_op_B_0>	1	1	–	Covered
bin <B_data_0,red_op_A_1,red_op_B_0>	14	1	–	Covered
bin <B_data_walkingones[2],red_op_A_0,red_op_B_0>	1	1	–	Covered
bin <B_data_walkingones[1],red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_min,red_op_A_0,red_op_B_1>	5	1	–	Covered
bin <B_data_min,red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_max,red_op_A_0,red_op_B_1>	4	1	–	Covered
bin <B_data_0,red_op_A_0,red_op_B_1>	2	1	–	Covered
bin <B_data_max,red_op_A_0,red_op_B_0>	2	1	–	Covered
bin <B_data_0,red_op_A_0,red_op_B_0>	1	1	–	Covered
Illegal and Ignore Bins:				
ignore_bin assert_red_op_B	13		–	Occurred

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

3.9 9.Waveform

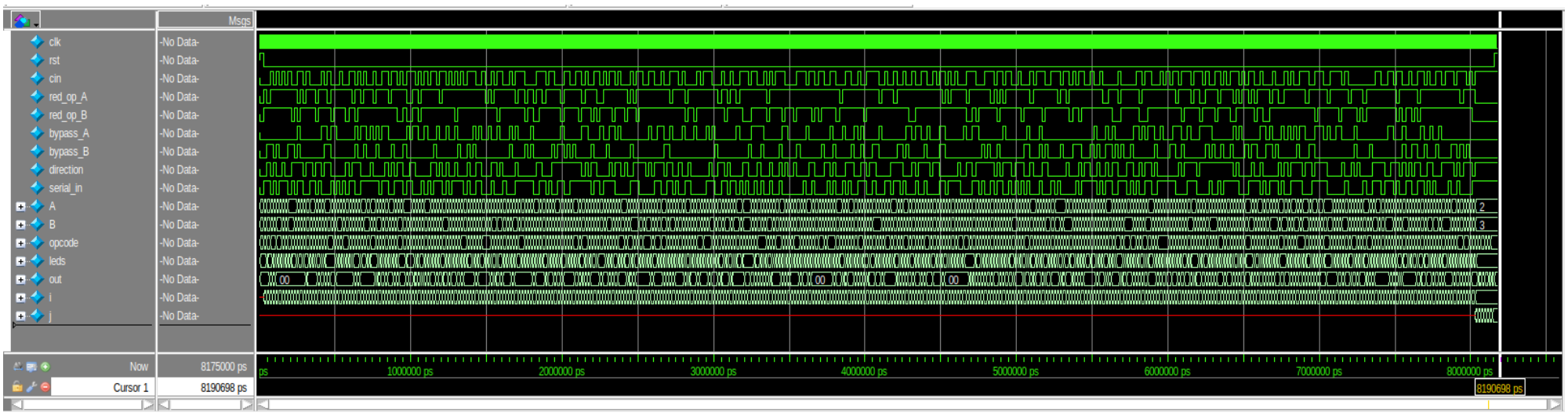

Figure 4: simulation waveform

```
VSIM(paused)> do run.do
# ** Warning: (vlib-34) Library already exists at "work".
# Errors: 0, Warnings: 1
# QuestaSim-64 vlog 2021.2_1 Compiler 2021.05 May 15 2021
# Start time: 06:04:48 on Apr 01,2025
# vlog -reportprogress 300 ALSU.sv ALSU_tb.sv ALSU_pkg.sv "+cover" -covercells
# -- Compiling module ALSU
# -- Compiling package ALSU_tb_sv_unit
# -- Importing package ALSU_pkg
# -- Compiling module ALSU_tb
# -- Compiling package ALSU_pkg
#
# Top level modules:
#       ALSU_tb
# End time: 06:04:48 on Apr 01,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
# vsim -voptargs="+acc" work.ALSU_tb -coverage
# Start time: 06:04:48 on Apr 01,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.ALSU_pkg(fast)
# Loading work.ALSU_tb_sv_unit(fast)
# Loading work.ALSU_tb(fast)
# Loading work.ALSU(fast)
# ** Warning: (vsim-8474) A higher value '4' is found in bin 'A_data_walkingones' of Coverpoint 'A_cp'. It is invalid and will be ignored.
#    Time: 0 ps  Iteration: 0  Process: /ALSU_tb/#INITIAL#173 File: ALSU_pkg.sv Line: 178
# ** Warning: (vsim-8474) A higher value '4' is found in bin 'B_data_walkingones' of Coverpoint 'B_cp'. It is invalid and will be ignored.
#    Time: 0 ps  Iteration: 0  Process: /ALSU_tb/#INITIAL#173 File: ALSU_pkg.sv Line: 185
# Phase 1: Full constrained randomization with constraints 1-7
# ALSU Phase 1 test completed
# Transitioning to Phase 2: Opcode verification
# Testbench completed.
# ** Note: $finish    : ALSU_tb.sv(315)
#    Time: 8175 ns  Iteration: 0  Instance: /ALSU_tb
# 1
# Break in Module ALSU_tb at ALSU_tb.sv line 315
```

Figure 5: Transcript : all test cases passed

4 Q4: my_mem

4.1 1. Testbench code

```
module tb_my_mem;
  //------------------------------------------------------------------------
  // 1. Setup and Data Structures
  //------------------------------------------------------------------------
  localparam int TESTS = 100;

  // DUT signals
  logic        clk;
  logic        write;
  logic        read;
  logic [7:0]  data_in;
  logic [15:0] address;
  logic [7:0]   data_out;

  // Instantiate the DUT (assuming my_mem is defined in another file)
  my_mem dut (
    .clk(clk),
    .write(write),
    .read(read),
    .data_in(data_in),
    .address(address),
    .data_out(data_out)
  );

  // Dynamic arrays for stimulus generation
  int      address_array[];          // Stores random addresses
  byte     data_to_write_array[];    // Stores random data values

  // Associative array for expected results, indexed by address (key type: int)
  logic [8:0] data_read_expect_assoc [int];

  // Queue to store the actual data read from the DUT
  byte data_read_queue[$];

  // Counters for self-checking
  int error_count = 0;
  int correct_count = 0;

  //------------------------------------------------------------------------
  // Clock Generation
  //------------------------------------------------------------------------
  initial begin
    clk = 0;
    forever #5 clk = ~clk; // 10 time units clock period
  end

  //------------------------------------------------------------------------
  // 2. Task Implementations
  //------------------------------------------------------------------------
  // Task: stimulus_gen
  // Generates random addresses and data values TESTS times.
  task stimulus_gen;
    int i;
    for (i = 0; i < TESTS; i++) begin
      address_array[i] = $urandom_range(0, 65535);  // 16-bit address space
      data_to_write_array[i] = $urandom();          // 8-bit random data
    end
  endtask
```

```

  // Task: golden_model
  // Populates the expected data associative array using the addresses and data values.
  task golden_model;
    int i;
    for (i = 0; i < TESTS; i++) begin
      // Calculate even parity using ~^ operator; expected memory stored as {parity, data}
      data_read_expect_assoc[address_array[i]] = {~^data_to_write_array[i], data_to_write_array[i]};
    end
  endtask

  // Task: check9Bits
  // Checks if the lower 8 bits of the stored data in the DUT match the expected data.
  task check9Bits(input int addr);
    if (data_out !== data_read_expect_assoc[addr][7:0] && dut.mem_array[addr][9] !== data_read_expect_assoc[addr][7:0]) begin
      error_count++;
      $stop;
    end else
      correct_count++;
  endtask

  //------------------------------------------------------------------------
  // 3. Initial Block Implementation
  //------------------------------------------------------------------------
  initial begin
    // Allocate dynamic arrays for stimulus generation
    address_array = new[TESTS];
    data_to_write_array = new[TESTS];

    // 3.1 Data preparation: generate stimulus and expected results.
    stimulus_gen();
    golden_model();

    // 3.2 Write Operations:
    // Drive write operations on the negative edge of the clock.
    write = 1;
    read  = 0;
    for (int i = 0; i < TESTS; i++) begin
      address = address_array[i];
      data_in = data_to_write_array[i];
      @(negedge clk);
    end
    write = 0;

    // 3.4 Read and Self-Checking:
    // loop through the associative array keys == loops over TESTS.
    foreach (data_read_expect_assoc[addr]) begin
      @(negedge clk);
      address = addr;       // Drive the address port with the stored address
      read = 1;
      @(negedge clk);       // Wait for read to take effect on positive edge

      if (data_read_expect_assoc.exists(addr)) begin
        // Call the check task comparing data read vs. expected
        check9Bits(addr);
      end

      // Store the read data into the queue
      data_read_queue.push_back(data_out);
    end
    read = 0;

    // 3.5 Test Completion and Reporting:
    $display("=== Test Completion Report ===");
    // Print out the read data stored in the queue using a while loop and pop_front.
    while (data_read_queue.size() > 0) begin
      $display("Read Data: %0h", data_read_queue.pop_front());
    end
    $display("Correct Count: %0d", correct_count);
    $display("Error Count   : %0d", error_count);

    $finish;
  end

endmodule
```

4.2 2. Bug Fixes

no bugs except cin_reg is one bit not two bits

4.3 3. Verification Plan

4.4 4. Do File

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

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
RAM_WR_1	Write Operation: When the write signal is asserted, the design should store the incoming 8-bit data along with its even parity bit (in the MSB) into the memory at the specified address.	- Randomly generate 16-bit addresses and 8-bit data values. - Drive these values on the negative clock edge and assert write until the next positive edge is reached, ensuring data is captured.		- Compare the stored memory content against the expected value using a golden model. - Flag mismatches in the testbench if the captured value deviates from the expected result.
RAM_RD_1	Read Operation: When the read signal is asserted, the design should output the lower 8 bits of the stored data from the memory corresponding to the specified address.	- Reuse the addresses from the write phase. - Drive the read signal on the proper clock edge to sample the stored data.		- Use self-check tasks to compare the DUT output with the expected data derived from the golden model. - Report and count errors if the returned data does not match the expected lower 8 bits.
RAM_PRIO_1	Priority Handling: When both write and read signals are potentially asserted, the design should prioritize the write operation and not allow simultaneous read and write.	- Generate scenarios where write and read signals could overlap. - Ensure that the write signal is asserted with higher priority (e.g., by timing write before read) in the stimulus.		- Verify that when both signals are active, the output corresponds to the write operation and that the read data is consistent with the stored value (i.e., read is not performed concurrently with a write).

Table 4: Verification Plan for the Single-Port RAM with Even Parity

4.5 5.Waveform

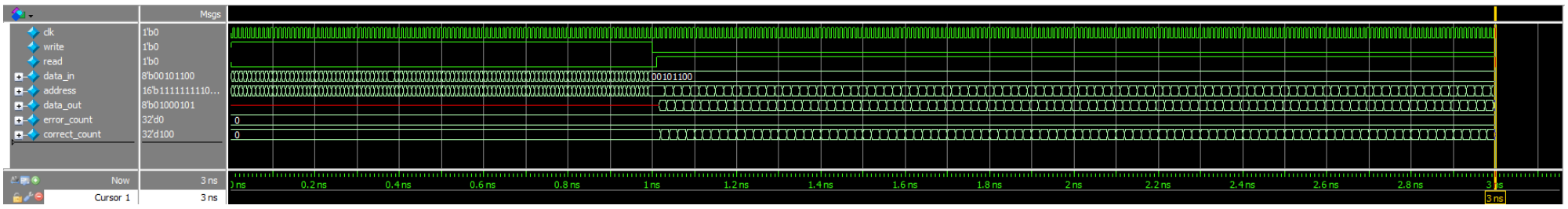

Figure 6: simulation waveform

```
# vsim -voptargs="+acc" work.tb_my_mem -coverage
# Start time: 18:11:14 on Mar 31,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt
# Loading sv_std.std
# Loading work.tb_my_mem(fast)
# Loading work.my_mem(fast)
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
#           File in use by: Administrator  Hostname: DESKTOP-B5P8C0V  ProcessID: 17172
#           Attempting to use alternate WLF file "./wlftv8ya4v".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#           Using alternate file: ./wlftv8ya4v
# === Test Completion Report ===
# Read Data: 2c
# Read Data: d6
# Read Data: 57
# Read Data: 7e
# Read Data: db
# Read Data: 59
# Read Data: 80
# Read Data: 7a
# Read Data: bd
# Read Data: 4e
# Read Data: 20
# Read Data: 4a
# Read Data: ad
# Read Data: c1
# Read Data: 74
# Read Data: 2f
# Read Data: d6
# Read Data: d4
# Read Data: ff
# Read Data: 7c
# Read Data: 74
# Read Data: 18
# Read Data: 5
# Read Data: 1c
# Read Data: c4
# Read Data: 6
# Read Data: ce
# Read Data: 81
# Read Data: 13
# Read Data: fb
# Read Data: 85
# Read Data: 35
# Read Data: a5
# Read Data: ba
# Read Data: 89
# Read Data: 14
# Read Data: 17
# Read Data: fe
# Read Data: f
# Read Data: b2
# Read Data: 33
# Read Data: 1c
# Read Data: 5e
# Read Data: 81
# Read Data: 4f
# Read Data: e0
# Read Data: 5c
# Read Data: 5e
# Read Data: e4
```

```
# Read Data: e4
# Read Data: 8
# Read Data: 98
# Read Data: 8f
# Read Data: 0
# Read Data: a1
# Read Data: fd
# Read Data: 4d
# Read Data: 96
# Read Data: 93
# Read Data: 8a
# Read Data: 43
# Read Data: 88
# Read Data: ac
# Read Data: d7
# Read Data: 9b
# Read Data: 8c
# Read Data: bb
# Read Data: 29
# Read Data: 3b
# Read Data: 21
# Read Data: 53
# Read Data: f
# Read Data: a2
# Read Data: 38
# Read Data: 59
# Read Data: d9
# Read Data: 83
# Read Data: 75
# Read Data: a3
# Read Data: 71
# Read Data: ab
# Read Data: 97
# Read Data: 2d
# Read Data: 82
# Read Data: ac
# Read Data: 48
# Read Data: c1
# Read Data: 6a
# Read Data: 82
# Read Data: ff
# Read Data: a8
# Read Data: c0
# Read Data: 2d
# Read Data: f8
# Read Data: 1
# Read Data: 59
# Read Data: bf
# Read Data: e9
# Read Data: a7
# Read Data: 88
# Read Data: 45
# Correct Count: 100
# Error Count  : 0
# ** Note: $finish    : tb_my_mem.sv(130)
#    Time: 3 ns  Iteration: 1  Instance: /tb_my_mem
```

Figure 7: Transcript : all test cases passed