# Assignment 4

# Digital Design Verification

# Contents

1	<b>Q1</b> :	ALSU	2
	1.1	1. Testbench code	2
	1.2	2. Package code	5
	1.3	3. Design code	8
	1.4	4. Bug Fixes	9
	1.5	5. Verification Plan	9
	1.6	6. Do File	9
	1.7	7. functional Coverage Report	9
	1.8	8. code Coverage Report	17
		9. Waveform	
2		SVA	<b>22</b>
	2.1	1. Code	22
_	0.0		
3	•	Counter	22
		1. Testbench code	
		2. Package code	
		3. Design code	
	3.4	4. Interface code	
	3.5	5. Bug Fixes	
	3.6	6. Verification Plan	
	3.7	7. Do File	
	3.8	8. code Coverage Report	
		9. functional coverage report	
	3.10	$10.  ext{Waveform}$	29
4		configuration register	<b>2</b> 9
	4.1	1. Testbench code	
	4.2	2. Design code	
		3. Bug report	
	4.4	4. Verification Plan	33
	4.5	5. Waveform	33

### 1 Q1: ALSU

#### 1.1 1. Testbench code

```
1 'timescale 1ns/1ps
3 import ALSU_pkg::*;
5 module ALSU_tb;
    // -----
    // Testbench signals
    logic clk;
    logic rst;
11
    logic cin;
    logic red_op_A;
13
    logic red_op_B;
14
    logic bypass_A;
    logic bypass_B;
    logic direction;
17
    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;
23
    // -----
    // DUT instantiation
26
    ALSU #(
      .INPUT_PRIORITY("A"),
29
      .FULL_ADDER("ON")
    ) dut (
31
      .clk
                 (clk),
      .rst
                 (rst),
      .cin
                 (cin),
34
                 (red_op_A),
      .red_op_A
35
      .red\_op\_B
                 (red_op_B),
      .bypass_A
                 (bypass_A),
      .bypass_B
                 (bypass_B),
      .direction (direction),
      .serial_in (serial_in),
      . A
                  (A),
      . В
                 (B),
      .opcode
                (opcode),
43
      .leds
                (leds),
                 (out)
      .out
46
    // Clock & reset generation
49
    initial begin
51
          clk = 0;
52
53
          forever begin
           #5 clk = clk;
54
     end
    // Create an object for random stimulus
60
    // -----
61
    alsu_rand_class rand_stim;
  // -----
    // reset task
    task do_reset();
      rst = 1;
      #10;
69
      // Check result against a golden model
70
        golden_model(
          rst, A, B, cin, serial_in, red_op_A, red_op_B, bypass_A, bypass_B,
          direction, opcode
        );
      #10;
      rst = 0;
    endtask
77
    // -----
    // Golden model for reference
80
    // -----
81
   task golden_model(
82
      input logic rst,
83
      input logic signed [2:0] A, B,
84
85
      input logic cin, serial_in, red_op_A, red_op_B,
      input logic bypass_A, bypass_B, direction,
86
87
      input logic [2:0] opcode
88 );
89
      logic signed [5:0] expected_out;
90
      logic [15:0] expected_leds;
91
92
      logic invalid_red_op, invalid_opcode, invalid;
93
94
      // Invalid condition handling
95
      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;
        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'hO: 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};
                 expected_out = {serial_in, expected_out[5:1]};
        end
        3'h5: begin // ROTATE
            if (direction)
                 expected_out = {expected_out[4:0], expected_out[5]};
                expected_out = {expected_out[0], expected_out[5:1]};
        end
        default: expected_out = 0;
    {\tt endcase}
    end
    // Wait another clock so the output is stable
    @(posedge clk);
    if ((out != expected_out) && (leds != expected_leds)) begin
        error("[ALSU]_{\sqcup}Mismatch_{\sqcup}with_{\sqcup}golden_{\sqcup}model:_{\sqcup}opcode=%0b._{\sqcup}out=%0d,_{\sqcup}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;
    {\tt cin}
              = 0;
    red_op_A = 0;
    red_op_B = 0;
    bypass_A = 0;
    bypass_B = 0;
    direction = 0;
    serial_in = 0;
    opcode = 0;
    Α
              = 0;
              = 0;
    В
    \ensuremath{//} 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();
```

100

101

102

103 104

105 106

107

108 109 110

112

113

114 115

116

117

118

119

120

121

122

123 124

125

126 127

128

129 130

131

132

133

134 135

136

137

138

139

140

142

143

144

145 146 147

148

149 150

152

153

154

155

156

157 158

159

160 161 162

163

164

165

166

167 168

169

170 171

172 173

174 175

177

178

179

180

181

182

183

184

185

186

187

188 189 190

191 192

193 194

```
// Phase 1: Full Constrained Randomization (Constraints 1-7 enabled)
// Disable constraint 8 here
// -----
rand_stim.disable_constrain_8();
display("Phase_1:_{\Box}Full_{\Box}constrained_{\Box}randomization_{\Box}with_{\Box}constraints_{\Box}1-7");
for (i = 0; i < 500; 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;
           = rand_stim.A;
           = 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();
$display("ALSU_Phase_1_test_completed");
// Transition to Phase 2: Opcode Verification
// -----
$display("Transitioning_to_Phase_2:_0pcode_verification");
// Force key signals to 0 as required
      = O;
rst
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
       = rand_stim.cin;
direction = rand_stim.direction;
serial_in = rand_stim.serial_in;
         = rand_stim.A;
         = 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_uwithout_uconstraints_ufailed_uin_uphase_u2");
   $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,
```

198

199

200

201

202

203 204

205

207

208

210 211

212

213

214 215

216

217

218

219

220

221

222 223

224

225 226 227

228

229

230

231

232 233

234 235

236

237

239 240

241242243

244

245

246 247

248

249 250

251

252 253

254

255

256 257

258

259

260 261

262

263

265

266

268 269

271

272 273

274 275

277

278

279 280

281

282

283 284

285

286 287

288

289

290

291

292 293

294

```
direction, opcode
          );
298
299
         // Sample only when not in reset or bypass mode
300
         if (!(rst || bypass_A || bypass_B)) begin
301
          rand_stim.sample();
302
303
304
       end
305
       // -----
307
       // End simulation and dump coverage data if needed
308
309
       // -----
       $display("Testbench completed.");
310
       // direct rst value for 0 \rightarrow 1
311
       @(posedge clk);
312
       rst
              = 1;
313
       #20;
314
       $finish;
315
     end
316
317
318 endmodule
        2. Package code
   package ALSU_pkg;
       // 1) Define the opcodes (including invalids)
       // -----
       typedef enum logic [2:0] {
                   = 3'h0, // 000
          OR_O
                    = 3'h1, // 001
          XOR_1
                    = 3'h2, // 010
          ADD_2
                  = 3'h3, // 011
= 3'h4, // 100
          MUL_3
          SHIFT_4
          ROTATE_5 = 3'h5, // 101
          INVALID_6 = 3'h6, // 110
12
          INVALID_7 = 3'h7 // 111
       } opcode_e;
14
       // -----
       // 2) 3-bit signed range is -4 .. +3
       // -----
       localparam logic signed [2:0] MAXNEG = -4; // 3'b100
19
       localparam logic signed [2:0] ZERO = 0;
20
       localparam logic signed [2:0] MAXPOS = 3; // 3'b011
21
22
       class alsu_rand_class;
23
          // -----
          // 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;
32
          rand bit
                                bypass_B;
          rand bit
                                direction;
34
          rand bit
                                serial_in;
          rand opcode_e
                                opcode;
          rand logic signed [2:0] A;
          rand logic signed [2:0] B;
38
          rand opcode_e
                                opcode_array[6];
          bit disable_all = 0;
          bit disable_8 = 0;
44
          // 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 };
              }
55
          // (b) For ADD or MUL, pick corner values of A,B more often
56
                 (MAXNEG, ZERO, MAXPOS) than the other possibilities.
57
                 Weighted distribution is used here.
58
          constraint c_adder_mult_corner {
              if (!disable_all) {
60
                  if (opcode inside {ADD_2, MUL_3}) {
61
                   A dist { MAXNEG := 3, ZERO := 3, MAXPOS := 3, [-3:-1] := 1, [1:2] := 1 };
62
                   B dist { MAXNEG := 3, ZERO := 3, MAXPOS := 3, [-3:-1] := 1, [1:2] := 1 };
63
                  }
64
              }
          }
66
67
68
          // (c) If opcode=OR or XOR and red_op_A=1, then A has exactly one bit set
69
                and B is 0 .
70
          constraint c_red_opA_onebit {
71
              if (!disable_all) {
72
                  if ((opcode==OR_O || opcode==XOR_1) && red_op_A==1) {
73
74
                   // 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,
                     := 5,
          OR_O
                     := 5,
          XOR_1
          ADD_2
                     := 5,
          MUL_3
                     := 5,
          SHIFT_4
                     := 5,
                    := 5
          ROTATE_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;
      }
    }
// Functional Coverage
covergroup cg;
  coverpoint rst;
  coverpoint cin;
  red_op_A_cp: coverpoint red_op_A{
    bins red_op_A_0 = \{0\};
    bins red_op_A_1 = \{1\};
    bins red_op_A_default = default;
  red_op_B_cp: coverpoint red_op_B{
    bins red_op_B_0 = \{0\};
    bins red_op_B_1 = \{1\};
    bins red_op_B_default = default;
  coverpoint bypass_A;
  coverpoint bypass_B;
  coverpoint direction;
  coverpoint serial_in;
  A_cp: coverpoint A {
    bins A_{data_0} = \{0\};
    bins A_data_max = {MAXPOS};
```

82

83

84

86

87 88

89

90

91 92

93

94

95

96

98

99

101

102

103

104

105

106

107

108

109 110

111

112

113 114

115

116

117

118

119

121

122 123

124

125 126

127

128 129

130

131 132

133

134

135

136 137

138

139 140 141

142

144

145

146

147

148

154

156

157 158

159

160

161

162 163 164

165 166

167

168

169

170

171

172

173

```
bins A_data_min = {MAXNEG};
    bins A_data_default = default;
    bins A_data_walkingones[] = {3'b001, 3'b010, 3'b100};
 B_cp: coverpoint B {
    bins B_data_0 = \{0\};
    bins B_data_max = {MAXPOS};
    bins B_data_min = {MAXNEG};
    bins B_data_default = default;
    bins B_data_walkingones[] = {3'b001, 3'b010, 3'b100};
 A_cp_mod: coverpoint A {
    bins A_{data_0} = \{0\};
    bins A_data_max = {MAXPOS};
    bins A_data_min = {MAXNEG};
  B_cp_mod: coverpoint B {
    bins B_data_0 = \{0\};
    bins B_data_max = {MAXPOS};
   bins B_data_min = {MAXNEG};
  ALU_cp: coverpoint opcode {
    bins Bins_shift[] = {SHIFT_4, ROTATE_5};
    bins Bins_arith[] = {ADD_2, MUL_3};
    bins Bins_bitwise[] = {OR_0, XOR_1};
    bins Bins_invalid = {INVALID_6, INVALID_7};
    bins Bins_trans = (OR_O => XOR_1 => ADD_2 => MUL_3 => SHIFT_4 => ROTATE_5);
  cross A_cp , red_op_A_cp{
  ignore_bins assert_red_op_A = binsof(A_cp.A_data_walkingones) && binsof(red_op_A_cp.red_op_A_1);
  cross B_cp , red_op_A_cp , red_op_B_cp{
  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};
  cross A_cp_mod, B_cp_mod, ALU_cp {
    bins arith_permutations = binsof(ALU_cp.Bins_arith) && binsof(A_cp_mod) && binsof(B_cp_mod);
    option.cross_auto_bin_max=0;
  cross cin, ALU_cp {
    bins addition_cin = binsof(ALU_cp.Bins_arith) intersect {ADD_2} && binsof(cin);
    option.cross_auto_bin_max=0;
  cross direction, ALU_cp {
    bins shift_rotate_direction = binsof(ALU_cp.Bins_shift) && binsof(direction);
    option.cross_auto_bin_max=0;
  cross serial_in, ALU_cp {
    bins shift_serial_in = binsof(ALU_cp.Bins_shift)intersect {SHIFT_4} && binsof(serial_in);
    option.cross_auto_bin_max=0;
 }
  cross A_cp, red_op_A_cp, B_cp, ALU_cp {
    bins or_xor_red_op_A = binsof(ALU_cp.Bins_bitwise) && binsof(red_op_A_cp.red_op_A_1) && binsof(A_cp.A_data_walkingones) &&
       binsof(B_cp.B_data_0);
    option.cross_auto_bin_max=0;
 }
  cross B_cp, red_op_B_cp, A_cp, ALU_cp {
    bins or_xor_red_op_B = binsof(ALU_cp.Bins_bitwise) && binsof(red_op_B_cp.red_op_B_1) && binsof(B_cp.B_data_walkingones) &&
       binsof(A_cp.A_data_0);
    option.cross_auto_bin_max=0;
 }
  cross ALU_cp, red_op_A_cp, red_op_B_cp {
    ignore_bins invalid_reduction = binsof(ALU_cp.Bins_bitwise) && (binsof(red_op_A_cp.red_op_A_1) ||
       binsof(red_op_B_cp.red_op_B_1));
 }
endgroup
// constructor
function new();
 cg = new();
endfunction
function void stop();
   cg.stop();
endfunction
function void start();
  cg.start();
endfunction
function void sample();
   cg.sample();
\verb"endfunction"
// disabled all constrains expect 8
function void disabled_all_constrains_expect_8();
  disable_all = 1;
  disable_8 = 0;
endfunction
```

178 179

180

181

182

183

184

186

187 188

189

190 191

192

193

194

195 196

198

199

201

202 203 204

205

206 207

210

211 212 213

214

215 216 217

218

220 221 222

223

224

225226227228

229

230

231 232 233

234

236 237

238 239

241242243

244

246 247

248 249

250

252

253 254

255

256

257 258

259 260

261 262 263

264

265 266

267

269

270

```
272
            // disable constrain 8
273
            function void disable_constrain_8();
274
              disable_all = 0;
275
              disable_8 = 1;
            endfunction
277
278
            // enable all constrains
279
            function void enable_all_constrains();
280
              disable_all = 0;
              disable_8 = 0;
282
            endfunction
283
284
285
        endclass
286
288
289 endpackage
   1.3 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";
 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;
 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;
16
17 //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];
20 assign invalid = invalid_red_op | invalid_opcode;
//Registering input signals
23 always @(posedge clk or posedge rst) begin
     if(rst) begin
24
         cin_reg <= 0;</pre>
25
         red_op_B_reg <= 0;</pre>
         red_op_A_reg <= 0;</pre>
27
         bypass_B_reg <= 0;</pre>
         bypass_A_reg <= 0;</pre>
         direction_reg <= 0;
30
         serial_in_reg <= 0;
31
         opcode_reg <= 0;
32
         A_reg <= 0;
33
         B_reg <= 0;
      end else begin
35
         cin_reg <= cin;</pre>
36
         red_op_B_reg <= red_op_B;</pre>
         red_op_A_reg <= red_op_A;</pre>
38
         bypass_B_reg <= bypass_B;</pre>
39
         bypass_A_reg <= bypass_A;</pre>
         direction_reg <= direction;</pre>
41
42
         serial_in_reg <= serial_in;</pre>
         opcode_reg <= opcode;</pre>
44
         A_reg <= A;
         B_reg \le B;
45
      end
47 end
48
49 //leds output blinking
50 always @(posedge clk or posedge rst) begin
      if(rst) begin
         leds <= 0;
      end else begin
53
54
          if (invalid)
           leds <= ~leds;</pre>
55
          else
56
            leds <= 0;
      end
59 end
61 //ALSU output processing
    always @(posedge clk or posedge rst) begin
62
      if(rst) begin
        out <= 0;
64
      end
65
      else begin
66
        if (bypass_A_reg && bypass_B_reg)
67
          out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;</pre>
68
        else if (bypass_A_reg)
70
          out <= A_reg;
        else if (bypass_B_reg)
71
72
          out <= B_reg;
        else if (invalid)
73
            out <= 0;
74
75
        else begin
            case (opcode)
76
              3'h0: begin
77
78
                 if (red_op_A_reg && red_op_B_reg)
                   out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;</pre>
```

```
else if (red_op_A_reg)
                   out <= |A_reg;
                 else if (red_op_B_reg)
                   out <= |B_reg;
84
                 else
                   out <= A_reg | B_reg;
85
86
               3'h1: begin
                 if (red_op_A_reg && red_op_B_reg)
                   out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;</pre>
                 else if (red_op_A_reg)
90
                   out <= ^A_reg;
91
92
                 else if (red_op_B_reg)
                   out <= ^B_reg;
93
                 else
94
                   out <= A_reg ^ B_reg;</pre>
               end
96
               3'h2: out <= A_reg + B_reg;
97
               3'h3: out <= A_reg * B_reg;
               3'h4: begin
99
                 if (direction_reg)
100
101
                   out <= {out[4:0], serial_in_reg};</pre>
                 else
102
103
                   out <= {serial_in_reg, out[5:1]};</pre>
104
               end
               3'h5: begin
105
                 if (direction_reg)
106
                   out <= {out [4:0], out [5]};
107
                 else
108
                   out <= {out[0], out[5:1]};
109
               end
110
             endcase
112
113
     end
114 end
115
116 endmodule
```

## 1.4 4. Bug Fixes

no bugs except cin\_reg is one bit not two bits

#### 1.5 5. Verification Plan

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

Table 1: Verification Plan

### 1.6 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 code_coverage_rpt.txt -du=ALSU
#—vcover report -details -cvg -output functional_coverage_report.txt ALSU_tb.ucdb
```

#### 1.7 7. functional Coverage Report

Coverage Report by instance with details

=== Instance: /ALSU\_pkg === Design Unit: work.ALSU\_pkg

Covergroup Coverage:
Covergroups
Coverpoints/Crosses
Covergroup Bins 1 99.12%nana22 nanana98 93 5 94.89%

vergroup	Metric	Goal	Bins	Status
PE /ALSU_pkg/alsu_rand_class/cg	99.12%	100		Uncovered
covered/total bins:	93	98	_	Oncovered
missing/total bins:	5	98	_	
% Hit:	94.89%	100	_	
			_	G 1
Coverpoint rst	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
Coverpoint cin	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	$\frac{2}{2}$		
% Hit:	100.00%	100		
			_	G 1
Coverpoint red_op_A_cp	100.00%	100	_	$\operatorname{Covered}$
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			covered
		2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
Coverpoint bypass_A	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
				Comonad
Coverpoint bypass_B	100.00%	100	_	$\operatorname{Covered}$
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	_	00,0104
			_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
Coverpoint serial_in	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	$100^{-}$	_	
Coverpoint A_cp	100.00%	100		Covered
			_	Covered
covered/total bins:	5	5	_	
missing/total bins:	0	5	_	
% Hit:	100.00%	100	_	
$\operatorname{Coverpoint}\ \operatorname{B_cp}$	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	5	5	_	
missing/total bins:	0	5	_	
% Hit:	100.00%	100		
			_	C 1
Coverpoint A_cp_mod	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	3	3	_	
missing/total bins:	0	3	_	
% Hit:	100.00%	100	_	
Coverpoint B_cp_mod	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	3	3	_	
missing/total bins:	0	3		
			_	
% Hit:	100.00%	100	_	
$\operatorname{Coverpoint} \ \operatorname{ALU\_cp}$	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	8	8	_	
missing/total bins:	0	8	_	
% Hit:	100.00%	100	_	
Cross #cross0#	100.00%	100	_	Covered
covered/total bins:	100.00%	8	_	COVOICU
			_	
missing/total bins:	0	8	_	
% Hit:	100.00%	100	_	
Cross #cross1#	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	18	18	_	
missing/total bins:	0	18	_	
% Hit:	100.00%	100	_	
Cross #cross2#	100.00%	100		Covered
			_	Covered
covered/total bins:	1	1	_	
missing/total bins:	0	1	_	
% Hit:	100.00%	100	_	
Cross #cross3#	100.00%	100	_	Covered
covered/total bins:	1	1	_	
missing/total bins:	0	1		
			_	
% Hit:	100.00%	100	_	· ·
Cross #cross4#	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	1	1	_	
missing/total bins:	0	1	_	
% Hit:	100.00%	100	_	
Cross #cross_5#	100.00%	100	_	Covered
covered/total bins:	100.00%	100	_	SSTORE
missing/total bins:	0	1	_	
		1	_	

M 1111	100000	100		
% Hit:	100.00%	100	_	C 1
Cross #cross6# covered/total bins:	100.00% 1	100	_	Covered
missing/total bins:	0	1 1	_	
% Hit:	100.00%	100		
Cross #cross7#	100.00%	100	_	Covered
covered/total bins:	1	1	_	Covered
missing/total bins:	0	1	_	
% Hit:	100.00%	100	_	
Cross #cross8#	80.76%	100	_	${ m Uncovered}$
covered/total bins:	21	26	_	
missing/total bins:	5	26	_	
% Hit:	80.76%	100	_	
$Covergroup \ instance \ \backslash /ALSU\_pkg :: alsu\_rand\_class :: cg$	00 100	100		
	99.12%	100	_	Uncovered
<pre>covered/total bins: missing/total bins:</pre>	93	98 98	_	
% Hit:	94.89%	100	_	
Coverpoint rst	100.00%	100	_	Covered
covered/total bins:	2	2	_	Covered
missing/total bins:	0	$\overline{2}$	_	
% Hit:	100.00%	100	_	
$ \text{bin } \text{ auto} \left[ 0 \right]$	268	1	_	$\operatorname{Covered}$
bin auto[1]	18	1	_	$\operatorname{Covered}$
Coverpoint cin	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	C 1
bin auto [0]	$\begin{array}{c} 142 \\ 144 \end{array}$	1 1	_	Covered Covered
bin auto[1] Coverpoint red_op_A_cp	100.00%	100	_	Covered
covered/total bins:	$\frac{100.0076}{2}$	2		Covered
missing/total bins:	0	$\frac{2}{2}$	_	
% Hit:	100.00%	100	_	
bin red_op_A_0	55	1	_	$\operatorname{Covered}$
bin red_op_A_1	231	1	_	Covered
default bin red_op_A_default	0		_	ZERO
$\operatorname{Coverpoint}\ \operatorname{red} \operatorname{-op} \operatorname{-B} \operatorname{-cp}$	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
bin red_op_B_0	47	1	_	Covered
bin red_op_B_1	239	1	_	Covered
default bin red_op_B_default Coverpoint bypass_A	$0 \\ 100.00\%$	100	_	ZERO Covered
covered/total bins:	$\frac{100.00\%}{2}$	$\frac{100}{2}$	_	Covered
missing/total bins:	0	$\frac{2}{2}$		
% Hit:	100.00%	100	_	
bin auto[0]	284	1	_	Covered
bin auto[1]	$\frac{1}{2}$	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]	281	1	_	Covered
bin auto[1]	5	1	_	Covered
Coverpoint direction covered/total bins:	$100.00\% \ 2$	$\frac{100}{2}$	_	Covered
missing/total bins:	0	$\frac{2}{2}$		
% Hit:	100.00%	100	_	
bin auto[0]	146	1	_	$\operatorname{Covered}$
bin auto[1]	140	1	_	$\operatorname{Covered}$
Coverpoint serial_in	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	Constant
bin auto [0]	$149 \\ 137$	1	_	Covered Covered
bin auto[1] Coverpoint A_cp	$137 \\ 100.00\%$	$\begin{matrix} 1 \\ 100 \end{matrix}$	_	Covered
covered/total bins:	$\frac{100.00\%}{5}$	5		Covered
missing/total bins:	0	5	_	
% Hit:	100.00%	100	_	
$\operatorname{bin} \ \operatorname{A_data_0}$	74	1	_	$\operatorname{Covered}$
bin A_data_max	37	1	_	Covered
bin A_data_min	37	1	_	Covered
bin A_data_walkingones[1]	31	1	_	Covered
bin A_data_walkingones[2]	32	1	_	Covered
default bin A_data_default	75	100	_	Occurred
Coverpoint B_cp	100.00%	100	_	$\operatorname{Covered}$
<pre>covered/total bins: missing/total bins:</pre>	$\frac{5}{0}$	5 5	_	
missing/total bins: % Hit:	100.00%	100	_	
bin $B_{-}data_{-}0$	53	100	_	Covered
bin B_data_max	41	1	_	Covered
bin B_data_min	44	1	_	Covered
bin B_data_walkingones[1]	38	1	_	Covered
bin B_data_walkingones[2]	33	1	_	Covered
default bin B_data_default	77		_	Occurred
Coverpoint A_cp_mod	100.00%	100	_	Covered
covered/total bins:	3	3	_	
missing/total bins: % Hit:	$0 \\ 100.00\%$	$\begin{array}{c} 3 \\ 100 \end{array}$	_	
/U <b>111</b> 0.	100.00/0	100	_	

bin A_data_0	74	1	_	Covered
bin A_data_max bin A_data_min	37 37	1 1	_	Covered Covered
Coverpoint B_cp_mod	100.00%	100	_	Covered
covered/total bins:	3	3	_	Covered
missing/total bins:	0	3	_	
% Hit:	100.00%	100	_	
bin B_data_0	53	1	_	Covered
bin B_data_max bin B_data_min	$\begin{array}{c} 41 \\ 44 \end{array}$	1 1	_	Covered Covered
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] bin Bins_shift [ROTATE_5]	47 $61$	1 1	_	Covered Covered
bin Bins_arith [ADD_2]	49	1	_	Covered
bin Bins_arith [MUL_3]	53	1	_	Covered
bin Bins_bitwise[OR_0]	29	1	_	Covered
bin Bins_bitwise [XOR_1]	33	1	_	Covered
bin Bins_invalid bin Bins_trans	14 1	1 1	_	Covered Covered
Cross #cross0#	100.00%	100	_	Covered
covered/total bins:	8	8	_	00.0100
missing/total bins:	0	8	_	
% Hit:	100.00%	100	_	
Auto, Default and User Defined Bins: bin <a_data_walkingones[2], red_op_a_0<="" td=""><td></td><td>1</td><td></td><td>Covered</td></a_data_walkingones[2],>		1		Covered
bin $A_{\text{data}}$ walkingones [2], red_op_A_0 bin $A_{\text{data}}$ walkingones [1], red_op_A_0		1	_	Covered
bin <a_data_min, red_op_a_1=""></a_data_min,>	31	1	_	Covered
${\rm bin}~<\!{\rm A_{-}data\_min}~, {\rm red\_op\_A\_0}\!>$	6	1	_	Covered
$bin < A_data_max, red_op_A_1>$	34	1	_	Covered
bin <a_data_0 ,="" red_op_a_1=""> bin <a_data_max ,="" red_op_a_0=""></a_data_max></a_data_0>	$\begin{array}{c} 45 \\ 3 \end{array}$	1 1	_	Covered Covered
bin $\langle A_{\text{data}} = \max_{\text{red}}   \text{red} = 0 \rangle$	$\frac{3}{29}$	1	_	Covered
Illegal and Ignore Bins:	20	-		00,0104
$ignore\_bin \ assert\_red\_op\_A$	56		_	Occurred
Cross #cross1#	100.00%	100	_	Covered
<pre>covered/total bins: missing/total bins:</pre>	18 0	18 18	_	
% Hit:	100.00%	100	_	
Auto, Default and User Defined Bins:				
bin $\langle B_{data\_walkingones}[2], red\_op_A_1$	, red_op_B_1> 22	1		Covered
bin <b_data_walkingones[2], red_op_a_1<="" td=""><td></td><td>1</td><td></td><td>Covered</td></b_data_walkingones[2],>		1		Covered
	2	1	_	Covered
$bin < B_{data\_walkingones}[1], red\_op_A_1$		1		C1
$bin < B_{data\_walkingones}[1], red_{op\_A\_1}$	25 red on B 0>	1	_	Covered
Sin (Baddou Wallingones [1]), red-op-11-1	2	1	_	Covered
$bin < B_data_min, red_op_A_1, red_op_B_1$		1	_	Covered
bin <b_data_min, red_op_a_1,="" red_op_b_0<="" td=""><td></td><td>1</td><td>_</td><td>Covered</td></b_data_min,>		1	_	Covered
bin <b_data_max, red_op_a_1,="" red_op_b_1<br="">bin <b_data_0, red_op_a_1,="" red_op_b_1=""></b_data_0,></b_data_max,>	> 37 35	1 1	_	$egin{array}{c} { m Covered} \end{array}$
bin <b_data_max, red_op_a_1,="" red_op_b_0<="" td=""><td></td><td>1</td><td>_</td><td>Covered</td></b_data_max,>		1	_	Covered
$bin < B_data_0, red_op_A_1, red_op_B_0 >$	14	1	_	Covered
$bin < B_{data\_walkingones}[2], red_{op\_A\_0}$				
bin <b_data_walkingones[1], red_op_a_0<="" td=""><td>2</td><td>1</td><td>_</td><td>Covered</td></b_data_walkingones[1],>	2	1	_	Covered
bin \D_dava_warkingones[1], red_op_n_o	3	1	_	Covered
$bin < B_data_min, red_op_A_0, red_op_B_1$		1	_	Covered
$ bin < B_{-}data_{-}min, red_{-}op_{-}A_{-}0, red_{-}op_{-}B_{-}0 $		1	_	Covered
bin <b_data_max, red_op_a_0,="" red_op_b_1<="" td=""><td></td><td>1</td><td>_</td><td>Covered</td></b_data_max,>		1	_	Covered
bin <b_data_0 ,="" red_op_a_0="" red_op_b_1=""> bin <b_data_max ,="" red_op_a_0="" red_op_b_0<="" td=""><td>&gt; 2</td><td>1 1</td><td>_</td><td><math>egin{array}{c} { m Covered} \end{array}</math></td></b_data_max></b_data_0>	> 2	1 1	_	$egin{array}{c} { m Covered} \end{array}$
bin $\langle B_{\text{data}} = 0, \text{red} = 0, red$	2	1	_	Covered
Illegal and Ignore Bins:				
ignore_bin assert_red_op_B	100,00%	100	_	Occurred
$\frac{\text{Cross } \#\text{cross}_{-2}\#}{\text{covered/total bins:}}$	100.00%	$100\\1$	_	Covered
missing/total bins:	0	1	_	
% Hit:	100.00%	100	_	
Auto, Default and User Defined Bins: bin arith_permutations	29	1		Covered
Cross #cross3#	100.00%	100	_	Covered
covered/total bins:	1	1	_	
missing/total bins:	0	1	_	
% Hit: Auto, Default and User Defined Bins:	100.00%	100	_	
bin addition_cin	49	1	_	Covered
$Cross \#cross_{-2}4\#$	100.00%	100	_	Covered
covered/total bins:	1	1	_	
missing/total bins: % Hit:	$0\\100.00\%$	$\begin{matrix} 1 \\ 100 \end{matrix}$	_	
Auto, Default and User Defined Bins:	100.0070	100		
bin shift_rotate_direction	108	1	_	Covered
$\frac{\text{Cross } \#\text{cross}\_5\#}{\text{covered/total bins:}}$	100.00%	100	_	Covered
missing/total bins:	$\frac{1}{0}$	1 1	_	
% Hit:	100.00%	100	_	
Auto, Default and User Defined Bins: bin shift_serial_in	47	1		Covered
5111 511110 SC1161_1II	41	1	_	Ouvered

Cross #cross6#	100.00%	100	_	Covered
<pre>covered/total bins: missing/total bins:</pre>	$\frac{1}{0}$	1 1	_	
% Hit:	100.00%	100	_	
Auto, Default and User Defined Bins: bin or_xor_red_op_A	5	1	_	Covered
Cross #cross7# covered/total bins:	100.00%	$100\\1$	_	Covered
missing/total bins:	0	1	_	
% Hit: Auto, Default and User Defined Bins:	100.00%	100	_	
bin or_xor_red_op_B	11	1	_	Covered
Cross #cross8# covered/total bins:	80.76% $21$	$\frac{100}{26}$	_	Uncovered
missing/total bins:	5	26	_	
% Hit: Auto, Default and User Defined Bins:	80.76%	100	_	
<pre>bin <bins_trans ,="" red_op_a_1="" red_op_b_0=""> bin <bins_invalid ,="" pre="" red_op_a_1="" red_op_b_<=""></bins_invalid></bins_trans></pre>		1	_	Covered
bin <bins_invalid ,="" red_op_a_1="" red_op_b_<="" td=""><td>3</td><td>1</td><td>_</td><td>Covered</td></bins_invalid>	3	1	_	Covered
	4	1	_	Covered
bin <bins_invalid ,="" red_op_a_0="" red_op_b_<="" td=""><td>1&gt;</td><td>1</td><td>_</td><td>Covered</td></bins_invalid>	1>	1	_	Covered
bin <bins_invalid ,="" red_op_a_0="" red_op_b_<="" td=""><td>0&gt;</td><td>1</td><td>_</td><td>Covered</td></bins_invalid>	0>	1	_	Covered
$\label{eq:bins_bitwise} bin < Bins_bitwise [XOR_1], red_op_A_0, $		1		Covered
$\label{eq:constraints} bin <& Bins\_bitwise \left[  OR\_0  \right], red\_op\_A\_0 \;, red$	_op_B_0>			
$\label{eq:bins_arith_MUL_3} bin <& Bins_arith_{}[MUL_3], red_op_A_1, red_op_A_2, red_op_A_3, red_op_A_4, red_op_A_5, red_op_A_5, red_op_A_6, red_op_$	_	1	_	Covered
bin <bins_shift [rotate_5],="" r<="" red_op_a_1,="" td=""><td><math>ed_{op}B_{1}&gt;</math></td><td>1</td><td>_</td><td>Covered</td></bins_shift>	$ed_{op}B_{1}>$	1	_	Covered
bin <bins_arith[mul_3], red_<="" red_op_a_1,="" td=""><td>56 op_B_0&gt;</td><td>1</td><td>_</td><td>Covered</td></bins_arith[mul_3],>	56 op_B_0>	1	_	Covered
bin <bins_shift ,="" [rotate.5]="" r<="" red_op_a_1="" td=""><td><math>\overline{4}</math></td><td>1</td><td>_</td><td>Covered</td></bins_shift>	$\overline{4}$	1	_	Covered
bin <bins_arith [add_2],="" red_<="" red_op_a_1,="" td=""><td><math>\overline{4}</math></td><td>1</td><td>_</td><td>Covered</td></bins_arith>	$\overline{4}$	1	_	Covered
	44	1	_	Covered
bin <bins_shift [shift_4],="" re<="" red_op_a_1,="" td=""><td>43</td><td>1</td><td>_</td><td>Covered</td></bins_shift>	43	1	_	Covered
bin <bins_arith [add_2],="" red_<="" red_op_a_1,="" td=""><td>1</td><td>1</td><td>_</td><td>Covered</td></bins_arith>	1	1	_	Covered
bin <bins_shift [shift_4],="" re<="" red_op_a_1,="" td=""><td><math>d_{-}op_{-}B_{-}0&gt;</math></td><td>1</td><td>_</td><td>Covered</td></bins_shift>	$d_{-}op_{-}B_{-}0>$	1	_	Covered
$\label{eq:bins_arith_MUL_3} \ \ \mathrm{bin} \ \ \mathrm{$	op_B_1>	1	_	Covered
bin <bins_shift [rotate_5],="" r<="" red_op_a_0,="" td=""><td>_</td><td>1</td><td></td><td>Covered</td></bins_shift>	_	1		Covered
$\label{eq:bins_arith_MUL_3} bin <& Bins_arith_{}[MUL_3], red_op_A_0, red_op_$	op_B_0>			
$\label{eq:bins_arith_ADD_2} bin <& Bins_arith_ADD_2] \ , red_op_A_0 \ , red_$	$op_B_1>$	1	_	Covered
bin <bins_shift [shift_4],="" re<="" red_op_a_0,="" td=""><td><math>d_{-}op_{-}B_{-}1&gt;</math></td><td>1</td><td>_</td><td>Covered</td></bins_shift>	$d_{-}op_{-}B_{-}1>$	1	_	Covered
bin <bins_shift[shift_4], re<="" red_op_a_0,="" td=""><td>1 d_op_B_0&gt;</td><td>1</td><td>_</td><td>Covered</td></bins_shift[shift_4],>	1 d_op_B_0>	1	_	Covered
• • • •	1	1	_	Covered
<pre>bin <bins_trans ,*="" ,red_op_a_0=""> bin <bins_trans ,*="" ,red_op_b_1=""></bins_trans></bins_trans></pre>	$0 \\ 0$	1 1	$\frac{2}{2}$	ZERO ZERO
$\label{eq:bins_shift} bin < Bins_shift [ROTATE_5] \ , red_op_A_0 \ , r$	$ed_{0}p_{B_{0}}>$	1	1	ZERO
$\label{eq:bins_arith_ADD_2} bin <& Bins_arith_{} [ADD_2] \;, red_op_A_0 \;, red_op_A_0$	op_B_0>			
Illegal and Ignore Bins:	0	1	1	ZERO
ignore_bin invalid_reduction	47		_	Occurred

## COVERGROUP COVERAGE:

Covergroup	Metric	$\operatorname{Goal}$	$\operatorname{Bins}$	Status	
TYPE /ALSU_pkg/alsu_rand_class/cg	99.12%	100		Uncovered	
covered/total bins:	93	98	_		
missing/total bins:	5	98	_		
% Hit:	94.89%	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	_	22,0204	
missing/total bins:	0	$\frac{2}{2}$	_		
% Hit:	100.00%	100	_		

Coverpoint bypass_A		100		Covered
	100.00%	100	_	Covered
acromod/total bing.	2			Covered
covered/total bins:	_	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
	100.00%	100		Covered
Coverpoint bypass_B			_	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	$\overline{2}$		
	-		_	
% Hit:	100.00%	100	_	
Coverpoint serial_in	100.00%	100	_	Covered
-	2			00.0104
covered/total bins:		2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
				O 1
$\operatorname{Coverpoint} A_{\operatorname{-}}\operatorname{cp}$	100.00%	100	_	Covered
covered/total bins:	5	5	_	
missing/total bins:	0	5	_	
% Hit:	100.00%	100	_	
$\operatorname{Coverpoint} \ \operatorname{B\_cp}$	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	5	5		
,				
missing/total bins:	0	5	_	
% Hit:	100.00%	100	_	
Coverpoint A_cp_mod	100.00%	100	_	Covered
				Covered
covered/total bins:	3	3	_	
missing/total bins:	0	3	_	
% Hit:	100.00%	100		
			_	
$\operatorname{Coverpoint}\ \operatorname{B\_cp\_mod}$	100.00%	100	_	Covered
covered/total bins:	3	3	_	
missing/total bins:	0	3	_	
% Hit:	100.00%	100	_	
Coverpoint ALU_cp	100.00%	100	_	Covered
				Covered
covered/total bins:	8	8	_	
missing/total bins:	0	8	_	
% Hit:	100.00%	100	_	
				<i>a</i> ,
$Cross \#cross_{}0\#$	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	8	8	_	
	0			
missing/total bins:	-	8	_	
% Hit:	100.00%	100	_	
Cross #cross1#	100.00%	100	_	Covered
				00.0104
covered/total bins:	18	18	_	
missing/total bins:	0	18	_	
% Hit:	100.00%	100	_	
				O 1
$Cross \#cross_{-2}\#$	100.00%	100	_	Covered
covered/total bins:	1	1	_	
missing/total bins:	0	1	_	
% Hit:	100.00%	100	_	
	100.00%	100	_	$\operatorname{Covered}$
Cross #cross3#	100.0070			
Cross #cross_3#			_	
covered/total bins:	1	1	_	
<pre>covered/total bins: missing/total bins:</pre>	$\begin{matrix} 1 \\ 0 \end{matrix}$	1 1	_	
<pre>covered/total bins: missing/total bins:</pre>	$\begin{matrix} 1 \\ 0 \end{matrix}$	1 1	_ _ _	
<pre>covered/total bins: missing/total bins: % Hit:</pre>	$1\\0\\100.00\%$	$1\\1\\100$	_ _ _	Covered
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross4#</pre>	$\begin{matrix} 1 \\ 0 \end{matrix}$	$1 \\ 1 \\ 100 \\ 100$	- - -	$\mathbf{Covered}$
<pre>covered/total bins: missing/total bins: % Hit:</pre>	$1\\0\\100.00\%$	$1\\1\\100$	_ _ _ _	Covered
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross_4#   covered/total bins:</pre>	$1\\0\\100.00\%\\100.00\%$	$1 \\ 1 \\ 100 \\ 100 \\ 1$	_ _ _ _ _	Covered
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross4#   covered/total bins:   missing/total bins:</pre>	$1\\0\\100.00\%\\100.00\%\\1\\0$	$egin{array}{c} 1 \\ 1 \\ 100 \\ 100 \\ 1 \\ 1 \end{array}$	- - - -	Covered
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross4#   covered/total bins:   missing/total bins:   % Hit:</pre>	$1\\0\\100.00\%\\100.00\%\\1\\0\\100.00\%$	$egin{array}{c} 1 \\ 100 \\ 100 \\ 1 \\ 1 \\ 100 \\ \end{array}$	- - - - -	
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross4#   covered/total bins:   missing/total bins:</pre>	$1\\0\\100.00\%\\100.00\%\\1\\0$	$egin{array}{c} 1 \\ 1 \\ 100 \\ 100 \\ 1 \\ 1 \end{array}$	- - - - -	Covered Covered
<pre>covered/total bins:    missing/total bins:    % Hit: Cross #cross4#    covered/total bins:    missing/total bins:    % Hit: Cross #cross5#</pre>	$1\\0\\100.00\%\\100.00\%\\1\\0\\100.00\%$	1 100 100 1 1 1 100 100	- - - - - -	
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross_4#   covered/total bins:   missing/total bins:   % Hit: Cross #cross_5#   covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1$	1 100 100 1 1 1 100 100	- - - - - -	
<pre>covered/total bins:    missing/total bins:    % Hit: Cross #cross4#    covered/total bins:    missing/total bins:    % Hit: Cross #cross5#    covered/total bins:    missing/total bins:    missing/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0$	1 100 100 1 1 1 100 100	- - - - - - -	
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross_4#   covered/total bins:   missing/total bins:   % Hit: Cross #cross_5#   covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1$	1 100 100 1 1 1 100 100	- - - - - - - -	
<pre>covered/total bins:   missing/total bins:   % Hit: Cross #cross4#   covered/total bins:   missing/total bins:   % Hit: Cross #cross5#   covered/total bins:   missing/total bins:   % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\%$	1 100 100 1 1 1 100 100 1	- - - - - - -	$\operatorname{Covered}$
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0$	1 100 100 1 1 1 100 100 1 100 100	- - - - - - - -	
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\%$	1 100 100 1 1 1 100 100 1	- - - - - - - - -	$\operatorname{Covered}$
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\%$	1 100 100 1 1 1 100 100 1 100 100	- - - - - - - - -	$\operatorname{Covered}$
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0$	1 100 100 1 1 1 100 100 1 100 100 1	- - - - - - - - -	$\operatorname{Covered}$
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 100 100 1 1 1 100 100 1 100 100 100	- - - - - - - - - - - - -	Covered Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0$	1 100 100 1 1 1 100 100 1 1 100 100 100	- - - - - - - - - - - - - - - - - - -	$\operatorname{Covered}$
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 100 100 1 1 1 100 100 1 100 100 100		Covered Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     % Hit: Cross #cross7#     covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0$	1 100 100 100 1 1 100 100 100 1 1 100 100 100	- - - - - - - - - - - - - - - - - - -	Covered Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/t</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 100 100 100 1 1 100 100 100 1 1 100 100 100		Covered Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 100 100 100 1 1 100 100 100 1 1 100 100 100 1		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 100 100 100 1 1 100 100 100 1 1 100 100 100		Covered Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 80.76\%$	1 100 100 1 1 1 100 100 1 1 1 100 100 1		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     % Hit: Cross #cross8#     covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21$	1 100 100 1 1 1 100 100 100 100 1 1 1 1		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     missi</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5$	1 100 100 1 1 1 100 100 1 1 1 100 100 1		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     missi</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5$	1 100 100 100 1 1 100 100 100 100 11 1 100 100 100 26 26		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21$	1 100 100 1 1 1 100 100 100 100 1 1 1 1		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     missi</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21$	1 100 100 100 1 1 100 100 100 100 100 1		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 0 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\%$	1 1 100 100 1 1 1 100 100 100 1 1 1 100 100 100 26 26 100		Covered  Covered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 0 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\%$	1 1 100 100 1 1 1 100 100 100 1 1 1 100 100 100 26 26 100		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93$	1 100 100 100 1 1 1 100 100 100 1 1 1 1		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5$	1 100 100 100 1 1 100 100 100 1 1 100 100 100 100 26 26 26 100		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93$	1 100 100 100 1 1 1 100 100 100 1 1 1 1		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: overgroup instance \/ALSU_pkg::alsu_rand_class::cg covered/total bins: missing/total bins: missing/total bins: missing/total bins: Mit:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\%$	1 100 100 100 1 1 100 100 100 100 11 1 100 100 100 26 26 100 100 98 98		Covered  Covered  Uncovered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins: missing/total bins: missing/total bins: missing/total bins: Coverpoint rst	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\$	1 100 100 100 1 1 100 100 100 1 1 100 100 100 100 26 26 100 100 98 98 98		Covered  Covered  Uncovered
<pre>covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:     % Hit: Coverpoint rst     covered/total bins:</pre>	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 0 \\ 100.00\% \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 2$	1 1 100 100 1 1 1 100 100 100 1 1 1 100 100 100 26 26 100 100 98 98 98 100 100 2		Covered  Covered  Uncovered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins: missing/total bins: missing/total bins: missing/total bins: Coverpoint rst	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\$	1 100 100 100 1 1 100 100 100 1 1 100 100 100 100 26 26 100 100 98 98 98		Covered  Covered  Uncovered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covered/total bins:     % Hit:  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 2 \\ 0$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 20 2		Covered  Covered  Uncovered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  overgroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Coverpoint rst     covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 2 \\ 0 \\ 100.00\% \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 100 100 98 98 98 100 100 22 2 100		Covered  Covered  Uncovered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 2 2 100 1		Covered  Covered  Uncovered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 2 \\ 1 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 2 \\ 0 \\ 100.00\% \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 100 100 98 98 98 100 100 22 2 100		Covered  Covered  Uncovered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268 \\ 18$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 2 2 100 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 20 \\ 80.76\% \\ 20$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 100 26 26 26 100 100 20 1 1 1 100 100 21 1 1 100 100	- - - - -	Covered  Covered  Uncovered  Covered  Covered
covered/total bins: missing/total bins: % Hit:  Cross #cross4# covered/total bins: missing/total bins: % Hit:  Cross #cross5# covered/total bins: missing/total bins: % Hit:  Cross #cross6# covered/total bins: missing/total bins: % Hit:  Cross #cross7# covered/total bins: missing/total bins: missing/total bins: % Hit:  Cross #cross8# covered/total bins: missing/total bins: % Hit:  overgroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins: missing/total bins: % Hit:  Coverpoint rst covered/total bins: missing/total bins: missing/total bins: % Hit:  Coverpoint rst covered/total bins: missing/total bins: missing/total bins: missing/total bins: missing/total bins: covered/total bins: missing/total bins: missing/total bins: missing/total bins: missing/total bins: covered/total bins: missing/total bins: missing/total bins: missing/total bins: covered/total bins: missing/total bins: missing/total bins: missing/total bins: missing/total bins: covered/total bins: missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 20 \\ 80.76\% \\ 20$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 100 1 1 1 100 100 26 26 100 100 26 26 100 100 2 2 100 1 1 1 100 2 2 100 2	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 20 \\ 80.76\% \\ 20$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 100 26 26 26 100 100 20 1 1 1 100 100 21 1 1 100 100	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  Covergoup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$ \begin{array}{c} 1\\ 0\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 21\\ 5\\ 80.76\%\\ 21\\ 80.76\%\\ 20.00\%\\ $	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 2 2 1 100 100 2 2 2 100 1 1 100 2 2 2 2	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered
covered/total bins: missing/total bins: % Hit: Cross #cross4# covered/total bins: missing/total bins: % Hit: Cross #cross5# covered/total bins: missing/total bins: missing/total bins: % Hit: Cross #cross6# covered/total bins: missing/total bins: % Hit: Cross #cross7# covered/total bins: missing/total bins: % Hit: Cross #cross8# covered/total bins: missing/total bins: % Hit: covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins: missing/total bins: % Hit: Coverpoint rst covered/total bins: missing/total bins: % Hit: Coverpoint covered/total bins: missing/total bins: missing/total bins: % Hit: Coverpoint covered/total bins: missing/total bins: % Hit: bin auto[0] bin auto[1] Coverpoint cin covered/total bins: missing/total bins: missing/total bins: missing/total bins: missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 20 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 100.00\% \\ 20 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ 6 \\ $	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 2 2 100 1 1 1 100 2 2 100 2 1 1 100 2 2 100 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross7#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     % Hit:  Covergoup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$ \begin{array}{c} 1\\ 0\\ 100.00\%\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 21\\ 5\\ 80.76\%\\ 20\\ 80.76\%\\ 20\\ 80.76\%\\ 20\\ 80.70\%\\ 2$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 98 98 98 100 100 2 2 1 100 100 2 2 2 100 1 1 100 2 2 2 2	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     % Hit:  covered/total bins:     missing/total bins:	$ \begin{array}{c} 1\\ 0\\ 100.00\%\\ 100.00\%\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 1\\ 0\\ 100.00\%\\ 80.76\%\\ 21\\ 5\\ 80.76\%\\ 21\\ 60.00\%\\ 26\\ 8\\ 100.00\%\\ 26\\ 8\\ 100.00\%\\ 26\\ 8\\ 100.00\%\\ 26\\ 8\\ 100.00\%\\ 26\\ 100.00\%\\ 21\\ 100.00\%\\ 21\\ 100.00\%\\ 21\\ 100.00\%\\ 21\\ 100.00\%\\ 21\\ 21\\ 21\\ 31\\ 31\\ 31\\ 31\\ 31\\ 31\\ 31\\ 31\\ 31\\ 3$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 28 98 98 100 100 2 2 100 1 1 1 100 2 2 100 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:     % Hit:     bin auto[0]     bin auto[1] Coverpoint cin     covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 21$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 28 98 98 100 100 2 2 100 1 1 1 100 2 2 100 1 1 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Overgroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 242 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 140 \\$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 26 26 26 100 100 20 2 1 1 1 100 20 2 1 100 1 1 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross7#     covered/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     % Hit: covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:     % Hit:     bin auto[0]     bin auto[1] Coverpoint cin     covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 21$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 28 98 98 100 100 2 2 100 1 1 1 100 2 2 100 1 1 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit: Cross #cross4#     covered/total bins:     missing/total bins:     % Hit: Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit: Covergroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:     bin auto[0]     bin auto[1] Coverpoint cin     covered/total bins:     missing/total bins:     more auto[0]     bin auto[1] Coverpoint red_op_A_cp     covered/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 44 \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 21 \\ 44 \\ 100.00\% \\ 22 \\ 0 \\ 100.00\% \\ 24 \\ 24 \\ 144 \\ 100.00\% \\ 25 \\ 27 \\ 27 \\ 30 \\ 30 \\ 30 \\ 30 \\ 30 \\ 30 \\ 30 \\ 3$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 100 1 1 1 100 100 26 26 100 100 28 98 98 100 100 20 1 1 1 100 2 2 100 1 1 1 100 2 2 100 1 1 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered  Covered
covered/total bins:     missing/total bins:     % Hit:  Cross #cross4#     covered/total bins:     missing/total bins:     % Hit:  Cross #cross5#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross6#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Cross #cross8#     covered/total bins:     missing/total bins:     missing/total bins:     missing/total bins:     % Hit:  Overgroup instance \/ALSU_pkg::alsu_rand_class::cg  covered/total bins:     missing/total bins:	$1 \\ 0 \\ 100.00\% \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 1 \\ 0 \\ 100.00\% \\ 21 \\ 5 \\ 80.76\% \\ 21 \\ 5 \\ 80.76\% \\ 99.12\% \\ 93 \\ 5 \\ 94.89\% \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 268 \\ 18 \\ 100.00\% \\ 242 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 142 \\ 144 \\ 100.00\% \\ 140 \\$	1 1 100 100 100 1 1 1 100 100 1 1 1 100 100 1 1 1 100 100 100 26 26 26 100 100 26 26 26 100 100 20 2 1 1 1 100 20 2 1 100 1 1 1 1	- - - - -	Covered  Covered  Uncovered  Covered  Covered  Covered  Covered  Covered  Covered  Covered

% Hit:	100.00%	100	_	
bin red_op_A_0	55	1	_	Covered
bin red_op_A_1	231	1	_	Covered
default bin red_op_A_default	0		_	ZERO
Coverpoint red_op_B_cp	100.00%	100	_	Covered
covered/total bins:	2	2	_	00,0104
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
bin $red_{op}B_0$	47	1		Covered
			_	Covered
bin red_op_B_1	239	1	_	
default bin red_op_B_default	0	100	_	ZERO
Coverpoint bypass_A	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
bin auto [0]	284	1	_	Covered
bin auto[1]	2	1	_	Covered
Coverpoint bypass_B	100.00%	100	_	Covered
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
$ \text{bin auto} \left[  0  \right]$	281	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]	146	1	_	Covered
bin auto[1]	140	1	_	Covered
Coverpoint serial_in	100.00%	100	_	Covered
covered/total bins:	2	$\frac{100}{2}$	_	20.0104
missing/total bins:	0	$\frac{2}{2}$	_	
			_	
% Hit:	100.00%	100	_	Comons
bin auto [0]	149	1	_	Covered
bin auto[1]	137	1	_	Covered
$\operatorname{Coverpoint} A_{\operatorname{-cp}}$	100.00%	100	_	Covered
covered/total bins:	5	5	_	
missing/total bins:	0	5	_	
% Hit:	100.00%	100	_	
$bin A_data_0$	74	1	_	Covered
bin A_data_max	37	1	_	Covered
bin A_data_min	37	1	_	Covered
bin A_data_walkingones[1]	31	1	_	Covered
bin A_data_walkingones [2]	32	1	_	Covered
default bin A_data_default	75		_	Occurred
Coverpoint $B_{-}cp$	100.00%	100	_	Covered
covered/total bins:	5	5	_	Covered
missing/total bins:	0	5		
% Hit:	100.00%	100		
bin B_data_0	53	1	_	Covered
bin B_data_max				Covered
	41	1	_	
bin B_data_min	44	1	_	Covered
bin B_data_walkingones[1]	38	1	_	Covered
bin B_data_walkingones[2]	33	1	_	Covered
default bin B_data_default	77		_	Occurred
$\operatorname{Coverpoint} A\_\operatorname{cp\_mod}$	100.00%	100	_	Covered
covered/total bins:	3	3	_	
missing/total bins:	0	3	_	
% Hit:	100.00%	100	_	
$bin A_data_0$	74	1	_	Covered
bin A_data_max	37	1	_	Covered
bin A_data_min	37	1	_	Covered
$\operatorname{Coverpoint}\ \operatorname{B\_cp\_mod}$	100.00%	100	_	Covered
covered/total bins:	3	3	_	
missing/total bins:	0	3	_	
% Hit:	100.00%	100	_	
bin B_data_0	53	1	_	Covered
bin B_data_max	41	1	_	Covered
bin B_data_min	44	1	_	Covered
Coverpoint ALU_cp	100.00%	100	_	Covered
covered/total bins:	8	8	_	· · - <del></del>
missing/total bins:	0	8	_	
% Hit:	100.00%	100	_	
bin Bins_shift [SHIFT_4]	47	100	_	Covered
bin Bins_shift [ROTATE.5]	47 61	1		Covered
i i		1 1	_	Covered Covered
bin Bins_arith [ADD_2]	49 52	1	_	
bin Bins_arith [MUL_3]	53	1	_	Covered
bin Bins_bitwise [OR_0]	29	1	_	Covered
bin Bins_bitwise [XOR_1]	33	1	_	Covered
bin Bins_invalid	14	1	_	Covered
bin Bins_trans	1	1	_	Covered
Cross #cross0#	100.00%	100	_	Covered
covered/total bins:		8	_	
missing/total bins:	8			
	8 0	8	_	
% Hit:	8		_ _	
% Hit: Auto, Default and User Defined Bins:	8 0	8	_	
% Hit:	8 0	8	_ _ _	Covered
% Hit: Auto, Default and User Defined Bins:	$8 \\ 0 \\ 100.00\%$	8 100	_ _ _	Covered Covered
% Hit: Auto, Default and User Defined Bins: bin <a_data_walkingones[2], red_op_a_0=""></a_data_walkingones[2],>	$\begin{array}{c} 8 \\ 0 \\ 100.00\% \end{array}$	8 100 1	- - - -	
% Hit: Auto, Default and User Defined Bins: bin <a_data_walkingones[2], red_op_a_0=""> bin <a_data_walkingones[1], red_op_a_0=""></a_data_walkingones[1],></a_data_walkingones[2],>	$\begin{array}{c} 8 \\ 0 \\ 100.00\% \\ \end{array}$	8 100 1 1	- - - - -	Covered
% Hit: Auto, Default and User Defined Bins: bin <a_data_walkingones[2], red_op_a_0=""> bin <a_data_walkingones[1], red_op_a_0=""> bin <a_data_min, red_op_a_1=""> bin <a_data_min, red_op_a_0=""></a_data_min,></a_data_min,></a_data_walkingones[1],></a_data_walkingones[2],>	$   \begin{array}{c}     8 \\     0 \\     100.00\%   \end{array} $ $   \begin{array}{c}     4 \\     3 \\     31   \end{array} $	8 100 1 1 1	- - - - -	Covered Covered
% Hit: Auto, Default and User Defined Bins: bin <a_data_walkingones [2],="" red_op_a_0=""> bin <a_data_walkingones [1],="" red_op_a_0=""> bin <a_data_min, red_op_a_1=""> bin <a_data_min, red_op_a_0=""> bin <a_data_max, red_op_a_1=""></a_data_max,></a_data_min,></a_data_min,></a_data_walkingones></a_data_walkingones>	$   \begin{array}{c}     8 \\     0 \\     100.00\%   \end{array} $ $   \begin{array}{c}     4 \\     3 \\     31 \\     6 \\     34   \end{array} $	8 100 1 1 1 1	- - - - - -	Covered Covered Covered Covered
% Hit: Auto, Default and User Defined Bins: bin <a_data_walkingones[2], red_op_a_0=""> bin <a_data_walkingones[1], red_op_a_0=""> bin <a_data_min, red_op_a_1=""> bin <a_data_min, red_op_a_0=""></a_data_min,></a_data_min,></a_data_walkingones[1],></a_data_walkingones[2],>	$   \begin{array}{c}     8 \\     0 \\     100.00\%   \end{array} $ $   \begin{array}{c}     4 \\     3 \\     31 \\     6   \end{array} $	8 100 1 1 1 1 1	- - - - -	Covered Covered Covered

```
bin < A_data_max, red_op_A_0 >
                                                              3
                                                                           1
                                                                                             Covered
                                                             29
                                                                           1
        bin < A_data_0, red_op_A_0 >
                                                                                             Covered
    Illegal and Ignore Bins:
                                                                                             Occurred
        ignore_bin assert_red_op_A
                                                             56
                                                        100.00\%
                                                                         100
Cross #cross__1#
                                                                                             Covered
    covered/total bins:
                                                             18
                                                                          18
    missing/total bins:
                                                              0
                                                                          18
                                                        100.00\%
   % Hit:
                                                                         100
    Auto, Default and User Defined Bins:
        bin <B_data_walkingones[2], red_op_A_1, red_op_B_1>
                                                                           1
                                                                                             Covered
        bin <B_data_walkingones [2], red_op_A_1, red_op_B_0>
                                                                           1
                                                                                             Covered
        bin <B_data_walkingones[1], red_op_A_1, red_op_B_1>
                                                                           1
                                                                                             Covered
        bin <B_data_walkingones[1], red_op_A_1, red_op_B_0>
                                                                                             Covered
                                                                           1
        bin <B_data_min, red_op_A_1, red_op_B_1>
                                                             31
                                                                                             Covered
                                                                           1
        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} >
                                                             37
                                                                                             Covered
                                                                           1
        bin < B_data_0, red_op_A_1, red_op_B_1 >
                                                             35
                                                                                             Covered
                                                                           1
        bin <B_data_max, red_op_A_1, red_op_B_0>
                                                                                             Covered
                                                              1
                                                                           1
        bin < B_data_0, red_op_A_1, red_op_B_0 >
                                                             14
                                                                                             Covered
                                                                           1
        bin <B_data_walkingones[2], red_op_A_0, red_op_B_0>
                                                                           1
                                                                                             Covered
        bin <B_data_walkingones[1], red_op_A_0, red_op_B_0>
                                                                                             Covered
                                                                           1
        \label{eq:bin} \begin{array}{ll} \mathrm{bin} \ <\! \mathrm{B\_data\_min} \ , \\ \mathrm{red\_op\_A\_0} \ , \\ \mathrm{red\_op\_B\_1} \! > \\ \end{array}
                                                                                             Covered
                                                                           1
        bin <B_data_min, red_op_A_0, red_op_B_0>
                                                              3
                                                                                             Covered
                                                                           1
        bin <B_data_max, red_op_A_0, red_op_B_1>
                                                              2
                                                                           1
                                                                                             Covered
        bin < B_data_0, red_op_A_0, red_op_B_1>
                                                                                             Covered
                                                                           1
        bin <B_data_max, red_op_A_0, red_op_B_0>
                                                                                             Covered
                                                              1
                                                                           1
                                                                                             Covered
        bin < B_data_0, red_op_A_0, red_op_B_0 >
                                                              2
                                                                           1
    Illegal and Ignore Bins:
        ignore_bin assert_red_op_B
                                                             15
                                                                                             Occurred
Cross #cross__2#
                                                        100.00\%
                                                                         100
                                                                                             Covered
    covered/total bins:
                                                              1
                                                                           1
    missing/total bins:
                                                              0
                                                                           1
   % Hit:
                                                        100.00\%
                                                                         100
    Auto, Default and User Defined Bins:
        bin arith_permutations
                                                                           1
                                                                                             Covered
                                                        100.00\%
                                                                                             Covered
Cross #cross_3#
                                                                         100
    covered/total bins:
                                                              1
                                                                           1
    missing/total bins:
                                                              0
                                                                           1
   % Hit:
                                                        100.00\%
                                                                         100
    Auto, Default and User Defined Bins:
                                                                                             Covered
        bin addition_cin
                                                             49
                                                                           1
                                                        100.00\%
Cross #cross__4#
                                                                         100
                                                                                             Covered
    covered/total bins:
                                                              1
                                                                           1
    missing/total bins:
                                                                           1
   % Hit:
                                                        100.00\%
                                                                         100
    Auto, Default and User Defined Bins:
        bin shift_rotate_direction
                                                            108
                                                                                             Covered
Cross #cross_5#
                                                        100.00\%
                                                                         100
                                                                                             Covered
    covered/total bins:
                                                              1
                                                                           1
    missing/total bins:
                                                                           1
   % Hit:
                                                        100.00\%
                                                                         100
    Auto, Default and User Defined Bins:
        bin shift_serial_in
                                                             47
                                                                           1
                                                                                             Covered
Cross #cross__6#
                                                        100.00\%
                                                                         100
                                                                                             Covered
    covered/total bins:
                                                              1
                                                                           1
    missing/total bins:
                                                              0
                                                                           1
                                                        100.00\%
   % Hit:
                                                                         100
    Auto, Default and User Defined Bins:
        bin or_xor_red_op_A
                                                                           1
                                                                                             Covered
Cross #cross__7#
                                                        100.00\%
                                                                         100
                                                                                             Covered
    covered/total bins:
                                                                           1
                                                              1
    missing/total bins:
                                                              0
                                                                           1
                                                        100.00\%
   % Hit:
                                                                         100
    Auto, Default and User Defined Bins:
        bin or_xor_red_op_B
                                                             11
                                                                                             Covered
Cross #cross__8#
                                                         80.76\%
                                                                         100
                                                                                             Uncovered
   covered/total bins:
    missing/total bins:
                                                              5
                                                                          ^{26}
                                                         80.76\%
                                                                         100
   % Hit:
    Auto, Default and User Defined Bins:
        bin <Bins_trans, red_op_A_1, red_op_B_0>
                                                              1
                                                                           1
                                                                                             Covered
        bin <Bins_invalid , red_op_A_1 , red_op_B_1>
                                                                                             Covered
                                                              3
                                                                           1
        bin <Bins_invalid , red_op_A_1 , red_op_B_0>
                                                                                             Covered
                                                              4
                                                                           1
        bin <Bins_invalid ,red_op_A_0 ,red_op_B_1>
                                                                                             Covered
                                                              4
                                                                           1
        bin <Bins_invalid , red_op_A_0 , red_op_B_0>
                                                              3
                                                                                             Covered
                                                                           1
        bin <Bins_bitwise [XOR_1], red_op_A_0, red_op_B_0>
                                                                                             Covered
        bin <Bins_bitwise [OR_0], red_op_A_0, red_op_B_0>
                                                                                             Covered
        bin <Bins_arith [MUL_3], red_op_A_1, red_op_B_1>
                                                                                             Covered
                                                              47
                                                                           1
        bin <Bins_shift [ROTATE_5], red_op_A_1, red_op_B_1>
```

	56	1	_	Covered
bin <bins_arith [mul_3],="" red_op_a_1,="" red_op_b_0=""></bins_arith>	4	1		Covered
bin <bins_shift [rotate.5],="" red_op_a_1,="" red_op_b<="" td=""><td></td><td>1</td><td>_</td><td>Covered</td></bins_shift>		1	_	Covered
	4	1	_	Covered
bin <bins_arith [add_2],="" red_op_a_1,="" red_op_b_1=""></bins_arith>	> 44	1	_	Covered
bin <bins_shift[shift_4], red_op_a_1,="" red_op_b_<="" td=""><td></td><td>1</td><td></td><td>Covered</td></bins_shift[shift_4],>		1		Covered
	43	1	_	Covered
bin <bins_arith [add.2],="" red_op_a_1,="" red_op_b_0=""></bins_arith>	1	1	_	Covered
bin <bins_shift [shift_4],="" red_op_a_1,="" red_op_b_<="" td=""><td></td><td>1</td><td></td><td>Covered</td></bins_shift>		1		Covered
	2	1	_	Covered
bin <bins_arith [mul_3],="" red_op_a_0,="" red_op_b_1=""></bins_arith>	> 1	1	_	Covered
bin <bins_shift ,="" [rotate_5]="" red_op_a_0="" red_op_b<="" td=""><td>B<sub>-</sub>1&gt;</td><td>1</td><td></td><td>covered</td></bins_shift>	B <sub>-</sub> 1>	1		covered
him (Ding swith [MIII 9] and an A O and an D Os	1	1	_	Covered
bin <bins_arith [mul_3],="" red_op_a_0,="" red_op_b_0=""></bins_arith>	1	1	_	Covered
$\label{eq:bins_arith} bin <& Bins_arith [ADD.2], red_op_A_0, red_op_B_1> \\$	>			
bin <bins_shift[shift_4], red_op_a_0,="" red_op_b_<="" td=""><td>4</td><td>1</td><td>_</td><td>Covered</td></bins_shift[shift_4],>	4	1	_	Covered
om Coms_smit[Smr1_4], red_op_A_o, red_op_B_	1	1	_	Covered
$\label{eq:bins_shift} bin < Bins_shift [SHIFT_4], red_op_A_0, red_op_B_0$	<0>			
bin <bins_trans, red_op_a_0,*=""></bins_trans,>	$\frac{1}{0}$	1 1	$\frac{-}{2}$	Covered ZERO
bin <bins_trans,*,red_op_b_1></bins_trans,*,red_op_b_1>	0	1	$\frac{2}{2}$	ZERO
bin <bins_shift [rotate.5],="" red_op_a_0,="" red_op_b<="" td=""><td><math>B_{-}0 &gt;</math></td><td></td><td></td><td></td></bins_shift>	$B_{-}0 >$			
him (Ding swith [ADD 0] and an A 0 and an D 0	0	1	1	ZERO
bin <bins_arith [add_2],="" red_op_a_0,="" red_op_b_0=""></bins_arith>	0	1	1	ZERO
Illegal and Ignore Bins:	-			
ignore_bin invalid_reduction	47		_	Occurred

TOTAL COVERGROUP COVERAGE: 99.12% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 99.12%

## 1.8 8. code Coverage Report

Coverage Report by DU with details

	Coverage:		Bin	s Hits	Misses	Coverage
Br	anches		33	2 32	0	$\overline{100.00\%}$
			Brancl	n Details====		
Branch	Coverage	for Des	sign Unit work.	ALSU		
Liı	ne	Item		Count	Source	, -
File	ALSU.sv					
			II	F Branch———		
24				1016		coming in to IF
24		1		4		est) begin
35		1		1012	$_{ m end}$	else begin
Branch	totals: 2	hits c	of 2 branches =	100.00%		
			II	F Branch———		
51				1019	$\operatorname{Count}$	coming in to IF
51		1		6	if (r	est) begin
53		1		1013	$\operatorname{end}$	else begin
Branch	totals: 2	hits c	of 2 branches =	100.00%		
			II	F Branch———		
54				1013	$\operatorname{Count}$	coming in to IF
54		1		784		if (invalid)
56		1		229		else
Branch	totals: 2	hits c	of 2 branches =	100.00%		
			II	F Branch———		
63				984		coming in to IF
63		1		4	if (r	est) begin
66		1		980	else	begin
Branch	totals: 2	hits o	of 2 branches =	100.00%		
			II	F Branch——		
67				980		coming in to IF
67		1		51	i f	(bypass_A_reg && bypass_B_reg
69		1		200		se if (bypass_A_reg)
71		1		182		se if (bypass_B_reg)
73		1		418	el	se if (invalid)
75		1		129	el	se begin
Branch	totals: 5	hits c	of 5 branches =	100.00%		
			С	ASE Branch—		

76 77 87 97 98 99 105 Branch totals: 7 l	1 1 1 1 1 1 hits of	7 branches =		129 35 38 10 18 15 11 2	Count All F		3'h0: 3'h1: 3'h2: 3'h3: 3'h4: 3'h5:	begin out <= A_r out <= A_r begin	eg + B_reg; eg * B_reg;
78 78 80 82 84 Branch totals: 4 l	1 1 1 1 hits of	4 branches :	IF Branch = 100.00% IF Branch	35 5 7 13 10	Count	comi	if (else	red_op_A_r if (red_o if (red_o	
88 88 90 92 94 Branch totals: 4 l	1 1 1 1 hits of	4 branches =		38 6 8 14 10	Count		if (else else	red_op_A_r if (red_o if (red_o	
100 100 102 Branch totals: 2 l	1 1 hits of		= 100.00% IF Branch		Count		if (else	to IF	
106 108 Branch totals: 2 l  Condition Coverage Enabled Covera	e :	2 branches :			Misses	Cov	else	direction_	reg)
Condition Coverage	e for D		dition Det						
Line 67 Item Condition totals:	n 1 2 of 2 Covered	Reason for	eg && byp covered	pass_B_r = 100.00					
Line 67 Item Condition totals:  Input Term bypass_A_reg bypass_B_reg	n 1 2 of 2 Covered Y	(bypass_A_reinput terms  Reason for	eg && byp covered no cover	pass_B_r = 100.00 age H	Iint				
Line 67 Item Condition totals:  Input Term bypass_A_reg bypass_B_reg	1 2 of 2  Covered  Y Y Y  Hits F 1 b 1 b 1 b	(bypass_A_reinput terms  Reason for	eg && byp covered no cover	pass_B_r = 100.00	$\frac{1}{2}$ Sking constant $\frac{1}{2}$ B_reg $\frac{1}{2}$	ondit	ion (s)		
Line 67 Item Condition totals:  Input Term bypass_A_reg bypass_B_reg  Rows: Row 1: Row 2: Row 3: Row 4:	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b cocused 1	(bypass_A_reinput terms  Reason for  EC Target  ypass_A_reg_ ypass_B_reg_ ypass_B_reg_ Condition Vic (red_op_A_red)	eg && byp covered no cover  0 1 0 1 ew— eg && red	Non-masbypassbypassbypass-	$\begin{array}{c} \text{Sking co} \\ \text{B\_reg} \\ \text{A\_reg} \\ \text{A\_reg} \\ \text{eg} \end{array}$	ondit	ion(s)		
Line 67 Item Condition totals:  Input Term Condition totals:  Input Term Condition totals:  Row 1: Row 2: Row 3: Row 4:  Line 78 Item Condition totals:	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b 2 ocused 1 2 of 2  Covered	(bypass_A_reinput terms  Reason for  EC Target  ypass_A_reg_ ypass_B_reg_ ypass_B_reg_ cypass_B_reg_ input terms  Reason for	eg && byp covered  no cover  0 1 0 1 ew eg && red covered	Non-mas bypass bypass bypass	$\begin{array}{c} \text{Sking co} \\ \text{B\_reg} \\ \text{A\_reg} \\ \text{A\_reg} \\ \text{eg} \end{array}$	ondit	ion(s)		
Line 67 Item Condition totals:  Input Term 6 bypass_A_reg bypass_B_reg  Rows: If Row 1: Row 2: Row 3: Row 4:  Line 78 Item Condition totals:  Input Term 6 red_op_A_reg red_op_B_reg	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b Covered  1 covered  Y Y  Y	Reason for  Reason for  EC Target  Typass_A_reg_ Typass_B_reg_ Typass_B_	eg && byp covered no cover  0 1 0 1 ew eg && red covered no cover	Non-mas bypass_ bypass_ bypass_ bypass_ 100.00 age H	$\frac{\text{Sking cos}}{\text{B_reg}}$ $\frac{\text{B_reg}}{\text{A_reg}}$ $\frac{\text{A_reg}}{\text{O}\%}$ $\text{Hint}$				
Line 67 Item Condition totals:  Input Term 6 bypass_A_reg bypass_B_reg  Rows: If Row 1: Row 2: Row 3: Row 4:  Line 78 Item Condition totals:  Input Term 6 red_op_A_reg red_op_B_reg	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b 1 coused 1 2 of 2  Covered  Y Y  Hits F  1 r 1 r 1 r	(bypass_A_reinput terms  Reason for  EC Target  ypass_A_reg_ ypass_B_reg_ ypass_B_reg_ Condition Vic (red_op_A_reinput terms  Reason for	eg && byp covered no cover  0 1 0 1 ew eg && red covered no cover	Non-mas bypass bypass bypass	Sking   Colored   Sking   Co				
Line 67 Item Condition totals:  Input Term 6 bypass_A_reg bypass_B_reg  Rows: Input Row 1: Row 2: Row 3: Row 4:  Line 78 Item Condition totals:  Input Term 6 red_op_A_reg red_op_B_reg  Rows: Input Row 2: Row 3: Row 4:	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b 1 b 1 c Y Y  Hits F  1 r 1 r 1 r 1 r 1 r 1 r 1 r 1 r 1 r 1	Reason for  EC Target  Typass_A_reg_ Typass_A_reg_ Typass_B_reg_ Typass_	eg && byp covered no cover  0 1 0 1 ew eg && red covered no cover  0 1 0 1	Non-mas	Sking   Colored				
Line 67 Item Condition totals:  Input Term 6 bypass_A_reg bypass_B_reg  Rows: Input Row 2: Row 3: Row 4:  Line 78 Item Condition totals:  Input Term 6 red_op_A_reg red_op_B_reg  Rows: Input Row 2: Row 3: Row 4:  Line 88 Item Condition totals:	1 2 of 2  Covered  Y Y  Hits F  1 b 1 b 1 b 1 b 1 b 1 c Y Y  Hits F  1 r 1 r 1 r 1 r 1 r 1 r 1 r 1 r 1 r 1	(bypass_A_reinput terms  Reason for  EC Target  ypass_A_reg_ ypass_B_reg_ ypass_B_reg_ ypass_B_reg_ Tondition Vic (red_op_A_reinput terms)  EC Target  EC Target  ed_op_A_reg_ ed_op_B_reg_ cd_op_B_reg_ ed_op_B_reg_ condition Vic (red_op_A_reinput terms)	eg && byp covered no cover  0 1 0 1 ew eg && red covered 1 0 1 0 1 covered 1 covered 1 covered	Non-mas bypass_ bypass_ bypass_ bypass_ red_op_ red_op_ red_op_ red_op_ = 100.00	Sking   Colored				
Line 67 Item Condition totals:  Input Term 6 bypass_A_reg bypass_B_reg  Rows: Input Row 1: Row 2: Row 3: Row 4:  Input Term 6 Condition totals:  Input Term 6 red_op_A_reg red_op_B_reg  Rows: Input Row 2: Row 3: Row 4:  Input Term 6 Row 1: Row 2: Row 3: Row 4:  Input Row 2: Row 3: Row 4:  Input Row 3: Row 4:  Input Row 3: Row 4:  Input Row 3: Row 4:	Covered  Y Y  Hits F  1 b 1 b 1 b 1 b 1 c 2 of 2  Covered  Y Y  Hits F  1 r 1 r 1 r 1 r 1 r 2 of 2	Reason for  EC Target  Typass_A_reg_ Typass_B_reg_ Typass_	eg && byp covered no cover  0 1 0 1 ew eg && red covered 1 0 1 0 1 covered 1 covered 1 covered	Non-mas bypass_ bypass_ bypass_ bypass_ red_op_ red_op_ red_op_ red_op_ = 100.00	Sking   Co				

```
Row
        4:
                     1
                        red_op_B_reg_1
                                                red_op_A_reg
Expression Coverage:
    Enabled Coverage
                                            Covered
                                    Bins
                                                        Misses
                                                                Coverage
                                       8
                                                             0
                                                                 100.00\%
    Expressions
                                  Expression Details
Expression Coverage for Design Unit work.ALSU —
  File ALSU.sv
                 -Focused Expression View-
           18 Item
                       1 ((red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]))
Line
Expression totals: 4 of 4 input terms covered = 100.00%
                            Reason for no coverage
     Input Term
                   Covered
                         Y
   red_op_A_reg
  red_op_B_reg
                         Y
  opcode_reg[1]
                         Y
  opcode_reg[2]
                         Y
                        FEC Target
     Rows:
                  _{
m Hits}
                                                Non-masking condition(s)
  Row
        1:
                     1
                        red_op_A_reg_0
                                                ((opcode_reg[1]
                                                                    opcode_reg[2]) && ~red_op_B_reg)
                        red_op_A_reg_1
                                                ((opcode_reg[1]
                                                                    opcode_reg[2]) && ~red_op_B_reg)
  Row
        2:
                     1
                        red_op_B_reg_0
                                                ((opcode_reg[1]
                                                                    opcode_reg[2]) && ~red_op_A_reg)
  Row
        3:
                     1
                                                ((opcode_reg[1]
                                                                    opcode_reg[2]) && ~red_op_A_reg)
                        red_op_B_reg_1
  Row
        4:
                     1
                                                                  red_op_B_reg) && ~opcode_reg[2])
  Row
                        opcode_reg[1]_0
                                                ((red_op_A_reg
        5:
                     1
                                                                  red_op_B_reg) && ~opcode_reg[2])
                                                ((red_op_A_reg
  Row
        6:
                     1
                        opcode_reg[1]_1
                                                                  red_op_B_reg) && ~opcode_reg[1])
  Row
        7:
                        opcode_reg[2]_0
                                                ((red_op_A_reg
                     1
                                                                  red_op_B_reg) && ~opcode_reg[1])
  Row
        8:
                     1
                        opcode_reg[2]_1
                                                ((red_op_A_reg
                 -Focused Expression View-
Line
           19 Item
                       1 (\operatorname{opcode\_reg}[1] \& \operatorname{opcode\_reg}[2])
Expression totals: 2 of 2 input terms covered = 100.00%
     Input Term
                   Covered
                            Reason for no coverage
                                                        Hint
                         Y
  opcode_reg[1]
  opcode_reg[2]
                         Y
                        FEC Target
     Rows:
                  _{
m Hits}
                                                Non-masking condition(s)
  Row
        1:
                        opcode_reg[1]_0
                                                opcode_reg[2]
  Row
                        opcode_reg[1]_1
                                                opcode_reg[2]
  Row
        3:
                        opcode_reg[2]_0
                                                opcode_reg[1]
                        opcode_reg[2]_1
                                                opcode_reg[1]
  Row
        4:
                 -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
                          Y
  invalid_red_op
  invalid_opcode
                          Y
     Rows:
                  Hits
                        FEC Target
                                                Non-masking condition(s)
                                                ~invalid_opcode
  Row
        1:
                        invalid_red_op_0
  Row
        2:
                        invalid_red_op_1
                                                ~invalid_opcode
                                                ~invalid_red_op
  Row
        3:
                        invalid_opcode_0
                        invalid_opcode_1
                                                ~invalid_red_op
  Row
        4:
Statement Coverage:
    Enabled Coverage
                                    _{\mathrm{Bins}}
                                               Hits
                                                        Misses Coverage
                                                                 100.00\%
                                      48
    Statements
                                                 48
                                  =Statement Details:
Statement Coverage for Design Unit work.ALSU —
    Line
                  Item
                                              Count
                                                         Source
  File ALSU.sv
                                                         module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, c
    1
    2
                                                         parameter INPUT_PRIORITY = "A";
    3
                                                         parameter FULL_ADDER = "ON";
```

red\_op\_A\_reg\_0

red\_op\_A\_reg\_1

red\_op\_B\_reg\_0

1

1

 $red_op_B_reg$ 

red\_op\_A\_reg

Row

Row

Row

4

5

6

7

8 9 2:

3:

input signed [2:0] A, B;

output reg signed [5:0] out;

output reg [15:0] leds;

input [2:0] opcode;

input clk, cin, rst, red\_op\_A, red\_op\_B, bypass\_A, bypass\_B, direction, serial\_in;

```
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;
                                                       reg signed [2:0] A_reg, B_reg;
13
14
15
                                                       wire invalid_red_op, invalid_opcode, invalid;
16
17
                                                       //Invalid handling
                 1
                                              463
                                                       assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg
18
                                                       assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
                 1
                                              444
19
                                              228
                 1
                                                       assign invalid = invalid_red_op | invalid_opcode;
20
21
22
                                                       //Registering input signals
                 1
                                             1016
                                                       always @(posedge clk or posedge rst) begin
23
                                                         if (rst) begin
24
                                                             \operatorname{cin\_reg} \; <= \; 0;
                 1
25
                                                4
                                                4
26
                 1
                                                            red_op_B_reg \ll 0;
                                                4
27
                 1
                                                            red_op_A_reg \ll 0;
28
                 1
                                                4
                                                            bypass_B_reg \ll 0;
                                                4
29
                 1
                                                            bypass_Areg \ll 0;
                                                4
30
                 1
                                                             direction_reg \ll 0;
                                                4
31
                 1
                                                             serial_in_reg \ll 0;
32
                 1
                                                4
                                                            opcode_reg \ll 0;
33
                 1
                                                4
                                                            A_reg \ll 0;
34
                 1
                                                4
                                                            B_reg \ll 0;
                                                         end else begin
35
                 1
                                             1012
36
                                                             cin\_reg \le cin;
                                             1012
37
                 1
                                                            red_op_B_reg <= red_op_B;
38
                 1
                                             1012
                                                            red_op_A_reg <= red_op_A;
39
                 1
                                             1012
                                                            bypass_B_reg <= bypass_B;
40
                 1
                                             1012
                                                            bypass_A_reg <= bypass_A;
                 1
                                             1012
                                                             direction_reg <= direction;
41
                 1
                                             1012
                                                             serial_in_reg <= serial_in;
42
                                                            opcode_reg <= opcode;
                 1
                                             1012
43
44
                 1
                                             1012
                                                            A_reg \ll A;
                 1
                                             1012
                                                            B_reg \ll B;
45
46
                                                         \operatorname{end}
                                                       end
47
48
49
                                                       //leds output blinking
50
                 1
                                             1019
                                                       always @(posedge clk or posedge rst) begin
                                                         if (rst) begin
51
                 1
                                                            leds \ll 0;
52
                                                6
53
                                                         end else begin
54
                                                              if (invalid)
55
                 1
                                              784
                                                                leds \ll ~leds;
56
                                                              else
                 1
                                              229
57
                                                                leds \ll 0;
58
                                                         end
59
                                                       \operatorname{end}
60
61
                                                       //ALSU output processing
                 1
                                              984
                                                       always @(posedge clk or posedge rst) begin
62
63
                                                         if (rst) begin
                 1
                                                4
                                                           out \leq 0;
64
65
                                                         \operatorname{end}
66
                                                         else begin
67
                                                            if (bypass_A_reg && bypass_B_reg)
68
                 1
                                               51
                                                             out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
69
                                                            else if (bypass_A_reg)
70
                 1
                                              200
                                                             out \leq A<sub>reg</sub>;
71
                                                            else if (bypass_B_reg)
72
                 1
                                              182
                                                             out \leq B<sub>reg</sub>;
73
                                                            else if (invalid)
74
                 1
                                              418
                                                                out \leq 0;
75
                                                            else begin
76
                                                                case (opcode)
77
78
                                                                     if (red_op_A_reg && red_op_B_reg)
                                                                       out <= (INPUT_PRIORITY == "A")? | A_reg: | B_reg;
79
                 1
                                                5
                                                                     else if (red_op_A_reg)
81
                 1
                                                7
                                                                      out \leq |A_reg|;
                                                                     else if (red_op_B_reg)
82
                                                                       out <= |B_reg;
                 1
                                               13
84
                                                                    _{
m else}
                 1
                                               10
                                                                       out <= A_reg | B_reg;
85
86
                                                                  end
                                                                  3'h1: begin
87
                                                                     if (red_op_A_reg && red_op_B_reg)
88
                                                                       out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
                 1
                                                6
89
                                                                     else if (red_op_A_reg)
90
91
                 1
                                                8
                                                                       out \leq ^A_reg;
                                                                    else if (red_op_B_reg)
92
                 1
                                               14
                                                                      out \leq ^B_reg;
93
                                                                    else
94
                                                                       out <= A_reg ^ B_reg;
                 1
                                               10
95
96
                                                                  end
97
                 1
                                               10
                                                                  3'h2: out \ll A_reg + B_reg;
                 1
                                               18
                                                                  3'h3: out \ll A_reg * B_reg;
98
99
                                                                  3'h4: begin
100
                                                                    if (direction_reg)
                 1
101
                                               11
                                                                      out \le {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		6		$out \le \{out[4:0], out[5]\};$
108					${ m else}$
109	1		5		$out <= \{out[0], out[5:1]\};$
Toggle Covera	ge:				
Enabled (	Coverage	$\operatorname{Bins}$	$_{ m Hits}$	Misses	Coverage
Toggles	<del></del>	118	118	0	$\overline{100.00\%}$

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
$\mathrm{B}[0\!-\!2]$	1	1	100.00
$\mathrm{B} \mathtt{\_reg} \left[ 0 - 2 \right]$	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
$\operatorname{cin}_{-}\operatorname{reg}$	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
$\operatorname{opcode}\left[0-2 ight]$	1	1	100.00
$\operatorname{opcode\_reg}[0-2]$	1	1	100.00
$\operatorname{out}\left[0-5 ight]$	1	1	100.00
$\operatorname{red}\operatorname{\mathtt{op}} olimits_A$	1	1	100.00
$red_op_A_reg$	1	1	100.00
$_{\mathrm{red\_op\_B}}$	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

Toggle Coverage = 100.00% (118 of 118 bins)

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

#### 1.9 9.Waveform



Figure 1: simulation waveform

```
# 10h Teact modutes:
       ALSU tb
# End time: 17:54:11 on Apr 16,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 6
# vsim -voptargs="+acc" work.ALSU_tb -coverage
# Start time: 17:54:11 on Apr 16,2025
# ** Note: (vsim-8009) Loading existing optimized design _opt
# 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.
                    : ALSU tb.sv(315)
# ** Note: $finish
    Time: 10175 ns Iteration: 0 Instance: /ALSU tb
# 1
# Break in Module ALSU tb at ALSU tb.sv line 315
VSIM 2>
```

Figure 2: Transcript : all test cases passed

### 2 Q2: SVA

#### 2.1 1. Code

1 //Q1

```
property p_1;
   @(posedge clk) a |-> ##2 b;
   endproperty
   assert property (p_1);
   property p_2;
   @(posedge clk) (a & b) |-> ##[1:3] c;
11 endproperty
assert property (p_2);
14
15
sequence s11b;
    @(posedge clk) b ##2 (!b);
18 endsequence
19
20 //Q4-a
property p_3;
     @(poseedge clk) $onehot(Y);
23 endproperty
24
assert property (p_3);
27 //Q4-b
property p_4;
    @(posedge clk) (D == 0) |-> ##1 (!valid);
30 endproperty
assert property (p_4);
```

#### 3 Q3: Counter

#### 3.1 1. Testbench code

```
import counter_pkg::*;
  module counter_tb(counter_intf.TEST count_if_tb);
      //-----
      // Create an instance of the random config class
      // Direct assertion: asynchronous reset assertion
      //----
12
      always_comb begin
13
         if(!count_if_tb.rst_n) begin
            a_reset: assert final(count_if_tb.count_out == 0);
15
16
      end
17
18
      //-----
19
      // SVA replaced golden model
20
     //-----
21
     // When the load control signal is active, then the dout has the value of the din
22
23
      property p_load;
         @(posedge count_if_tb.clk) disable iff (!count_if_tb.rst_n) (!count_if_tb.load_n) |=> (count_if_tb.count_out ===
24
            $past(count_if_tb.data_load));
25
      endproperty
      p1:assert property (p_load) else
26
          \texttt{\$error}("SVA_{\sqcup}p\_load_{\sqcup}failed:_{\sqcup}when_{\sqcup}load_{\sqcup}asserted,_{\sqcup}count\_out_{\sqcup}!=_{\sqcup}data\_load");
```

```
// When the load control signal is not active, and the enable is off then the dout does not change
           @(posedge count_if_tb.clk) disable iff (!count_if_tb.rst_n) (count_if_tb.load_n&&!count_if_tb.ce) |=> (count_if_tb.count_out ===
              $past(count_if_tb.count_out));
       endproperty
34
       p2:assert property (p_load_en) else
           $error("SVAup_load_enufailed:uwhenuloadunotuasserteduanduenableuoff,ucount_outu!=upast(count_out)");
       cover property (p_load_en);
38
39
       // When the load control signal is not active and the enable is active, and the up_down is high then the dout is incremented.
       property p_inc;
           @(posedge count_if_tb.clk) disable iff (!count_if_tb.rst_n) (count_if_tb.load_n && count_if_tb.ce && count_if_tb.up_down)
42
                                  |=> (count_if_tb.count_out === $past(count_if_tb.count_out) + 1'b1);
44
       p3:assert property (p_inc) else
45
           $error("SVA_p_inc_failed:_increment_behavior");
       cover property (p_inc);
       // When the load control signal is not active and the enable is active, and the up_down is low then the dout is decremented.
50
51
       property p_dec;
           @(posedge count_if_tb.clk) disable iff (!count_if_tb.rst_n) (count_if_tb.load_n && count_if_tb.ce && !count_if_tb.up_down)
                                  |=> (count_if_tb.count_out === $past(count_if_tb.count_out) - 1'b1);
53
54
       endproperty
55
       p4:assert property (p_dec) else
           $error("SVA_p_dec_failed:_decrement_behavior");
56
       cover property (p_dec);
       // max_count output is high when the counter output is maximum.
61
       always_comb begin
           if(count_if_tb.rst_n && count_if_tb.count_out === {count_if_tb.WIDTH{1'b1}}) begin
62
              p_max_flag: assert final(count_if_tb.max_count === 1'b1) else
                                     $error("SVA_p_max_flag_failed:_max_count_mismatch");
64
65
           end
       //zero output is high when the counter output is zero.
68
       always_comb begin
           if(count_if_tb.rst_n && count_if_tb.count_out === {count_if_tb.WIDTH{1'b0}}) begin
70
              p_zero_flag: assert final(count_if_tb.zero === 1'b1) else
71
                                     $error("SVAup_zero_flagufailed:uzerouflagumismatch");
           end
73
74
     //-----
     // Main verification process: Drives the testbench flow
     //-----
78
     initial begin
79
       // Create config object for randomization
81
       cfg = new();
       // 2) Run random tests
83
       for (int i = 0; i < 500; i++) begin
         if (!cfg.randomize()) begin
85
           $error("Randomization_failed!");
           $finish;
         end
         @(negedge count_if_tb.clk); // Wait for clock edge
         // Drive signals from random config
         count_if_tb.rst_n = cfg.rst_n;
         count_if_tb.load_n = cfg.load_n;
         count_if_tb.up_down = cfg.up_down;
         count_if_tb.ce
96
                           = cfg.ce;
         count_if_tb.data_load= cfg.data_load;
         cfg.count_out = count_if_tb.count_out;
99
100
         cfg.sample();
       end
102
103
104
       $display("All done. End of simulation.");
       $finish; // End simulation
105
106
108 endmodule
   3.2 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;
11
12
           13
           // DUT signals we want to randomize
           //-----
```

cover property (p\_load);

41

52

57 58

63

67

77

80

91 92

93

94

97

101

bit

clk; // clock signal

```
rand bit
                                        // Active-low reset
17
                         rst_n;
                                        // Active-low load
          rand bit
                         load_n;
                                       // 1 => increment, 0 => decrement
          rand bit
                         up_down;
                                        // clock enable
                         ce;
          rand logic [WIDTH-1:0] data_load;
21
          logic [WIDTH-1:0] count_out;
22
23
          //========
24
          // Coverage group
          //========
27
          covergroup cg ;
            cp1: coverpoint rst_n;
28
29
            cp2: coverpoint load_n;
            cp3: coverpoint up_down;
            cp4: coverpoint ce;
31
            cp5: coverpoint data_load;
            cp6: coverpoint count_out;
          \verb"endgroup"
34
          //=========
          // Constructor
37
          //=========
          function new();
39
           cg = new();
40
          \verb"endfunction"
42
          function void sample();
            cg.sample();
          endfunction
45
46
          //----
          // 3) Constraints to meet the 70\%/30\% guidelines
          // use distribution for probability
          //-----
          constraint reset_deactivated_most {
51
            // Reset low 30% of time, high 70%
52
            rst_n dist { 1 := 70, 0 := 30 };
53
54
          constraint load_active_70 {
            // load_n=0 is "active" \Rightarrow 70% of time
57
            load_n dist { 0 := 70, 1 := 30 };
59
          constraint enable_active_70 {
           // ce=1 => 70% of time
62
            ce dist { 1 := 70, 0 := 30 };
63
65
          constraint up_down_dist {
66
            // Distribution constraint: 50\% chance of 0, 50\% chance of 1
            up_down dist { 0 := 50, 1 := 50 };
68
69
          constraint up_down_data_load_c {
71
               if (up_down) {
              // up_down=1 => pick data_load mostly in lower half
              data_load dist {
74
                [0 : (1 << (WIDTH-1))-1] := 80,
75
                [(1 << (WIDTH-1)) : (1 << WIDTH)-1] := 20
              };
                } else {
              // up_down=0 => pick data_load mostly in upper half
              data_load dist {
80
81
                [0 : (1 << (WIDTH-1))-1] := 20,
               [(1 << (WIDTH-1)) : (1 << WIDTH)-1] := 80
              };
83
               }
          }
85
      endclass
89 endpackage
  3.3 3. Design code
2 // Author: Kareem Waseem
  // Course: Digital Verification using SV & UVM
4 //
5 // Description: Counter Design
6 //
```

```
8 module counter (counter_intf.DUT count_if_dut);
  always @(posedge count_if_dut.clk or negedge count_if_dut.rst_n) begin
10
      if (!count_if_dut.rst_n)
11
          count_if_dut.count_out <= 0;</pre>
12
      else if (!count_if_dut.load_n)
13
          count_if_dut.count_out <= count_if_dut.data_load;</pre>
      else if (count_if_dut.ce) begin
15
          if (count_if_dut.up_down)
16
17
              count_if_dut.count_out <= count_if_dut.count_out + 1;</pre>
18
              count_if_dut.count_out <= count_if_dut.count_out - 1;</pre>
19
20
      end
21 end
22
  assign count_if_dut.max_count = (count_if_dut.count_out == {count_if_dut.WIDTH{1'b1}})? 1:0;
24 assign count_if_dut.zero = (count_if_dut.count_out == 0)? 1:0;
```

```
26 endmodule
```

#### 3.4 4. Interface code

```
interface counter_intf (
      input bit clk
5 //-----
6 // Local parameters: Override the default WIDTH if needed
8 localparam WIDTH = 4;
12 // Declare all signals used to interface with DUT or top
14 bit rst_n;
bit load_n;
bit up_down;
17 bit ce;
18 bit [WIDTH-1:0] data_load;
19 bit [WIDTH-1:0] count_out;
20 bit max_count;
21 bit zero;
modport DUT ( input clk,rst_n,load_n,up_down,ce,data_load,
               output count_out,max_count,zero);
modport TEST ( output clk,rst_n,load_n,up_down,ce,data_load,
                input count_out, max_count, zero);
29 endinterface
```

#### 3.5 5. Bug Fixes

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

#### 3.6 6. Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
COUNTER_1	When reset (rst_n) is asserted, the out-	Directed at the start of	Coverage point to track	A checker in the testbench
	put count_out should be zero.	the simulation, followed	how many times reset	ensures count_out is 0
		by randomization with a	is asserted and confirm	when $rst_n == 0$ .
		constraint to keep reset	count_out == 0.	
		active for 30% of the time.		
$COUNTER_2$	When load (load_n) is asserted,	Randomization of load_n	Coverage point to track	A checker in the test-
	count_out should take the value of	with a 70% probability	how many times load_n is	bench verifies count_out
	data_load.	of being active (low) and	asserted and the range of	== data_load when
		randomized data_load	data_load values.	load_n == 0.
		values.		
COUNTER_3	When ce is enabled, the counter	Randomization with 70%	Coverage point to cap-	A checker compares
	should increment or decrement based	chance for ce being high,	ture the toggling of	count_out with the
	on up_down.	and a 50-50 distribution	up_down and transitions	expected increment
		for up_down.	of count_out.	or decrement, verified
				against the golden model.
$COUNTER_4$	max_count should be asserted when	Random tests allowing	Coverage point to check	A checker validates
	count_out reaches its maximum value.	the counter to reach its	how often max_count is	max_count == 1 when
		maximum possible value	triggered.	count_out == max
		$(\{WIDTH\{1'b1\}\}).$		value.
$COUNTER_{-5}$	zero should be asserted when	Directed reset tests and	Coverage point for how of-	A checker verifies zero ==
	count_out is zero.	decrement tests pushing	ten zero is asserted.	1 only when count_out
		count_out to zero.		== 0.

Table 2: Verification Plan for Counter Design

### 3.7 7. Do File

```
vlib work
vlog counter_pkg.sv counter.sv counter_intf.sv counter_tb.sv top_module.sv +cover -covercells
vsim -voptargs=+acc work.top -cover
do wave.do
coverage save top.ucdb -onexit
run -all
do coverage.do

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

#### 3.8 8. code Coverage Report

Coverage Report by DU with details

```
— Design Unit: work.counter
```

Branch Coverage:

```
100.00\%
    Branches
                                      10
                                                 10
                                                             0
                                 =Branch Details=
Branch Coverage for Design Unit work.counter
    Line
                  Item
                                              Count
                                                         Source
  File counter.sv
                                      \operatorname{-IF} Branch-
    11
                                                207
                                                        Count coming in to IF
                     1
                                                 81
                                                             if (!count_if_dut.rst_n)
    11
                                                             else if (!count_if_dut.load_n)
                     1
                                                 98
    13
                                                             else if (count_if_dut.ce) begin
                                                 22
    15
                                                  6
                                                         All False Count
Branch totals: 4 hits of 4 branches = 100.00%
                                      -IF Branch
                                                 22
    16
                                                         Count coming in to IF
                     1
                                                                 if (count_if_dut.up_down)
    16
                                                 13
                                                  9
    18
                     1
Branch totals: 2 hits of 2 branches = 100.00\%
                                      \operatorname{-IF} Branch-
                                                        Count coming in to IF
    23
                                                145
                     1
                                                         assign count_if_dut.max_count = (count_if_dut.count_out == {count_if_dut.WIDTH{1'b1}}
    23
                                                10
                                                135
                                                         assign count_if_dut.max_count = (count_if_dut.count_out == {count_if_dut.WIDTH{1'b1}}
    23
                     2
Branch totals: 2 hits of 2 branches = 100.00\%
                                      \operatorname{-IF} Branch-
    24
                                                145
                                                         Count coming in to IF
                                                         assign count_if_dut.zero = (count_if_dut.count_out == 0)? 1:0;
    24
                     1
                                                34
                     ^{2}
                                                111
                                                         assign count_if_dut.zero = (count_if_dut.count_out == 0)? 1:0;
Branch totals: 2 hits of 2 branches = 100.00%
Condition Coverage:
    Enabled Coverage
                                    _{\mathrm{Bins}}
                                            Covered
                                                               Coverage
                                                        Misses
    Conditions
                                                                 100.00\%
                                                            0
                                 =Condition Details=
Condition Coverage for Design Unit work.counter —
  File counter.sv
                -{
m Focused} Condition View-
           23 Item
                     1 \quad (count_if_dut.count_out = \{count_if_dut.WIDTH\{1\}\})
Condition totals: 1 of 1 input term covered = 100.00%
                                             Input Term
                                                           Covered Reason for no coverage
  (count_if_dut.count_out = \{count_if_dut.WIDTH\{1\}\})
                                                                  Y
                  Hits FEC Target
                                                                                   Non-masking condition(s)
     Rows:
 Row
                        (count_if_dut.count_out = \{count_if_dut.WIDTH\{1\}\})_0
 Row
                        (count_if_dut.count_out = \{count_if_dut.WIDTH\{1\}\})_1
                 -Focused Condition View-
           24 Item
                     1 (count_if_dut.count_out == 0)
Condition totals: 1 of 1 input term covered = 100.00%
                      Input Term
                                    Covered Reason for no coverage
  (count_if_dut.count_out = 0)
                                                           Non-masking condition(s)
     Rows:
                  Hits FEC Target
        1:
                        (count_if_dut.count_out == 0)_0
  Row
                        (count_if_dut.count_out == 0)_1
Statement Coverage:
    Enabled Coverage
                                    Bins
                                               Hits
                                                       Misses Coverage
                                                                 100.00\%
    Statements
                                                             0
                                  =Statement Details:
Statement Coverage for Design Unit work.counter —
                  Item
    Line
                                              Count
                                                        Source
  File counter.sv
                                                        module counter_intf.DUT count_if_dut);
    8
    9
                                                207
                                                         always @(posedge count_if_dut.clk or negedge count_if_dut.rst_n) begin
    10
                     1
                                                             if (!count_if_dut.rst_n)
    11
                     1
                                                 81
                                                                 count_if_dut.count_out <= 0;</pre>
    12
```

Enabled Coverage

Bins

Misses

Coverage

```
else if (!count_if_dut.load_n)
 13
                                                                                           1
                                                                                                                                                                                                                                                   98
                                                                                                                                                                                                                                                                                                                                          count_if_dut.count_out <= count_if_dut.data_load;</pre>
 14
                                                                                                                                                                                                                                                                                                                   else if (count_if_dut.ce) begin
 15
16
                                                                                                                                                                                                                                                                                                                                          if (count_if_dut.up_down)
                                                                                            1
                                                                                                                                                                                                                                                   13
                                                                                                                                                                                                                                                                                                                                                                 count_if_dut.count_out <= count_if_dut.count_out + 1;</pre>
17
18
                                                                                                                                                                                                                                                                                                                                          else
                                                                                           1
                                                                                                                                                                                                                                                        9
                                                                                                                                                                                                                                                                                                                                                                 count_if_dut.count_out <= count_if_dut.count_out - 1;</pre>
19
20
                                                                                                                                                                                                                                                                                                            end
                                                                                                                                                                                                                                                                                            \quad \text{end} \quad
21
22
                                                                                            1
                                                                                                                                                                                                                                             145
                                                                                                                                                                                                                                                                                            assign \ count\_if\_dut.max\_count = (count\_if\_dut.count\_out == \{count\_if\_dut.WIDTH\{1'b1, assign, count\_if\_dut.width]\} + (count\_if\_dut.max\_count, assign, count\_if\_dut.width) + (count\_if\_dut.max\_count, assign, count\_if\_dut.max\_count, assign, as
23
                                                                                           1
                                                                                                                                                                                                                                             145
                                                                                                                                                                                                                                                                                            assign count_if_dut.zero = (count_if_dut.count_out == 0)? 1:0;
24
```

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

#### 3.9 9. functional coverage report

Coverage Report by instance with details

Covergroup Bins	40	40   0   100.	00%		
overgroup		Metric	Goal	Bins	Status
	<u> </u>	100.00%	100		Covered
covered/total bins:		40	40	_	
missing/total bins:		0	40	_	
% Hit:		100.00%	100	_	
Coverpoint cp1		100.00%	100	_	$\operatorname{Covered}$
covered/total bins:		2	2	_	
missing/total bins:		0	2	_	
% Hit:		100.00%	100	_	
bin auto $[0]$		46	1	_	$\widehat{\text{Covered}}$
bin auto[1]		126	1	_	$\widehat{\text{Covered}}$
Coverpoint cp2		100.00%	100	_	$\operatorname{Covered}$
covered/total bins:		$\frac{2}{2}$	$\frac{2}{2}$	_	
missing/total bins:		0	2	_	
% Hit:		100.00%	100	_	
bin auto[0]		129	1	_	Covered
bin auto[1]		43	1	_	Covered
Coverpoint cp3		100.00%	100	_	Covered
covered/total bins:		$\frac{2}{2}$	$\frac{2}{2}$	_	
missing/total bins:		100.0007	2	_	
% Hit:		100.00%	100	_	C 1
bin auto[0]		80	1	_	Covered
bin auto[1]		92	1	_	Covered
Coverpoint cp4		100.00%	100	_	Covered
covered/total bins:		2	2	_	
missing/total bins:		100.0007	2	_	
% Hit:		100.00%	100	_	Covered
bin auto[0]		51 $121$	1	_	Covered
bin auto[1]		100.00%	$1\\100$	_	Covered
Coverpoint cp5		160.00%	160	_	Covered
covered/total bins:		0	16	_	
missing/total bins: % Hit:		100.00%	100	_	
bin auto $[0]$		7	100	_	Covered
		10		_	Covered
$egin{array}{ll}  ext{bin auto} \left[ 1  ight] \  ext{bin auto} \left[ 2  ight] \end{array}$		10	1	_	Covered
bin auto[2]		12	$\frac{1}{1}$	_	Covered
bin auto $[4]$		10	1	_	Covered
bin auto $\begin{bmatrix} 4 \end{bmatrix}$		15	1	_	Covered
bin auto[6]		9	1	_	Covered
bin auto[0]		13	1	_	Covered
bin auto[8]		12	1	_	Covered
bin auto[9]		10	1	_	Covered
bin auto[10]		12	1	_	Covered
bin auto[11]		13	1	_	Covered
bin auto [12]		9	1	_	Covered
bin auto[13]		13	1	_	Covered
bin auto[14]		11	1	_	Covered
bin auto[15]		6	1	_	Covered
Coverpoint cp6		100.00%	100	_	Covered
covered/total bins:		16	16	_	00,0104
missing/total bins:		0	16	_	
% Hit:		100.00%	100	_	
bin auto[0]		57	1	_	Covered
bin auto[1]		10	1	_	Covered
bin auto[2]		6	1	_	Covered
bin auto [3]		6	1	_	Covered
		$\stackrel{\circ}{3}$	1	_	Covered
bin auto [4]		ð	1	_	Covered

bin auto [6]	5	1	_	Covered
bin auto [7]	7	1	_	Covered
bin auto [8]	9	1	_	Covered
bin auto [9]	10	1	_	Covered
bin auto [10]	9	1	_	Covered
bin auto[11]	8	1	_	Covered
bin auto [12]	7	1	_	Covered
bin auto [13]	7	1	_	Covered
bin auto [14]	7	1	_	Covered
bin auto [15]	10	1	_	Covered

### COVERGROUP COVERAGE:

overgroup	Metric	Goal	Bins	Status
	100.00%	100		Covered
covered/total bins:	40	40	_	
missing / total bins:	0	40	_	
% 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]	46	1	_	Covered
bin auto[1]	126	1	_	Covered
Coverpoint cp2	100.00%	100	_	Covered
<pre>covered/total bins: missing/total bins:</pre>	$\frac{2}{0}$	$\frac{2}{2}$	_	
% Hit:	100.00%	100	_	
bin auto [0]	100.0076 $129$	1		Covered
bin auto[1]	43	1	_	Covered
Coverpoint cp3	100.00%	100	_	Covered
covered/total bins:	2	$\overline{2}$	_	
missing/total bins:	0	$\stackrel{-}{2}$	_	
% Hit:	100.00%	100	_	
bin auto[0]	80	1	_	Covered
bin auto[1]	92	1	_	Covered
Coverpoint cp4	100.00%	100	_	Covered
covered/total bins:	2	2	_	
missing/total bins:	0	2	_	
% Hit:	100.00%	100	_	
$ \text{bin auto} \left[  0  \right]$	51	1	_	$\operatorname{Covered}$
bin auto [1]	121	1	_	$\operatorname{Covered}$
Coverpoint cp5	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	16	16	_	
missing/total bins:	0	16	_	
% Hit:	100.00%	100	_	
bin auto [0]	7	1	_	Covered
bin auto[1]	10	1	_	Covered
bin auto [2]	10	1	_	Covered
bin auto [3]	$\frac{12}{10}$	1	_	Covered
bin auto [4]	10	1	_	Covered
bin auto[5]	$\begin{array}{c} 15 \\ 9 \end{array}$	1	_	Covered Covered
$\begin{array}{c}  ext{bin auto} [6] \\  ext{bin auto} [7] \end{array}$	$\frac{9}{13}$	1	_	Covered
bin auto[8]	$\frac{13}{12}$	1		Covered
bin auto[9]	10	1	_	Covered
bin auto [10]	$\frac{10}{12}$	1	_	Covered
bin auto[11]	13	1	_	Covered
bin auto [12]	9	1	_	Covered
bin auto [13]	13	1	_	Covered
bin auto [14]	11	1	_	Covered
bin auto [15]	6	1	_	$\operatorname{Covered}$
Coverpoint cp6	100.00%	100	_	$\operatorname{Covered}$
covered/total bins:	16	16	_	
missing/total bins:	0	16	_	
% Hit:	100.00%	100	_	
$ \text{bin auto} \left[  0  \right]$	57	1	_	$\operatorname{Covered}$
bin auto[1]	10	1	_	$\operatorname{Covered}$
$ \text{bin } \text{ auto} \left[ 2 \right]$	6	1	_	$\operatorname{Covered}$
$\operatorname{bin} \ \operatorname{auto} \left[ 3 \right]$	6	1	_	$\operatorname{Covered}$
$\mathrm{bin} \ \mathrm{auto} \left[ 4 \right]$	3	1	_	$\operatorname{Covered}$
bin auto [5]	11	1	_	Covered
bin auto [6]	5	1	_	Covered
bin auto [7]	7	1	_	Covered
bin auto [8]	9	1	_	Covered
bin auto [9]	10	1	_	Covered
bin auto[10]	9	1	_	Covered
bin auto [11]	8	1	_	Covered
bin auto [12]	() <del> </del>	1	_	Covered
bin auto [13]	· (	1	_	Covered
bin auto [14]	7	1	_	Covered
$bin \ \ auto \ [15]$	10	1	_	$\operatorname{Covered}$

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

Total Coverage By Instance (filtered view): 100.00%

▼ Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
<pre>/top/test/coverp_dec</pre>	SVA	1	Off	30	1	Unli	1	100%		<b>I</b> √	0	0	0 ps	0
/top/test/coverp_inc	SVA	1	Off	26	1	Unli	1	100%		i 🗸	0	0	0 ps	0
/top/test/coverp_load_en	SVA	1	Off	16	1	Unli	1	100%		l 🗸	0	0	0 ps	0
/top/test/coverp_load	SVA	1	Off	191	1	Unli	1	100%		<b>V</b>	0	0	0 ps	0

Figure 3: cover property

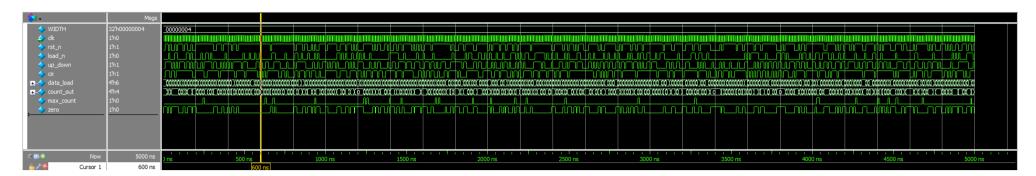


Figure 4: simulation waveform

```
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0' because of 'restart'.
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.counter_intf(fast__1)
# Loading work.counter_intf(fast__1)
# Loading work.counter_pkg(fast)
# Loading work.counter_pkg(fast)
# Loading work.counter_pkg(fast)
# Loading work.counter_bg(fast)
#
```

Figure 5: Transcript : all test cases passed

# 4 Q4: configuration register

#### 4.1 1. Testbench code

module config\_reg\_tb;

```
// Enumerated type for refister addresses
       typedef enum logic [2:0] {
          ADCO_REG = 3'dO,
          ADC1_REG = 3'd1,
          TEMP_SENSORO_REG = 3'd2,
          TEMP_SENSOR1_REG = 3'd3,
          ANALOG_TEST = 3'd4,
          DIGITAL\_TEST = 3'd5,
12
          AMP_GAIN = 3'd6,
13
          DIGITAL\_CONFIG = 3'd7
      } reg_addr_e;
15
17
       // Golden model for reset values of register
18
       // using assosiative array with type string key
19
      logic [15:0] reset_assoc [string];
21
22
       //-----
       // instantiate configuration register
24
       //-----
      logic clk;
      logic reset;
      logic write;
      logic [15:0] data_in;
      reg_addr_e address;
30
      logic [15:0] data_out;
31
32
      config_reg dut (
33
         .clk(clk),
          .reset(reset),
          .write(write),
36
37
           .data_in(data_in),
          .address(address),
38
           .data_out(data_out)
39
      );
40
41
42
      // Clock generator
43
      //----
44
       initial begin
45
46
          clk = 1'b0;
          forever #5 clk = ~clk;
47
48
49
50
51
       // Task to check dut during reseting
52
       task check_rest();
53
54
      begin
55
          reg_addr_e addr;
```

```
reset = 1'b1;
   #10;
   reset = 1'b0;
    $display("check reset task");
    for (int i = 0; i < addr.num(); i++) begin</pre>
       if(i==0) begin
            addr = addr.first();
            address = addr;
        end else if (i==(addr.num()-1))begin
            addr = addr.last();
            address = addr;
        end else begin
            addr = addr.next();
            address = addr;
        end
        #10;
        //compare registers content after resting with associative array that hold reset values
        if (data_out !== reset_assoc[addr.name()]) begin
            display("Mismatch_at_%s:_expected_%h,_got_%h", addr.name(), reset_assoc[addr.name()], data_out);
        end
end
endtask
// Task to check register values
//-----
task check_registers();
   reg_addr_e addr;
   logic [15:0] expected_output;
   // corner cases
    $display("check_register_values_all_zero_write");
    write = 1'b1;
    data_in = 16'b0;
    for (int i = 0; i < addr.num(); i++) begin</pre>
        if(i==0) begin
            addr = addr.first():
            address = addr;
        end else if (i==(addr.num()-1))begin
           addr = addr.last();
            address = addr;
        end else begin
           addr = addr.next();
            address = addr;
        end
        #10;
        if (data_out !== 16'b0) begin
            display("Mismatch_at_%s:_expected_%h,_got_%h", addr.name(), 16'b0, data_out);
        end
    display("check_register_values_all_ones_write");
    write = 1'b1;
    data_in = 16'hFFFF;
    for (int i = 0; i < addr.num(); i++) begin</pre>
       if(i==0) begin
            addr = addr.first();
            address = addr;
        end else if (i==(addr.num()-1))begin
            addr = addr.last();
            address = addr;
        end else begin
            addr = addr.next();
            address = addr;
        end
        #10;
        if (data_out !== 16'hFFFF) begin
            $display("Mismatchuatu%s:uexpectedu%h,ugotu%h", addr.name(), 16'hFFFF, data_out);
    end
    $display("check_register_values_random,write");
    write = 1'b1;
    expected_output = $urandom_range(0,16'hFFFF);
    data_in = expected_output;
    for (int i = 0; i < addr.num(); i++) begin</pre>
        if(i==0) begin
            addr = addr.first();
            address = addr;
        end else if (i==(addr.num()-1))begin
            addr = addr.last();
            address = addr;
        end else begin
            addr = addr.next();
            address = addr;
        end
        #10:
        if (data_out !== expected_output) begin
            $display("Mismatchuatu%s:uexpectedu%h,ugotu%h", addr.name(), expected_output, data_out);
        end
    end
```

63

64

66

67 68

69

70

71

72

73 74

76

78 79

81

82 83 84

85

87

90 91

92

93

95

96

99

101

102

103

104

105 106

107

108

109

110 111 112

113

114

115 116

117

118

119 120

121

122

123

124

125

126

127

128

130 131 132

133

134

136

137 138

139

140 141

142

143

144

145 146

147

148

149

150

152

153

```
157
                                                                 endtask
  158
                                                                 //----
  159
  160
                                                               // simulation steps
  161
                                                                 initial begin
  162
                                                                                             //initialize inputs
  163
                                                                                              reset = 1'b0; //un activated reset
  164
                                                                                                write = 1'b0;
                                                                                                data_in = 15'b0;
  166
                                                                                                address = ADCO_REG;
  167
  168
  169
                                                                                                // set reset values in associative array to hold it to the end of simulation
                                                                                               reset_assoc["ADCO_REG"]=16'hFFFF;
  170
                                                                                                reset_assoc["ADC1_REG"]=16',h0;
  171
                                                                                               reset_assoc["TEMP_SENSORO_REG"]=16'h0;
  172
                                                                                                reset_assoc["TEMP_SENSOR1_REG"]=16'h0;
  173
                                                                                                reset_assoc["ANALOG_TEST"]=16'hABCD;
  174
                                                                                                reset_assoc["DIGITAL_TEST"]=16',h0;
                                                                                                reset_assoc["AMP_GAIN"]=16'h0;
  176
                                                                                                reset_assoc["DIGITAL_CONFIG"]=16'h1;
  178
                                                                                                 // reset dut
  179
                                                                                                @(posedge clk);
                                                                                                check_rest();
  181
  182
                                                                                                @(posedge clk);
                                                                                                check_registers();
  184
  185
                                                                                                $finish;
  186
                                                               end
187
 189 endmodule
                               4.2 2. Design code
                                               module config_reg(
                                                                                               input logic clk,
                                                                                                input logic reset,
                                                                                                input logic write,
                                                                                                input logic [15:0] data_in,
                                                                                                input logic [2:0] address,
                                                                                                output logic [15:0] data_out
         s );
                               'protected
      11
                                                                \texttt{MTI!\#DU; Wt7W} \texttt{E} \texttt{vp\$.h, RJ@F"cok2*:fU|]?KQ7"kDn2YBkJW1x7+D][a92\T\Xo+#4'mT1<\texttt{VEU}] } 
      12
                                                               13
                                                               \label{local-equation} $$ 1^4 = V^2 + V^
      14
                                                               {cQ#=]/_xJpe_kwG?wE4qLAR5TU$3sI(^E#*,iTK7'RVBKXe\YOiZwxmlmAjGo2]1_1uC2}##1#I
                                                               P\#a7n^{e11LPZ*1}[3rTr=iEkb@T>J-pmux, $Z.\vQ2[-1Jajw\sjirkh-'$[Ko7r*\?+$WWAB_-', Function of the context of th
                                                               ?e\_<0`aB.a=Yi7aXRznB;\\ \\ \\ 7ou^{-}=!OV-p2^mDDu'sRiAn!m[J{Ho}+-e1=Y7VDVT[G?v=$nL[:Kaw!]] + (Authority for the context of the 
      17
                                                               EZTH \} x Uz 5 KaliVG ] [KZ2 evwm~|tZlk3xeGs2o[GG]p[BAw!w>eQaTz*R1\#TxX{n>^}! EC$ekE\#Z7 ] | EC$ekE#Z7 
                                                                 , B5sWuxN^{h}uxN^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux^{h}ux
                                                               3+H*vpW@U\{x\{\$R@W^\$>,]E\$v\#BR\{a+,uWuA\}WxQ\}X'JX[05xkbl\#B\#/Dxo[*wV,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,FqZXmAy=I-\\^nK^*U,F
                                                               \verb|a<j^D{**'Q^XzeaXL$Bsow<3*51XBz*W*Z$CXMPK^zCvpDU5DT1aTJ=_Das=>o-TCs?aDE1}U5irC|
      22
                                                                 <A,~vQi}QOuo;!VY@]C-+ROh'?\Ku'pxvE!2(.Ss3JD'Q'eD?Esjp^[6]Z+u3n1]1urmi}@$eiDk
      23
      24
                                                               12 + 731 - ExuR, n < *I!oW^; 1BE; *DBxTpVWXBhq][<} = KoTyU > 2jsY R!U[sD+0 > _ {1U^Ev!27T Y Extended on the content of the c
                                                               s \\ 1q * \$5 \\ T6 \\ kv ! ! \\ hLEI \\ wsx 'CXe \\ \#GeQ * Y '[OvRkpr \\ aloxGHpoW 'lj \\ \} \\ xRpjG ! \\ oTem?Q \\ , mIp '7^11G] \\ property 
      25
                                                               ^ep=(IH<oPD}QBJr<1p-'{sm5z@HBG,#rCiEQY}_1r[>;rr]u,wCKpGu}*w>1vvBr#j>s3Bz^12x
                                                                 \label{eq:condition} $$ \sim \mathbb{I}^{n}_{i,k}^{n} = \mathbb{I}^{i_{x,k}} = \mathbb{I}^{n}_{i,k}^{n} = \mathbb{I}^{
                                                                3v@DZH1kp'@H0^EoI"'Kj3Wo'3*={jE$m*bo_}xBVs]E[psIrjABY+s]usrsi!\L\#_\#J/Vm\HVCerline{ } The property of the pr
      29
                              'endprotected
     34 endmodule⊔//⊔test
                                                                        3. Bug report
                                       Bug 1
                                                                 a) Design Input for Bug to Appear:
                                                                                                   \label{eq:conditional} Read \ ABCD \ from \ \ register \ \ ANALOG\_TEST \ \ after \ \ reset \, .
                                                               b) Expected Behavior:
                                                                                                     Register ANALOG_TEST should be readable as ABCD.
                                                                  c) Observed Behavior:
                                                                                                    Register ANALOG_TEST reads as ABCC instead of ABCD.
                              \mathrm{Bug}\ 2
                                                                                      Design Input for Bug to Appear:
                                                                 a)
                                                                                                     Write 0000 to register ADCO_REG.
                                                               b) Expected Behavior:
                                                                                                     Register ADCO_REG should be writable as 0000.
                                                                  c) Observed Behavior:
                                                                                                     Register ADCO_REG is written as FFFF instead of 0000.
                              Bug 3
                                                                 a) Design Input for Bug to Appear:
```

Write FFFF to register TEMP\_SENSORO\_REG.

b) Expected Behavior:

Register TEMP\_SENSORO\_REG should be writable as FFFF.

c) Observed Behavior:

 $Register\ TEMP\_SENSOR0\_REG\ is\ written\ as\ FFFE\ instead\ of\ FFFF.$ 

Bug 4

a) Design Input for Bug to Appear:

Write FFFF to register DIGITAL\_TEST.

b) Expected Behavior:

Register DIGITAL\_TEST should be writable as FFFF.

c) Observed Behavior:

Register DIGITAL\_TEST is written as 0000 instead of FFFF.

Bug 5

a) Design Input for Bug to Appear:

Write FFFF to register DIGITAL\_CONFIG.

b) Expected Behavior:

Register DIGITAL\_CONFIG should be writable as FFFF.

c) Observed Behavior:

Register DIGITAL\_CONFIG is written as 7FFF instead of FFFF.

Bug 6

a) Design Input for Bug to Appear:

Write 37E7 to register ADCO\_REG.

b) Expected Behavior:

Register ADCO\_REG should be writable as 37E7.

c) Observed Behavior:

Register ADCOREG is written as B7E7 instead of 37E7.

 $\mathrm{Bug}\ 7$ 

a) Design Input for Bug to Appear:

Write 37E7 to register ADC1\_REG.

b) Expected Behavior:

Register ADC1\_REG should be writable as 37E7.

c) Observed Behavior:

Register ADC1\_REG is written as E737 instead of 37E7.

Bug 8

a) Design Input for Bug to Appear:

Write 37E7 to register TEMP\_SENSOR0\_REG.

b) Expected Behavior:

Register TEMP.SENSOR0\_REG should be writable as  $37\mathrm{E}7.$ 

c) Observed Behavior:

Register TEMP\_SENSOR0\_REG is written as 6FCE instead of 37E7.

Bug 9

a) Design Input for Bug to Appear:

Write 37E7 to register DIGITAL\_TEST.

b) Expected Behavior:

Register DIGITAL\_TEST should be writable as 37E7.

c) Observed Behavior:

Register DIGITAL\_TEST is written as FFFF instead of 37E7.

#### 4.4 4. Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
CONFIG_1	Upon reset, all registers should be set	Directed reset at the start	-	A checker in the testbench
	to their predefined reset values.	of the simulation, followed		compares each register's
		by random operations.		output with the expected
				reset value from the asso-
				ciative array.
CONFIG_2	Writing to a register should update its	Randomized write opera-	-	A checker verifies that the
	value to the input data.	tions with varying data in-		register's output matches
		puts across all registers.		the input data after a
				write operation.
CONFIG_3	Registers should maintain their values	Randomized operations	-	A checker ensures that
	when no write operation is performed.	with a 50% chance of		register values remain un-
		write being disabled.		changed when write is not
				active.
CONFIG_4	The module should correctly handle si-	Directed tests where reset	-	A checker confirms that
	multaneous reset and write operations,	and write are asserted si-		reset values take prece-
	prioritizing reset.	multaneously.		dence over write opera-
				tions when both are ac-
				tive.
CONFIG_5	The module should correctly handle	Directed tests with data	-	A checker verifies correct
	edge cases, such as maximum and min-	inputs set to all zeros, all		register behavior for edge
	imum data values.	ones, and random values.		case data inputs.

Table 3: Verification Plan for Configuration Register Design

#### 4.5 5. Waveform

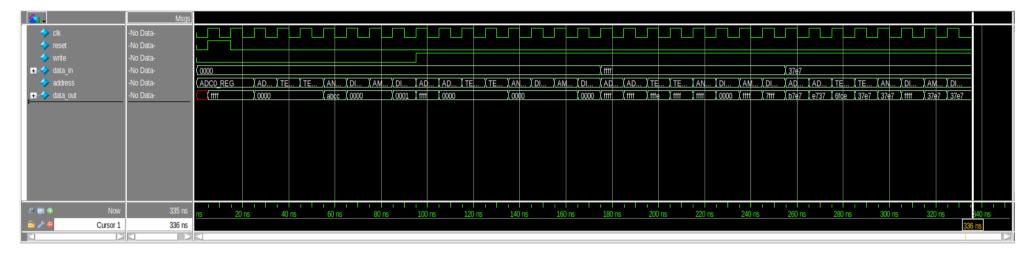


Figure 6: simulation waveform

```
VSIM 5> run -all
# check reset task
# Mismatch at ANALOG TEST: expected abcd, got abcc
# check register values all zero write
# Mismatch at ADCO REG: expected 0000, got ffff
# check register values all ones write
# Mismatch at TEMP SENSORO REG: expected ffff, got fffe
# Mismatch at DIGITAL_TEST: expected ffff, got 0000
# Mismatch at DIGITAL_CONFIG: expected ffff, got 7fff
# check register values random write
# Mismatch at ADCO REG: expected 37e7, got b7e7
# Mismatch at ADC1 REG: expected 37e7, got e737
# Mismatch at TEMP SENSORO REG: expected 37e7, got 6fce
# Mismatch at DIGITAL TEST: expected 37e7, got ffff
                    : /home/tare/Desktop/assignment 4 - verification diploma/Q4/config reg tb.sv(186)
# ** Note: $finish
    Time: 335 ns Iteration: 0 Instance: /config_reg_tb
# Break in Module config reg tb at /home/tare/Desktop/assignment 4 - verification diploma/Q4/config reg tb.sv line 186
VSIM 6>
```

Figure 7: Transcript : all test cases passed