

MARIAM MOHAMED MARZOUK

ASSIGNMENT_6

I. RTL design

```
1 module ALSU(ALSU_IF.DUT_AL_IF);
2 parameter INPUT_PRIORITY = "A";
3 parameter FULL_ADDER = "ON";
4 reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
5 reg [2: 0] opcode_reg, A_reg, B_reg;
6 wire invalid_red_op, invalid_opcode, invalid;
7 assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] & opcode_reg[2]);
8 assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
9 assign invalid = invalid_red_op | invalid_opcode;
10 always @(posedge AL_if.clk or posedge AL_if.reset) begin
11 if(AL_if.reset) begin
12 cin_reg <= 0;
13 red_op_B_reg <= 0;
14 red_op_A_reg <= 0;
15 bypass_B_reg <= 0;
16 bypass_A_reg <= 0;
17 direction_reg <= 0;
18 serial_in_reg <= 0;
19 opcode_reg <= 0;
20 A_reg <= 0;
21 B_reg <= 0;
22 end else begin
23 cin_reg <= AL_if.cin;
24 red_op_B_reg <= AL_if.red_op_B;
25 red_op_A_reg <= AL_if.red_op_A;
26 bypass_B_reg <= AL_if.bypass_B;
27 bypass_A_reg <= AL_if.bypass_A;
28 direction_reg <= AL_if.direction;
29 serial_in_reg <= AL_if.serial_in;
30 opcode_reg <= AL_if.opcode;
31 A_reg <= AL_if.A;
32 B_reg <= AL_if.B;
33 end
34 end
35 always @(posedge AL_if.clk or posedge AL_if.reset) begin
36 if(AL_if.reset) begin
37 AL_if.leds <= 0;
38 end else begin
39 if (invalid)
40 AL_if.leds <= ~AL_if.leds;
41 else
42 AL_if.leds <= 0;
43 end
44 end
45 always @(posedge AL_if.clk or posedge AL_if.reset) begin
46 if(AL_if.reset) begin
47 AL_if.out <= 0;
48 end
49 else begin
50 if (invalid)
51 AL_if.out <= 0;
52 else if (bypass_A_reg == bypass_B_reg)
53 AL_if.out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
54 else if (bypass_A_reg)
55 AL_if.out <= A_reg;
56 else if (bypass_B_reg)
57 AL_if.out <= B_reg;
58 else begin
59 case (opcode_reg)
60 3'b000: begin
61 if (red_op_A_reg && red_op_B_reg)
62 AL_if.out = (INPUT_PRIORITY == "A")? ~A_reg: ~B_reg;
63 else if (red_op_A_reg)
64 AL_if.out <= ~A_reg;
65 else if (red_op_B_reg)
66 AL_if.out <= ~B_reg;
67 else
68 AL_if.out <= A_reg | B_reg;
69 end
70 3'b111: begin
71 if (red_op_A_reg && red_op_B_reg)
72 AL_if.out <= (INPUT_PRIORITY == "A")? ~A_reg: ~B_reg;
73 else if (red_op_A_reg)
74 AL_if.out <= ~A_reg;
75 else if (red_op_B_reg)
76 AL_if.out <= ~B_reg;
77 else
78 AL_if.out <= A_reg ^ B_reg;
79 end
80 3'b101: begin
81 if (FULL_ADDER == "ON") begin
82 AL_if.out <= A_reg + B_reg + cin_reg;
83 end
84 else
85 AL_if.out <= A_reg + B_reg;
86 end
87 3'b100: AL_if.out <= A_reg * B_reg;
88 3'b110: begin
89 if (direction_reg)
90 AL_if.out <= {AL_if.out[4: 0], serial_in_reg};
91 else
92 AL_if.out <= {serial_in_reg, AL_if.out[5: 1]};
93 end
94 3'b011: begin
95 if (direction_reg)
96 AL_if.out <= {AL_if.out[4: 0], AL_if.out[5: 1]};
97 else
98 AL_if.out <= {AL_if.out[0], AL_if.out[5: 1]};
99 end
100 endcase
101 end
102 end
103 end
104 endmodule
```

II. Top module

```
1 import pack_test::*;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4 module TOP ();
5   bit clk;
6   initial begin
7     clk = 0;
8     forever begin
9       #1 clk = ~clk;
10    end
11  end
12  ALSU_if al_if (clk);
13  ALSU DUT (al_if);
14  ALSU_Gold GOLD(al_if);
15  bind ALSU SVA assertions (al_if.DUT);
16  initial begin
17    uvm_config_db#(virtual ALSU_if):: set(null,"uvm_test_top","ALSU_if", al_if);
18    run_test("test");
19  end
20 endmodule
```

III. Interface module

```
1 interface ALSU_if (clk);
2   input bit clk;
3   logic reset, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
4   logic [2: 0] opcode;
5   logic signed [2: 0] A, B;
6   logic [15: 0] leds, leds_G;
7   logic [5: 0] out;
8   logic [5: 0] out_G;
9   modport DUT (input clk, reset, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in, opcode,A, B, output out, leds);
10  modport DUT_GOLD (input clk, reset, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in, opcode,A, B, output out_G, leds_G);
11 endinterface
```

IV. Golden Model



```
1 module ALSU_Gold (ALSU_if.DUT_GOLD AL_if);
2 parameter INPUT_PRIORITY = "A";
3 parameter FULL_ADDER = "ON";
4 reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
5 reg [2: 0] opcode_reg, A_reg, B_reg;
6 always @(posedge AL_if. clk or posedge AL_if. reset) begin
7 if(AL_if. reset) begin
8 cin_reg <= 0;
9 red_op_B_reg <= 0;
10 red_op_A_reg <= 0;
11 bypass_B_reg <= 0;
12 bypass_A_reg <= 0;
13 direction_reg <= 0;
14 serial_in_reg <= 0;
15 opcode_reg <= 0;
16 A_reg <= 0;
17 B_reg <= 0;
18 end else begin
19 cin_reg <= AL_if. cin;
20 red_op_B_reg <= AL_if. red_op_B;
21 red_op_A_reg <= AL_if. red_op_A;
22 bypass_B_reg <= AL_if. bypass_B;
23 bypass_A_reg <= AL_if. bypass_A;
24 direction_reg <= AL_if. direction;
25 serial_in_reg <= AL_if. serial_in;
26 opcode_reg <= AL_if. opcode;
27 A_reg <= AL_if. A;
28 B_reg <= AL_if. B;
29 end
30 end
31 always @(posedge AL_if. clk or posedge AL_if. reset) begin
32 if(AL_if. reset)begin
33 AL_if. out_G <= 6'h00;
34 AL_if. leds_G <= 16'h0000;
35 end
36 else begin
37 case (opcode_reg)
38 3'b000: begin
39 if (red_op_A_reg && red_op_B_reg)
40 AL_if. out_G <= |A_reg;
41 else if (red_op_A_reg)
42 AL_if. out_G <= |A_reg;
43 else if (red_op_B_reg)
44 AL_if. out_G <= |B_reg;
45 else
46 AL_if. out_G <= A_reg|B_reg;
47 if (bypass_A_reg && bypass_B_reg ) begin
48 AL_if. out_G <= A_reg;
49 end
50 else if (bypass_A_reg) begin
51 AL_if. out_G <= A_reg;
52 end
```

```

1 end;
2 else if (bypass_B_req) begin
3   AL_if.out_S <= B_req;
4 end
5 end;
6 S'blk1: begin
7   if (red_op_A_req || red_op_B_req)
8     AL_if.out_S <= "A_req";
9   else if (red_op_A_req)
10    AL_if.out_S <= "A_req";
11  else if (red_op_B_req)
12    AL_if.out_S <= "B_req";
13  else
14    AL_if.out_S <= A_req||B_req;
15  if (bypass_A_req || bypass_B_req) begin
16    AL_if.out_S <= A_req;
17 end
18 else if (bypass_A_req) begin
19  AL_if.out_S <= A_req;
20 end
21 else if (bypass_B_req) begin
22  AL_if.out_S <= B_req;
23 end
24 end;
25 S'blk2: begin
26  if (red_op_A_req || red_op_B_req) begin
27    AL_if.out_S = 0;
28    AL_if.leds_S = ~AL_if.leds_S;
29  end
30  else if (bypass_A_req || bypass_B_req) begin
31    AL_if.out_S <= A_req;
32  end
33  else if (bypass_A_req) begin
34    AL_if.out_S <= A_req;
35  end
36  else if (bypass_B_req) begin
37    AL_if.out_S <= B_req;
38  end
39  else begin
40    AL_if.out_S <= A_req + B_req + cin_req;
41  end
42 end;
43 S'blk3: begin
44  if (red_op_A_req || red_op_B_req) begin
45    AL_if.out_S = 0;
46    AL_if.leds_S = ~AL_if.leds_S;
47  end
48  else if (bypass_A_req || bypass_B_req) begin
49    AL_if.out_S <= A_req;
50  end
51  else if (bypass_A_req) begin
52    AL_if.out_S <= A_req;
53  end
54  else if (bypass_B_req) begin
55    AL_if.out_S <= B_req;
56  end
57  else begin
58    AL_if.out_S <= A_req + B_req;
59  end
60 end;
61 S'blk4: begin
62  if (red_op_A_req || red_op_B_req) begin
63    AL_if.out_S = 0;
64    AL_if.leds_S = ~AL_if.leds_S;
65  end
66  else if (bypass_A_req || bypass_B_req) begin
67    AL_if.out_S <= A_req;
68  end
69  else if (bypass_A_req) begin
70    AL_if.out_S <= A_req;
71  end
72  else if (bypass_B_req) begin
73    AL_if.out_S <= B_req;
74  end
75  else begin
76    if (direction_req) begin
77      AL_if.out_S <= {AL_if.out_S[4:8], serial_in_req};
78    end
79    else
80      AL_if.out_S <= {serial_in_req, AL_if.out_S[5:1]};
81  end
82 end;
83 S'blk5: begin
84  if (red_op_A_req || red_op_B_req) begin
85    AL_if.out_S = 0;
86    AL_if.leds_S = ~AL_if.leds_S;
87  end
88  else if (bypass_A_req || bypass_B_req) begin
89    AL_if.out_S <= A_req;
90  end
91  else if (bypass_A_req) begin
92    AL_if.out_S <= A_req;
93  end
94  else if (bypass_B_req) begin
95    AL_if.out_S <= B_req;
96  end
97  else begin
98    if (direction_req) begin
99      AL_if.out_S <= {AL_if.out_S[4:8], AL_if.out_S[5]};
100    end
101    else
102      AL_if.out_S <= {AL_if.out_S[0], AL_if.out_S[5:1]};
103  end
104 end;
105 S'blk6: begin
106  AL_if.out_S = 0;
107  AL_if.leds_S = ~AL_if.leds_S;
108 end;
109 S'blk7: begin
110  AL_if.out_S = 0;
111  AL_if.leds_S = ~AL_if.leds_S;
112 end;
113 endcase;
114 end;
115 end
116 endmodule //ALU

```

V. Sequence item

```

1 package ALSU_seq_item_pkg;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4 typedef enum {OR, XOR, ADD, MULT, SHIFT, ROTATE, INVALID_6, INVALID_7 } Opcode_e;
5 localparam MAXPOS=3, MAXNEG=-4, ZERO=0;
6 class ALSU_seq_item extends uvm_sequence_item;
7   `uvm_object_utils(ALSU_seq_item);
8   rand Logic signed [2:0] A, B;
9   rand Opcode_e opcode;
10  rand Logic rst;
11  rand Logic cin,red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
12  logic signed [5:0] out;
13  logic signed [5:0] out_golden;
14  logic [15:0] leds;
15  logic [15:0] leds_golden;
16  function new(string name="ALSU_seq_item");
17    super.new(name);
18  endfunction : new
19  virtual function string convert2string();
20    return
21      $sformatf(" %s reset=%0b , A=%0b , B=%0b , opcode=%0b , cin=%0h red_op_A=%0b red_op_B=%0b bypass_A=%0b bypass_B=%0b direction=%0b serial_in=%0b",
22      super.convert2string(),rst, A, B, opcode, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in);
23  endfunction: convert2string
24  virtual function string convert2string_stimulus();
25    return
26      $sformatf("reset=%0b , A=%0b , B=%0b , opcode=%0b , cin=%0h red_op_A=%0b red_op_B=%0b bypass_A=%0b bypass_B=%0b direction=%0b serial_in=%0b",
27      rst, A,B, opcode, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in);
28  endfunction: convert2string_stimulus
29 //ALSU_4
30 constraint reset {
31   rst dist {0:=99, 1:=1};
32 }
33 constraint ADD_MULT {
34   if( opcode==MULT || opcode==ADD )
35   {
36     A dist { MAXPOS:=30 , ZERO:=30 , MAXNEG:=30 , [-MAXPOS:-1]:=10, [1:2]:=10 };
37     B dist { MAXPOS:=30 , ZERO:=30 , MAXNEG:=30 , [-MAXPOS:-1]:=10, [1:2]:=10 };}}
38
39 constraint OR_XOR {
40   if( ( opcode==OR || opcode==XOR) && red_op_A )
41   {
42     A dist { 2:=90, -4:=90, [-3:0]:=10, 3:=30 };
43     B==ZERO;
44   }
45   if( ( opcode==OR || opcode==XOR) && red_op_B )
46   {
47     B dist{ 2:=90, -4:=90 , [-3:0]:=10, 3:=30 };
48     A==ZERO;
49   }
50 //ALSU_3
51 constraint Opcode_Invalid_Cases{
52   opcode dist { [0:5]:=90 , [6:7]:=10 };
53 }
54 //ALSU_4
55 constraint Bypass_A_B{
56   bypass_B dist {ZERO:=80, 1:=20};
57   bypass_A dist {ZERO:=80, 1:=20};
58 }
59 endclass : ALSU_seq_item
60 endpackage : ALSU_seq_item_pkg

```

VI. Sequence

```
● ● ●
1 package pack_seq;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_seq_item::*;
5 class shift_reg_reset_sequence extends uvm_sequence #(shift_reg_seq_item);
6 `uvm_object_utils(shift_reg_reset_sequence);
7 shift_reg_seq_item seq_item;
8 function new(string name = "shift_reg_reset_sequence");
9 super. new(name);
10 endfunction
11 task body;
12 seq_item = shift_reg_seq_item::type_id:: create("seq_item");
13 start_item(seq_item);
14 seq_item. reset = 1;
15 seq_item. serial_in = 0;
16 seq_item. red_op_B = 0;
17 seq_item. red_op_A = 0;
18 seq_item. bypass_A = 0;
19 seq_item. bypass_B = 0;
20 seq_item. cin = 0;
21 seq_item. direction = 0;
22 seq_item.A = 0;
23 seq_item. B = 0;
24 seq_item. opcode = 0;
25 finish_item(seq_item);
26 endtask
27 endclass
28 class shift_reg_main_sequence extends uvm_sequence #(shift_reg_seq_item);
29 `uvm_object_utils(shift_reg_main_sequence);
30 shift_reg_seq_item seq_item;
31 function new(string name = "shift_reg_main_sequence");
32 super. new(name);
33 endfunction
34 task body;
35 repeat(1000) begin
36 seq_item = shift_reg_seq_item::type_id:: create("seq_item");
37 start_item(seq_item);
38 assert (seq_item. randomize());
39 finish_item(seq_item);
40 end
41 endtask
42 endclass
43 class direct_test_sequence extends uvm_sequence #(shift_reg_seq_item);
44 `uvm_object_utils(direct_test_sequence);
45 shift_reg_seq_item seq_item;
46 function new(string name = "direct_test_sequence");
47 super. new(name);
48 endfunction
49 task body;
50 seq_item = shift_reg_seq_item::type_id:: create("seq_item");
51 start_item(seq_item);
52 seq_item. A = 50;
53 seq_item. B = 50;
54 #1;
55 seq_item. opcode = 3'b000;
56 #2;
57 seq_item. opcode = 3'b001;
58 #2;
59 seq_item. opcode = 3'b010;
60 #2;
61 seq_item. opcode = 3'b011;
62 #2;
63 seq_item. opcode = 3'b100;
64 #2;
65 seq_item. opcode = 3'b101;
66 #2;
67 finish_item(seq_item);
68 endtask
69 endclass
70 endpackage
71
```

VII. Scoreboard

```
1 `include "uvm_macros.svh"
2 package scoreboard;
3   import uvm_pkg::*;
4   import pack_seq_item::*;
5
6   class scoreboard extends uvm_scoreboard;
7     `uvm_component_utils(scoreboard)
8
9     uvm_analysis_export #(shift_reg_seq_item) sb_export;
10    uvm_tlm_analysis_fifo #(shift_reg_seq_item) sb_fifo;
11    shift_reg_seq_item seq_item_sb;
12    int error_count = 0;
13    int correct_count = 0;
14
15   function new(string name = "scoreboard", uvm_component parent = null);
16     super.new(name, parent);
17   endfunction
18
19   function void build_phase(uvm_phase phase);
20     super.build_phase(phase);
21     sb_export = new("sb_export", this);
22     sb_fifo = new("sb_fifo", this);
23   endfunction
24
25   function void connect_phase(uvm_phase phase);
26     super.connect_phase(phase);
27     sb_export.connect(sb_fifo.analysis_export);
28   endfunction
29
30   task run_phase(uvm_phase phase);
31     super.run_phase(phase);
32     forever begin
33       sb_fifo.get(seq_item_sb);
34       if (seq_item_sb.out != seq_item_sb.out_G) begin
35         `uvm_error("run_phase", $sformatf("comparison failed, transaction received by the DUT: %s", seq_item_sb.convert2string()));
36         error_count++;
37       end else begin
38         `uvm_info("run_phase", $sformatf("correct output: %s", seq_item_sb.convert2string()), UVM_LOW);
39         correct_count++;
40       end
41     end
42   endtask
43
44   function void report_phase(uvm_phase phase);
45     super.report_phase(phase);
46     `uvm_info("report_phase", $sformatf("Total successful counts:
47     ", correct_count); $sformatf("Total failed counts:
48     ", error_count); UVM_MEDIUM);
49   endclass
50 endpackage
51
```

VIII. Coverage collector

```
1 package sh_coverage;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_seq_item::*;
5 class ccoverage extends uvm_component;
6 `uvm_component_utils(ccoverage);
7 uvm_analysis_export #(shift_reg_seq_item) cov_export;
8 uvm_tlm_analysis_fifo #(shift_reg_seq_item) cov_fifo;
9 shift_reg_seq_item seq_item_cov;
10 covergroup cvg;
11 A_cp: coverpoint seq_item_cov.A {
12     bins data_0 = {ZERO};
13     bins data_max = {MAXPOS};
14     bins data_min = {MAXNEG};
15     bins data_walkingones = {3'b001, 3'b010, 3'b100};
16     bins data_default = default;
17 }
18 B_cp: coverpoint seq_item_cov.B {
19     bins data_0 = {ZERO};
20     bins data_max = {MAXPOS};
21     bins data_min = {MAXNEG};
22     bins data_walkingones = {3'b001, 3'b010, 3'b100};
23     bins data_default = default;
24 }
25 ALU_cp: coverpoint seq_item_cov.opcode{
26     bins Bins_shift[] = {SHIFT, ROTATE};
27     bins Bins_arith[] = {ADD, MULT};
28     bins Bins_bitwise[] = {OR, XOR};
29     illegal_bins Bins_invalid = {INVALID_6, INVALID_7};
30 }
31 red_A_cp: coverpoint seq_item_cov.red_op_A {
32     bins Bins_red_A = {0, 1};
33 }
34 red_B_cp: coverpoint seq_item_cov.red_op_B {
35     bins Bins_red_B = {0, 1};
36 }
37 cin_cp: coverpoint seq_item_cov.cin {
38     bins Bins_Cin = {0, 1};
39 }
40 serial_in_cp: coverpoint seq_item_cov.serial_in {
41     bins Bins_serial_in = {0, 1};
42 }
43 direction_cp: coverpoint seq_item_cov.direction {
44     bins Bins_Direction = {0, 1};
45 }
46 ALU_cross1: cross A_cp, B_cp, ALU_cp {
47     bins cross_ab_arith = binsof(ALU_cp.Bins_arith) && binsof(A_cp) intersect {ZERO, MAXNEG, MAXPOS}
48         && binsof(B_cp) intersect {ZERO, MAXNEG, MAXPOS};
49     ignore_bins ignore_ab_arith!= binsof(ALU_cp.Bins_arith) && binsof(A_cp) intersect {ZERO, MAXNEG, MAXPOS}&& binsof(B_cp) intersect {ZERO, MAXNEG, MAXPOS} ;
50 }
51 ALU_cross2: cross ALU_cp, cin_cp {
52     bins cross_cin_arith = binsof(ALU_cp.Bins_arith) && binsof(cin_cp.Bins_Cin);
53 }
54 ALU_cross3: cross ALU_cp, serial_in_cp, direction_cp {
55     bins cross_serial_shift = binsof(ALU_cp.Bins_shift) && binsof(serial_in_cp.Bins_serial_in);
56     bins cross_shift_direction = binsof(ALU_cp.Bins_shift) && binsof(direction_cp.Bins_Direction);
57     ignore_bins ignore_serial_shift = !binsof(ALU_cp.Bins_shift);
58     ignore_bins ignore_shift_direction = !binsof(ALU_cp.Bins_shift);
59 }
60 ALU_cross4: cross A_cp, B_cp, ALU_cp, red_A_cp, red_B_cp {
61     bins cross_bitwise_redA = binsof(ALU_cp.Bins_bitwise) && binsof(red_A_cp.Bins_red_A[1])
62         && binsof(A_cp.data_walkingones) && binsof(B_cp.data_0);
63     bins cross_bitwise_redB = binsof(ALU_cp.Bins_bitwise) && binsof(red_B_cp.Bins_red_B[1])
64         && binsof(B_cp.data_walkingones) && binsof(A_cp.data_0);
65     illegal_bins cross_red_A_not_bitwise
66         = binsof(ALU_cp) intersect {ADD, MULT, SHIFT, ROTATE} && binsof(red_A_cp.Bins_red_A[1]);
67     illegal_bins cross_red_B_not_bitwise
68         = binsof(ALU_cp) intersect {ADD, MULT, SHIFT, ROTATE} && binsof(red_B_cp.Bins_red_B[1]);
69 }
70 endgroup
71 function new(string name = "ccoverage", uvm_component parent = null);
72 super. new(name, parent);
73 cvg = new();
74 endfunction
75 function void build_phase (uvm_phase phase);
76 super. build_phase(phase);
77 cov_export = new("cov_export",this);
78 cov_fifo = new("cov_fifo",this);
79 endfunction
80 function void connect_phase(uvm_phase phase);
81 super. connect_phase(phase);
82 cov_export.connect(cov_fifo.analysis_export);
83 endfunction
84 task run_phase (uvm_phase phase);
85 super. run_phase(phase);
86 forever begin
87     cov_fifo.get(seq_item_cov);
88     cvg.sample();
89 end
90 endtask
91 endclass
92 endpackage
```

IX. SVA Module

```
● ● ●
1  module ALSU_assertions (
2      input Logic signed [2:0] A, B,
3      input Logic clk, rst, cin, serial_in, red_op_A, red_op_B,
4      input Logic [2:0] opcode,
5      input Logic bypass_A, bypass_B, direction,
6      input Logic [15:0] leds,
7      input Logic signed [5:0] out
8  );
9      Logic invalid;
10     assign invalid = (opcode == 7 || opcode == 6 || ((opcode != 1 && opcode != 0) && (red_op_B || red_op_A)));
11
12    // Reset check
13    always_comb if (rst) assert final(out == 0 && leds == 0);
14
15    // Combined properties for operations
16    property check_operations;
17        @posedge clk disable iff (rst)
18        (invalid || bypass_A || bypass_B || red_op_A || red_op_B || opcode == 0 || opcode == 1) |->
19        ##2 (
20            (bypass_A ? out == A :
21             bypass_B ? out == B :
22             red_op_A ? out == |A :
23             red_op_B ? out == |B :
24             opcode == 0 ? out == (A | B) :
25             opcode == 1 ? out == (A ^ B) :
26             opcode == 2 ? out[2:0] == ($past(A, 2) + $past(B, 2) + $past(cin, 2)) :
27             opcode == 3 ? out == A * B :
28             opcode == 4 ? out == {direction ? $past(out[4:0]) : serial_in, $past(out[5:1])} : // Shift/Rotate
29             out == {serial_in, $past(out[4:0])} // Rotate
30         )
31        );
32    endproperty
33
34    // Assertions and coverage for operations
35    assert_operations_ap: assert property(check_operations);
36    cover_operations_cp: cover property(check_operations);
37
38    // Invalid opcode check
39    property invalid_opcode;
40        @posedge clk disable iff (rst) (opcode == 6 || opcode == 7) |-> ##2 (out == 0 && leds == ~ $past(leds));
41    endproperty
42
43    // Assertion and coverage for invalid opcode
44    assert_invalid_ap: assert property(invalid_opcode);
45    cover_invalid_cp: cover property(invalid_opcode);
46  endmodule : ALSU_assertions
47
48
```

X. Driver module

```
1 package pack_driver;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_object::*;
5 import pack_sequencer::*;
6 import pack_seq_item::*;
7 class DRIVER extends uvm_driver #(shift_reg_seq_item );
8 `uvm_component_utils(DRIVER);
9 virtual ALSU_if sh_vif;
10 shift_reg_seq_item stim_seq_item;
11 function new(string name = "DRIVER", uvm_component parent = null);
12 super. new(name, parent);
13 endfunction //new()
14 task run_phase (uvm_phase phase);
15 super. run_phase(phase);
16 forever begin
17 stim_seq_item = shift_reg_seq_item::type_id:: create("stim_seq_item");
18 seq_item_port. get_next_item(stim_seq_item);
19 sh_vif. reset = stim_seq_item. reset;
20 sh_vif. direction = stim_seq_item. direction;
21 sh_vif. cin = stim_seq_item. cin;
22 sh_vif. serial_in = stim_seq_item. serial_in;
23 sh_vif. bypass_A = stim_seq_item. bypass_A;
24 sh_vif. bypass_B = stim_seq_item. bypass_B;
25 sh_vif. red_op_A = stim_seq_item. red_op_A;
26 sh_vif. red_op_B = stim_seq_item. red_op_B;
27 sh_vif. A = stim_seq_item. A;
28 sh_vif. B = stim_seq_item. B;
29 sh_vif. opcode = stim_seq_item. opcode;
30 @(negedge sh_vif. clk);
31 stim_seq_item. out = sh_vif. out;
32 stim_seq_item. leds = sh_vif. leds;
33 stim_seq_item. out_G = sh_vif. out_G;
34 stim_seq_item. leds_G = sh_vif. leds_G;
35 seq_item_port. item_done();
36 `uvm_info("run_phase", stim_seq_item. convert2string_stimulus(), UVM_HIGH);
37 end
38 endtask
39 endclass
40 endpackage
41
```

XI. Agent module



```
1 package pack_agent;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_driver::*;
5 import pack_sequencer::*;
6 import pack_seq_item::*;
7 import pack_mon::*;
8 import pack_object::*;
9 class sh_agent extends uvm_agent;
10 `uvm_component_utils(sh_agent);
11 object alsu_config_obj_test;
12 DRIVER alsu_dr;
13 MYSequencer sqr;
14 MONITOR mon;
15 uvm_analysis_port #(shift_reg_seq_item) agt_ap;
16 function new(string name = "sh_agent", uvm_component parent = null);
17 super. new(name, parent);
18 endfunction //new()
19 function void build_phase (uvm_phase phase);
20 super. build_phase(phase);
21 if (!uvm_config_db #(object):: get(this,"","CGO", alsu_config_obj_test)) begin
22 `uvm_fatal("build_phase","DRIVER unable to get the virtual interface");
23 end
24 if (alsu_config_obj_test.sel_mod == UVM_ACTIVE) begin
25 alsu_dr = DRIVER::type_id:: create("alsu_dr",this);
26 sqr = MYSequencer::type_id:: create("sqr",this);
27 end
28 mon = MONITOR::type_id:: create("mon",this);
29 agt_ap = new("agt_ap",this);
30 endfunction
31 function void connect_phase(uvm_phase phase);
32 super. connect_phase(phase);
33 if (alsu_config_obj_test. sel_mod == UVM_ACTIVE) begin
34 alsu_dr.seq_item_port.connect(sqr. seq_item_export);
35 alsu_dr.sh_vif = alsu_config_obj_test.alsu_config_vif;
36 end
37 mon.sh_vif = alsu_config_obj_test. alsu_config_vif;
38 mon. mon_ap. connect(agt_ap);
39 endfunction
40 endclass
41 endpackage
```

XII. Environment module



```
1 `include "uvm_macros.svh"
2 package pack_env;
3 import uvm_pkg::*;
4 import pack_agent::*;
5 import scoreboard::*;
6 import sh_coverage::*;
7 import pack_seq_item::*;
8 class alsu_env extends uvm_env;
9 `uvm_component_utils(alsu_env);
10 sh_agent agt;
11 scoreboard sb;
12 coveragee cov;
13 uvm_analysis_port #(shift_reg_seq_item) agt_ap;
14 function new(string name = "als_env", uvm_component parent = null);
15 super. new(name, parent);
16 endfunction
17 function void build_phase (uvm_phase phase);
18 super. build_phase(phase);
19 agt = sh_agent::type_id:: create("agt",this);
20 sb = scoreboard::type_id:: create("sb",this);
21 cov = coveragee::type_id:: create("cov",this);
22 agt_ap = new("agt_ap",this);
23 endfunction
24 function void connect_phase(uvm_phase phase);
25 super. connect_phase(phase);
26 agt.agt_ap. connect(sb.sb_export);
27 agt.agt_ap. connect (cov.cov_export);
28 endfunction
29 endclass //shift_env
30 endpackage
31
```

XIII. Monitor

```
1 package pack_mon;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_seq_item::*;
5 class MONITOR extends uvm_monitor;
6 `uvm_component_utils(MONITOR);
7 virtual ALSU_if sh_vif;
8 shift_reg_seq_item rsp_seq_item;
9 uvm_analysis_port #(shift_reg_seq_item) mon_ap;
10 function new(string name = "monitor", uvm_component parent = null);
11 super. new(name, parent);
12 endfunction //new()
13 function void build_phase (uvm_phase phase);
14 super. build_phase(phase);
15 mon_ap = new("mon_ap",this);
16 endfunction
17 task run_phase (uvm_phase phase);
18 super. run_phase(phase);
19 forever begin
20   rsp_seq_item = shift_reg_seq_item::type_id:: create("rsp_seq_item");
21   @(negedge sh_vif. clk);
22   rsp_seq_item. reset = sh_vif. reset;
23   rsp_seq_item. direction = sh_vif. direction;
24   rsp_seq_item. cin = sh_vif. cin;
25   rsp_seq_item. serial_in = sh_vif. serial_in;
26   rsp_seq_item. bypass_A = sh_vif. bypass_A;
27   rsp_seq_item. bypass_B = sh_vif. bypass_B;
28   rsp_seq_item. red_op_A = sh_vif. red_op_A;
29   rsp_seq_item. red_op_B = sh_vif. red_op_B;
30   rsp_seq_item. A = sh_vif. A;
31   rsp_seq_item. B = sh_vif. B;
32   rsp_seq_item. opcode = sh_vif. opcode;
33   rsp_seq_item. out = sh_vif. out;
34   rsp_seq_item. leds = sh_vif. leds;
35   rsp_seq_item. out_G = sh_vif. out_G;
36   rsp_seq_item. leds_G = sh_vif. leds_G;
37   mon_ap. write(rsp_seq_item);
38 `uvm_info("run_phase", rsp_seq_item. convert2string(), UVM_HIGH);
39 end
40 endtask
41 endclass
42 endpackage
43
```

XIV. Sequencer

```
● ● ●  
1 package pack_sequencer;  
2 import pack_seq_item::*;  
3 import uvm_pkg::*;  
4 `include "uvm_macros.svh"  
5 class MYSequencer extends uvm_sequencer #(shift_reg_seq_item);  
6 `uvm_component_utils(MYSequencer);  
7 function new(string name = "MYSequencer", uvm_component parent = null);  
8 super. new(name, parent);  
9 endfunction  
10 endclass  
11 endpackage
```

XV. Configuration

```
● ● ●  
1 package pack_object;  
2 `include "uvm_macros.svh"  
3 import uvm_pkg::*;  
4 class object extends uvm_object;  
5 `uvm_object_utils(object);  
6 virtual ALSU_if alsu_config_vif;  
7 uvm_active_passive_enum sel_mod;  
8 function new (string name = "object");  
9 super. new(name);  
10 endfunction  
11 endclass  
12 endpackage  
13
```

XVI. Test bench

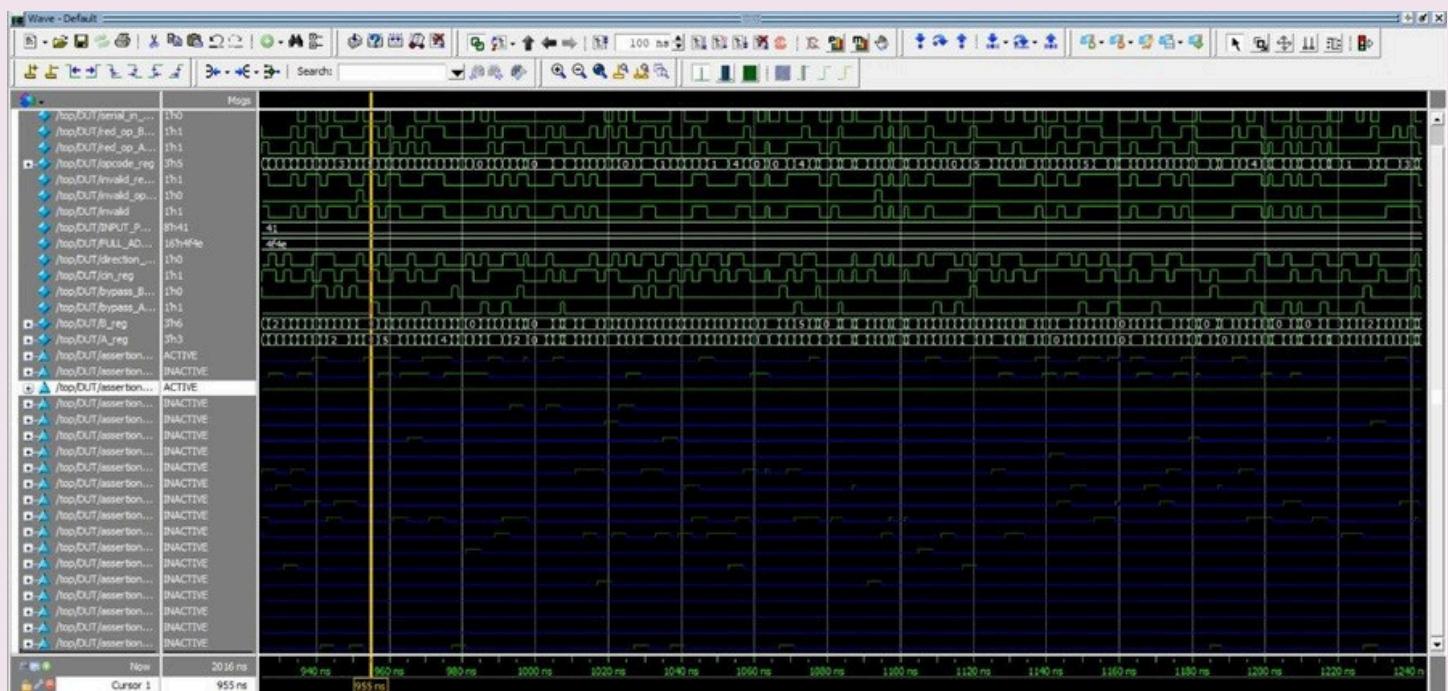


```
1 package pack_test;
2 `include "uvm_macros.svh"
3 import uvm_pkg::*;
4 import pack_env::*;
5 import pack_object::*;
6 import pack_seq::*;
7 class test extends uvm_test;
8 `uvm_component_utils(test)
9 alsu_env env;
10 object alsu_config_obj_test;
11 shift_reg_main_sequence main_seq;
12 shift_reg_reset_sequence reset_seq;
13 direct_test_sequence dirc_seq;
14 function new (string name = "test", uvm_component parent = null);
15 super . new(name, parent);
16 endfunction //new()
17 function void build_phase (uvm_phase phase);
18 super . build_phase(phase);
19 env = alsu_env :: type_id :: create("env",this);
20 alsu_config_obj_test = object :: type_id :: create ("alsu_config_obj_test");
21 main_seq = shift_reg_main_sequence::type_id:: create("main_seq");
22 reset_seq = shift_reg_reset_sequence::type_id:: create("reset_seq");
23 dirc_seq = direct_test_sequence::type_id:: create("dirc_seq");
24 if(! uvm_config_db #(virtual ALSU_if):: get(this,"","",ALSU_if, alsu_config_obj_test. alsu_config_vif))begin
25 `uvm_fatal("build_phase","the test unable to get the virtual interface");
26 end
27 alsu_config_obj_test. sel_mod = UVM_ACTIVE;
28 uvm_config_db #(object) :: set (this,"*","CGO", alsu_config_obj_test);
29 endfunction
30 task run_phase (uvm_phase phase);
31 super . run_phase (phase);
32 phase . raise_objection(this);
33 //reset sequence
34 `uvm_info("run_phase","reset asserted",UVM_LOW);
35 reset_seq. start(env. agt. sqr);
36 `uvm_info("run_phase","reset deasserted",UVM_LOW);
37 //main sequence
38 `uvm_info("run_phase","stimulus generated started",UVM_LOW);
39 main_seq. start(env. agt. sqr);
40 `uvm_info("run_phase","stimulus generated ended",UVM_LOW);
41 //direct test sequence
42 `uvm_info("run_phase","stimulus generated started",UVM_LOW);
43 dirc_seq. start(env. agt. sqr);
44 `uvm_info("run_phase","stimulus generated ended",UVM_LOW);
45 phase . drop_objection(this);
46 endtask //run_phase
47 endclass //test
48 endpackage
49
```

XVII. Print

```
| --- UVM Report Summary ---  
|  
| ** Report counts by severity  
| UVM_INFO : 1020  
| UVM_WARNING : 0  
| UVM_ERROR : 0  
| UVM_FATAL : 0  
| ** Report counts by id  
| [Questa UVM] 2  
| [RNTST] 1  
| [TEST_DONE] 1  
| [report_phase] 2  
| [run_phase] 1014
```

XVIII. Waveform



XIX. Functional Coverage report

```
Coverage Report by instance with details

=====
== Instance: /top/DUT/assertions
== Design Unit: work.SVA
=====

Directive Coverage:
  Directives      19      19      0  100.00%
DIRECTIVE COVERAGE:
-----
Name          Design Design Lang File(Line)    Hits Status
Unit       Unittype
-----
/ttop/DUT/assertions/cv1      SVA Verilog SVA  ALSU_SVA.sv(82)  12 Covered
/ttop/DUT/assertions/cv2      SVA Verilog SVA  ALSU_SVA.sv(83)  10 Covered
/ttop/DUT/assertions/cv3      SVA Verilog SVA  ALSU_SVA.sv(84)  14 Covered
/ttop/DUT/assertions/cv4      SVA Verilog SVA  ALSU_SVA.sv(85)   6 Covered
/ttop/DUT/assertions/cv5      SVA Verilog SVA  ALSU_SVA.sv(86)  77 Covered
/ttop/DUT/assertions/cv6      SVA Verilog SVA  ALSU_SVA.sv(87)  23 Covered
/ttop/DUT/assertions/cv7      SVA Verilog SVA  ALSU_SVA.sv(88)  74 Covered
/ttop/DUT/assertions/cv8      SVA Verilog SVA  ALSU_SVA.sv(89)  88 Covered
/ttop/DUT/assertions/cv9      SVA Verilog SVA  ALSU_SVA.sv(90)  88 Covered
/ttop/DUT/assertions/cv10     SVA Verilog SVA  ALSU_SVA.sv(91)  22 Covered
/ttop/DUT/assertions/cv11     SVA Verilog SVA  ALSU_SVA.sv(92)   9 Covered
/ttop/DUT/assertions/cv12     SVA Verilog SVA  ALSU_SVA.sv(93)  11 Covered
/ttop/DUT/assertions/cv13     SVA Verilog SVA  ALSU_SVA.sv(94)  10 Covered
/ttop/DUT/assertions/cv14     SVA Verilog SVA  ALSU_SVA.sv(95)  11 Covered
/ttop/DUT/assertions/cv15     SVA Verilog SVA  ALSU_SVA.sv(96)   5 Covered
/ttop/DUT/assertions/cv16     SVA Verilog SVA  ALSU_SVA.sv(97)  29 Covered
/ttop/DUT/assertions/cv17     SVA Verilog SVA  ALSU_SVA.sv(98) 105 Covered
/ttop/DUT/assertions/cv18     SVA Verilog SVA  ALSU_SVA.sv(99)  67 Covered
/ttop/DUT/assertions/cv19     SVA Verilog SVA  ALSU_SVA.sv(100) 64 Covered

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

Covergroup Coverage:
  Covergroups      1      na      na  100.00%
  Coverpoints/Crosses 12      na      na
  Covergroup Bins   71      71      0  100.00%
-----
Covergroup          Metric   Goal   Bins   Status
-----
TYPE /sh_coverage/coverage/cvg      100.00%  100      -  Covered
covered/total bins:                71        71      - 
missing/total bins:                 0        71      - 
% Hit:                            100.00%  100      - 
Coverpoint A_cp      100.00%  100      -  Covered
covered/total bins:                3        3      - 
missing/total bins:                 0        3      - 
% Hit:                            100.00%  100      - 
bin data_0                  188        1      -  Covered
bin data_max                 157        1      -  Covered
bin data_walkingones          354        1      -  Covered
default bin data_default        309      -      -  Occurred
Coverpoint B_cp      100.00%  100      -  Covered
covered/total bins:                3        3      - 
missing/total bins:                 0        3      - 
% Hit:                            100.00%  100      - 
bin data_0                  240        1      -  Covered
bin data_max                 154        1      -  Covered
bin data_walkingones          318        1      -  Covered
default bin data_default        296      -      -  Occurred
Coverpoint ALU_cp      100.00%  100      -  Covered
covered/total bins:                6        6      - 
missing/total bins:                 0        6      - 
% Hit:                            100.00%  100      - 
illegal_bin Bins_invalid        32        -      -  Occurred
bin Bins_shift[4]              170        1      -  Covered
bin Bins_shift[5]              173        1      -  Covered
bin Bins_arith[2]              156        1      -  Covered
bin Bins_arith[3]              137        1      -  Covered
bin Bins_bitwise[0]             172        1      -  Covered
bin Bins_bitwise[1]             168        1      -  Covered
Coverpoint red_A_cp      100.00%  100      -  Covered
covered/total bins:                1        1      - 
missing/total bins:                 0        1      - 
% Hit:                            100.00%  100      - 
bin Bins_red_A               1000        1      -  Covered
Coverpoint red_B_cp      100.00%  100      -  Covered
covered/total bins:                1        1      - 
missing/total bins:                 0        1      - 
% Hit:                            100.00%  100      - 
bin Bins_red_B               1000        1      -  Covered
Coverpoint cin_cp      100.00%  100      -  Covered
covered/total bins:                1        1      - 
missing/total bins:                 0        1      - 
% Hit:                            100.00%  100      - 
bin Bins_Cin                1000        1      -  Covered
```

% Hit:	100.00%	100	-	
bin Bins_red_A	1000	1	-	Covered
Coverpoint red_B_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin Bins_red_B	1000	1	-	Covered
Coverpoint cin_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin Bins_Cin	1000	1	-	Covered
Coverpoint serial_in_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin Bins_serial_in	1000	1	-	Covered
Coverpoint direction_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin Bins_Direction	1000	1	-	Covered
Cross ALU_cross1	100.00%	100	-	Covered
covered/total bins:	31	31	-	
missing/total bins:	0	31	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_ab_arith	100	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[1]>	21	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[0]>	21	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_arith[3]>	3	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_shift[5]>	26	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_arith[2]>	7	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_shift[4]>	26	1	-	Covered
bin <data_max,data_walkingones,Bins_bitwise[1]>	3	1	-	Covered
bin <data_0,data_walkingones,Bins_bitwise[1]>	15	1	-	Covered
bin <data_max,data_walkingones,Bins_bitwise[0]>	2	1	-	Covered
bin <data_0,data_walkingones,Bins_bitwise[0]>	19	1	-	Covered
bin <data_max,data_walkingones,Bins_arith[3]>	6	1	-	Covered
bin <data_0,data_walkingones,Bins_arith[3]>	6	1	-	Covered
bin <data_max,data_walkingones,Bins_shift[5]>	7	1	-	Covered
bin <data_0,data_walkingones,Bins_shift[5]>	10	1	-	Covered
bin <data_max,data_walkingones,Bins_arith[2]>	6	1	-	Covered
bin <data_0,data_walkingones,Bins_arith[2]>	6	1	-	Covered
bin <data_max,data_walkingones,Bins_shift[4]>	10	1	-	Covered
bin <data_0,data_walkingones,Bins_shift[4]>	9	1	-	Covered
bin <data_walkingones,data_max,Bins_bitwise[1]>	3	1	-	Covered
bin <data_walkingones,data_0,Bins_bitwise[1]>	49	1	-	Covered
bin <data_walkingones,data_max,Bins_bitwise[0]>	7	1	-	Covered
bin <data_walkingones,data_0,Bins_bitwise[0]>	33	1	-	Covered
bin <data_walkingones,data_max,Bins_arith[3]>	6	1	-	Covered
bin <data_walkingones,data_0,Bins_arith[3]>	7	1	-	Covered
bin <data_walkingones,data_max,Bins_shift[5]>	6	1	-	Covered
bin <data_walkingones,data_0,Bins_shift[5]>	12	1	-	Covered
bin <data_walkingones,data_max,Bins_arith[2]>	6	1	-	Covered
bin <data_walkingones,data_0,Bins_arith[2]>	9	1	-	Covered
bin <data_walkingones,data_max,Bins_shift[4]>	7	1	-	Covered
bin <data_walkingones,data_0,Bins_shift[4]>	6	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ignore_ab_arith	38	-	-	Occurred
Cross ALU_cross2	100.00%	100	-	Covered
covered/total bins:	5	5	-	
missing/total bins:	0	5	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_cin_arith	293	1	-	Covered

missing/total bins:	0	5	-
% Hit:	100.00%	100	-
Auto, Default and User Defined Bins:			
bin cross_cin_arith	293	1	- Covered
bin <Bins_bitwise[1],Bins_Cin>	168	1	- Covered
bin <Bins_bitwise[0],Bins_Cin>	172	1	- Covered
bin <Bins_shift[5],Bins_Cin>	173	1	- Covered
bin <Bins_shift[4],Bins_Cin>	170	1	- Covered
Cross ALU_cross3	100.00%	100	- Covered
covered/total bins:	2	2	-
missing/total bins:	0	2	-
% Hit:	100.00%	100	-
Auto, Default and User Defined Bins:			
bin cross_serial_shift	343	1	- Covered
bin cross_shift_direction	343	1	- Covered
Illegal and Ignore Bins:			
ignore_bin ignore_shift_direction	633	-	Occurred
ignore_bin ignore_serial_shift	633	-	Occurred
Cross ALU_cross4	100.00%	100	- Covered
covered/total bins:	16	16	-
missing/total bins:	0	16	-
% Hit:	100.00%	100	-
Auto, Default and User Defined Bins:			
bin cross_bitwise_redA	82	1	- Covered
bin cross_bitwise_redB	34	1	- Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[1],Bins_red_A,Bins_red_B>	21	1	- Covered
bin <data_max,data_walkingones,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	- Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[0],Bins_red_A,Bins_red_B>	21	1	- Covered
bin <data_max,data_walkingones,Bins_bitwise[0],Bins_red_A,Bins_red_B>	2	1	- Covered
bin <data_walkingones,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	- Covered
bin <data_walkingones,data_max,Bins_bitwise[0],Bins_red_A,Bins_red_B>	7	1	- Covered
bin <data_max,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	- Covered
bin <data_0,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	1	1	- Covered
bin <data_max,data_0,Bins_bitwise[1],Bins_red_A,Bins_red_B>	4	1	- Covered
bin <data_0,data_max,Bins_bitwise[0],Bins_red_A,Bins_red_B>	4	1	- Covered
bin <data_max,data_0,Bins_bitwise[1],Bins_red_A,Bins_red_B>	2	1	- Covered
bin <data_0,data_0,Bins_bitwise[1],Bins_red_A,Bins_red_B>	1	1	- Covered
bin <data_max,data_0,Bins_bitwise[0],Bins_red_A,Bins_red_B>	1	1	- Covered
bin <data_0,data_0,Bins_bitwise[0],Bins_red_A,Bins_red_B>	4	1	- Covered
Illegal and Ignore Bins:			
illegal_bin cross_red_B_not_bitwise	299	-	Occurred
illegal_bin cross_red_A_not_bitwise	299	-	Occurred

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
<hr/>				
TYPE /sh_coverage/ccoverage/cvg	100.00%	100	-	Covered
covered/total bins:	71	71	-	
missing/total bins:	0	71	-	
% Hit:	100.00%	100	-	
Coverpoint A_cp	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
% Hit:	100.00%	100	-	
bin data_0	188	1	-	Covered
bin data_max	157	1	-	Covered
bin data_walkingones	354	1	-	Covered
default bin data_default	309	-		Occurred
Coverpoint B_cp	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
% Hit:	100.00%	100	-	
bin data_0	240	1	-	Covered
bin data_max	154	1	-	Covered
bin data_walkingones	318	1	-	Covered
default bin data_default	296	-		Occurred
Coverpoint ALU_cp	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
illegal_bin Bins_invalid	32	-		Occurred
bin Bins_shift[4]	170	1	-	Covered
bin Bins_shift[5]	173	1	-	Covered
bin Bins_arith[2]	156	1	-	Covered
bin Bins_arith[3]	137	1	-	Covered
bin Bins_bitwise[0]	172	1	-	Covered
bin Bins_bitwise[1]	168	1	-	Covered
Coverpoint red_A_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	

```

% Hit:                                100.00%   100   -   Covered
bin Bins_red_B                         1000    1   -   Covered
Coverpoint cin_cp                      100.00%   100   -   Covered
covered/total bins:                   1       1   -   -
missing/total bins:                  0       1   -   -
% Hit:                                100.00%   100   -   Covered
bin Bins_Cin                           1000    1   -   Covered
Coverpoint serial_in_cp               100.00%   100   -   Covered
covered/total bins:                   1       1   -   -
missing/total bins:                  0       1   -   -
% Hit:                                100.00%   100   -   Covered
bin Bins_serial_in                    1000    1   -   Covered
Coverpoint direction_cp              100.00%   100   -   Covered
covered/total bins:                   1       1   -   -
missing/total bins:                  0       1   -   -
% Hit:                                100.00%   100   -   Covered
bin Bins_Direction                     1000    1   -   Covered
Cross ALU_cross1                      100.00%   100   -   Covered
covered/total bins:                   31      31   -   -
missing/total bins:                  0       31   -   -
% Hit:                                100.00%   100   -   -
Auto, Default and User Defined Bins:
  bin cross_ab_arith                  100     1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_bitwise[1]>
    21      1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_bitwise[0]>
    21      1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_arith[3]>
    3       1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_shift[3]>
    26      1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_arith[2]>
    7       1   -   Covered
  bin <data_walkingones,data_walkingones,Bins_shift[4]>
    26      1   -   Covered
  bin <data_max,data_walkingones,Bins_bitwise[1]>
    3       1   -   Covered
  bin <data_0,data_walkingones,Bins_bitwise[1]>
    15      1   -   Covered
  bin <data_max,data_walkingones,Bins_bitwise[0]>
    2       1   -   Covered
  bin <data_0,data_walkingones,Bins_bitwise[0]>
    19      1   -   Covered
  bin <data_max,data_walkingones,Bins_arith[3]>
    6       1   -   Covered
  bin <data_0,data_walkingones,Bins_arith[3]>
    6       1   -   Covered
  bin <data_max,data_walkingones,Bins_shift[5]>
    7       1   -   Covered
  bin <data_0,data_walkingones,Bins_shift[5]>
    10      1   -   Covered
  bin <data_max,data_walkingones,Bins_arith[2]>
    6       1   -   Covered
  bin <data_0,data_walkingones,Bins_arith[2]>
    6       1   -   Covered
  bin <data_max,data_walkingones,Bins_shift[4]>
    10      1   -   Covered
  bin <data_0,data_walkingones,Bins_shift[4]>
    9       1   -   Covered
  bin <data_walkingones,data_max,Bins_bitwise[1]>
    3       1   -   Covered
  bin <data_walkingones,data_0,Bins_bitwise[1]>
    49      1   -   Covered
  bin <data_walkingones,data_max,Bins_bitwise[0]>
    7       1   -   Covered
  bin <data_walkingones,data_0,Bins_bitwise[0]>
    33      1   -   Covered
  bin <data_walkingones,data_max,Bins_arith[3]>
    6       1   -   Covered
  bin <data_walkingones,data_0,Bins_arith[3]>
    7       1   -   Covered
  bin <data_walkingones,data_max,Bins_shift[5]>
    6       1   -   Covered
  bin <data_walkingones,data_0,Bins_shift[5]>
    12      1   -   Covered
  bin <data_walkingones,data_max,Bins_arith[2]>
    6       1   -   Covered
  bin <data_walkingones,data_0,Bins_arith[2]>
    9       1   -   Covered
  bin <data_walkingones,data_max,Bins_shift[4]>
    7       1   -   Covered
  bin <data_walkingones,data_0,Bins_shift[4]>
    6       1   -   Covered
Illegal and Ignore Bins:
  ignore_bin ignore_ab_arith            38     -   -   Occurred
Coverpoint Cross ALU_cross2           100.00%   100   -   Covered
covered/total bins:                   5       5   -   -
missing/total bins:                  0       5   -   -
% Hit:                                100.00%   100   -   -
Auto, Default and User Defined Bins:
  bin cross_cin_arith                 293     1   -   Covered
  bin <Bins_bitwise[1],Bins_Cin>
    168     1   -   Covered
  bin <Bins_bitwise[0],Bins_Cin>
    172     1   -   Covered
  bin <Bins_shift[5],Bins_Cin>
    173     1   -   Covered
  bin <Bins_shift[4],Bins_Cin>
    178     1   -   Covered
Cross ALU cross3                      100.00%   100   -   Covered

```

bin <data_walkingones,data_0,Bins_shift[4]>	7	1	-	Covered
	6	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ignore_ab_arith	38		-	Occurred
Cross ALU_cross2	100.00%	100	-	Covered
covered/total bins:	5	5	-	
missing/total bins:	0	5	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_cin_arith	293	1	-	Covered
bin <Bins_bitwise[1],Bins_Cin>	168	1	-	Covered
bin <Bins_bitwise[0],Bins_Cin>	172	1	-	Covered
bin <Bins_shift[5],Bins_Cin>	173	1	-	Covered
bin <Bins_shift[4],Bins_Cin>	170	1	-	Covered
Cross ALU_cross3	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_serial_shift	343	1	-	Covered
bin cross_shift_direction	343	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ignore_shift_direction	633		-	Occurred
ignore_bin ignore_serial_shift	633		-	Occurred
Cross ALU_cross4	100.00%	100	-	Covered
covered/total bins:	16	16	-	
missing/total bins:	0	16	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin cross_bitwise_redA	82	1	-	Covered
bin cross_bitwise_redB	34	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[1],Bins_red_A,Bins_red_B>	21	1	-	Covered
bin <data_max,data_walkingones,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	-	Covered
bin <data_walkingones,data_walkingones,Bins_bitwise[0],Bins_red_A,Bins_red_B>	21	1	-	Covered
bin <data_max,data_walkingones,Bins_bitwise[0],Bins_red_A,Bins_red_B>	2	1	-	Covered
bin <data_walkingones,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	-	Covered
bin <data_walkingones,data_max,Bins_bitwise[0],Bins_red_A,Bins_red_B>	7	1	-	Covered
bin <data_max,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	3	1	-	Covered
bin <data_0,data_max,Bins_bitwise[1],Bins_red_A,Bins_red_B>	1	1	-	Covered
bin <data_max,data_max,Bins_bitwise[0],Bins_red_A,Bins_red_B>	4	1	-	Covered
bin <data_0,data_max,Bins_bitwise[0],Bins_red_A,Bins_red_B>	4	1	-	Covered
bin <data_max,data_0,Bins_bitwise[1],Bins_red_A,Bins_red_B>	2	1	-	Covered
bin <data_0,data_0,Bins_bitwise[1],Bins_red_A,Bins_red_B>	1	1	-	Covered
bin <data_max,data_0,Bins_bitwise[0],Bins_red_A,Bins_red_B>	1	1	-	Covered
bin <data_0,data_0,Bins_bitwise[0],Bins_red_A,Bins_red_B>	4	1	-	Covered
Illegal and Ignore Bins:				
illegal_bin cross_red_B_not_bitwise	299		-	Occurred
illegal_bin cross_red_A_not_bitwise	299		-	Occurred

TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1

DIRECTIVE COVERAGE:

Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status
<hr/>						
/top/DUT/assertions/cv1	SVA	Verilog	SVA	ALSU_SVA.sv(82)	12	Covered
/top/DUT/assertions/cv2	SVA	Verilog	SVA	ALSU_SVA.sv(83)	10	Covered
/top/DUT/assertions/cv3	SVA	Verilog	SVA	ALSU_SVA.sv(84)	14	Covered
/top/DUT/assertions/cv4	SVA	Verilog	SVA	ALSU_SVA.sv(85)	6	Covered
/top/DUT/assertions/cv5	SVA	Verilog	SVA	ALSU_SVA.sv(86)	77	Covered
/top/DUT/assertions/cv6	SVA	Verilog	SVA	ALSU_SVA.sv(87)	23	Covered
/top/DUT/assertions/cv7	SVA	Verilog	SVA	ALSU_SVA.sv(88)	74	Covered
/top/DUT/assertions/cv8	SVA	Verilog	SVA	ALSU_SVA.sv(89)	88	Covered
/top/DUT/assertions/cv9	SVA	Verilog	SVA	ALSU_SVA.sv(90)	88	Covered
/top/DUT/assertions/cv10	SVA	Verilog	SVA	ALSU_SVA.sv(91)	22	Covered
/top/DUT/assertions/cv11	SVA	Verilog	SVA	ALSU_SVA.sv(92)	9	Covered
/top/DUT/assertions/cv12	SVA	Verilog	SVA	ALSU_SVA.sv(93)	11	Covered
/top/DUT/assertions/cv13	SVA	Verilog	SVA	ALSU_SVA.sv(94)	10	Covered
/top/DUT/assertions/cv14	SVA	Verilog	SVA	ALSU_SVA.sv(95)	11	Covered
/top/DUT/assertions/cv15	SVA	Verilog	SVA	ALSU_SVA.sv(96)	5	Covered
/top/DUT/assertions/cv16	SVA	Verilog	SVA	ALSU_SVA.sv(97)	29	Covered
/top/DUT/assertions/cv17	SVA	Verilog	SVA	ALSU_SVA.sv(98)	105	Covered
/top/DUT/assertions/cv18	SVA	Verilog	SVA	ALSU_SVA.sv(99)	67	Covered
/top/DUT/assertions/cv19	SVA	Verilog	SVA	ALSU_SVA.sv(100)	64	Covered

TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 19

Total Coverage By Instance (filtered view): 100.00%