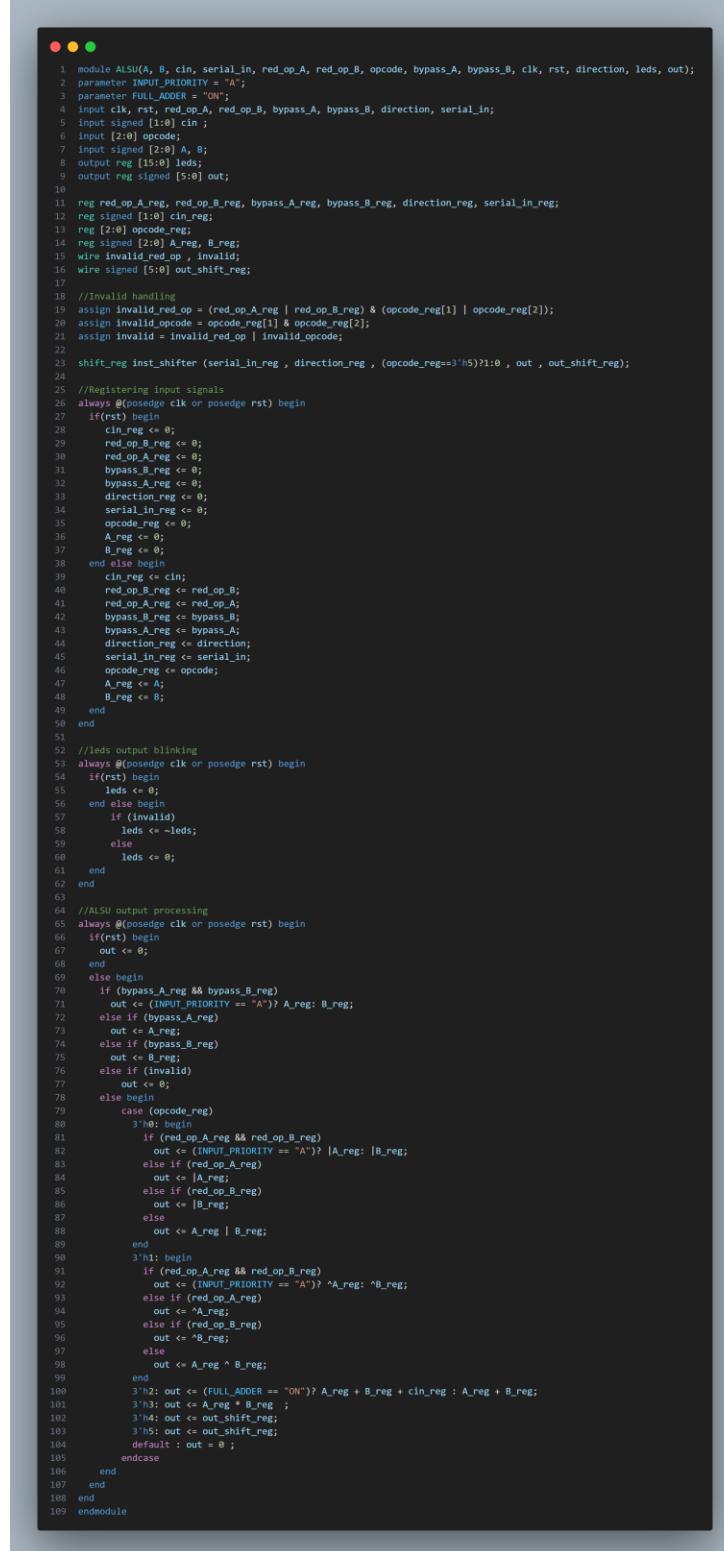


ALSU/SHIFTER environment

1. Design



```
1 module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
2 parameter INPUT_PRIORITY = "A";
3 parameter FULL_ADDER = "ON";
4 input clk, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
5 input signed [1:0] cin ;
6 input [2:0] opcode;
7 input signed [2:0] A, B;
8 output reg [15:0] leds;
9 output reg signed [5:0] out;
10
11 reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
12 reg signed [1:0] cin_reg;
13 reg [2:0] opcode_reg;
14 reg signed [2:0] A_reg, B_reg;
15 wire invalid_red_op , invalid;
16 wire signed [5:0] out_shift_reg;
17
18 //Invalid handling
19 assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
20 assign invalid_opcode = opcode_Reg[1] & opcode_Reg[2];
21 assign invalid = invalid_red_op | invalid_opcode;
22
23 shift_reg inst_shifter (serial_in_reg , direction_reg , (opcode_reg==3'h5)?1:0 , out , out_shift_reg);
24
25 //Registering input signals
26 always @ (posedge clk or posedge rst) begin
27   if(rst) begin
28     cin_reg <= 0;
29     red_op_B_reg <= 0;
30     red_op_A_reg <= 0;
31     bypass_B_reg <= 0;
32     bypass_A_reg <= 0;
33     direction_reg <= 0;
34     serial_in_reg <= 0;
35     opcode_reg <= 0;
36     A_reg <= 0;
37     B_reg <= 0;
38   end else begin
39     cin_reg <= cin;
40     red_op_B_reg <= red_op_B;
41     red_op_A_reg <= red_op_A;
42     bypass_B_reg <= bypass_B;
43     bypass_A_reg <= bypass_A;
44     direction_reg <= direction;
45     serial_in_reg <= serial_in;
46     opcode_reg <= opcode;
47     A_reg <= A;
48     B_reg <= B;
49   end
50 end
51
52 //leds output blinking
53 always @ (posedge clk or posedge rst) begin
54   if(rst) begin
55     leds <= 0;
56   end else begin
57     if (invalid)
58       leds <= ~leds;
59     else
60       leds <= 0;
61   end
62 end
63
64 //ALSU output processing
65 always @ (posedge clk or posedge rst) begin
66   if(rst) begin
67     out <= 0;
68   end
69   else begin
70     if (bypass_A_reg && bypass_B_reg)
71       out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
72     else if (bypass_A_reg)
73       out <= A_reg;
74     else if (bypass_B_reg)
75       out <= B_reg;
76     else if (invalid)
77       out <= 0;
78     else begin
79       case (opcode_reg)
80         3'h0: begin
81           if (red_op_A_reg && red_op_B_reg)
82             out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
83           else if (red_op_A_reg)
84             out <= |A_reg;
85           else if (red_op_B_reg)
86             out <= |B_reg;
87           else
88             out <= A_reg | B_reg;
89         end
90         3'h1: begin
91           if (red_op_A_reg && red_op_B_reg)
92             out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
93           else if (red_op_A_reg)
94             out <= ^A_reg;
95           else if (red_op_B_reg)
96             out <= ^B_reg;
97           else
98             out <= A_reg ^ B_reg;
99         end
100        3'h2: out <= (FULL_ADDER == "ON")? A_reg + B_reg + cin_reg : A_reg + B_reg;
101        3'h3: out <= A_reg * B_reg ;
102        3'h4: out <= out_shift_reg;
103        3'h5: out <= out_shift_reg;
104        default : out = 0 ;
105      endcase
106    end
107  end
108 end
109 endmodule
```

2. ALSU_Top :

```
● ○ ●
1 import uvm_pkg::*;
2 `include "uvm_macros.svh"
3 import ALSU_test_pkg::*;
4
5 module top();
6   bit clk ,reset ;
7
8   initial begin
9     forever
10       #1 clk =~clk ;
11   end
12
13   ALSU_if ALSU_inst_if (clk) ;
14   shift_reg_if SHIFTER_inst_if () ;
15
16   shift_reg SHIFTER_inst_DUT (SHIFTER_inst_if.serial_in, SHIFTER_inst_if.direction,
17                               SHIFTER_inst_if.mode, SHIFTER_inst_if.datain, SHIFTER_inst_if.dataout);
18
19   ALSU ALSU_inst_DUT (ALSU_inst_if.A, ALSU_inst_if.B, ALSU_inst_if.cin,
20                       ALSU_inst_if.serial_in, ALSU_inst_if.red_op_A,
21                       ALSU_inst_if.red_op_B, ALSU_inst_if.opcode,
22                       ALSU_inst_if.bypass_A, ALSU_inst_if.bypass_B,
23                       ALSU_inst_if.clk, ALSU_inst_if.rst, ALSU_inst_if.direction,
24                       ALSU_inst_if.leds, ALSU_inst_if.out);
25
26   assign SHIFTER_inst_if.serial_in = ALSU_inst_DUT.serial_in;
27   assign SHIFTER_inst_if.direction = ALSU_inst_DUT.direction;
28   assign SHIFTER_inst_if.mode = ALSU_inst_DUT.opcode;
29   assign SHIFTER_inst_if.datain = ALSU_inst_DUT.out;
30   assign SHIFTER_inst_if.reset = ALSU_inst_DUT.rst;
31
32
33   bind ALSU_inst_DUT ALSU_assertions inst_assertion (
34     .clk(ALSU_inst_if.clk),
35     .rst(ALSU_inst_if.rst),
36     .out(ALSU_inst_if.out),
37     .leds(ALSU_inst_if.leds),
38     .opcode(ALSU_inst_if.opcode),
39     .bypass_A(ALSU_inst_if.bypass_A),
40     .bypass_B(ALSU_inst_if.bypass_B),
41     .red_op_A(ALSU_inst_if.red_op_A),
42     .red_op_B(ALSU_inst_if.red_op_B),
43     .direction(ALSU_inst_if.direction),
44     .A(ALSU_inst_if.A),
45     .B(ALSU_inst_if.B),
46     .cin(ALSU_inst_if.cin),
47     .serial_in(ALSU_inst_if.serial_in)
48 );
49
50
51   initial begin
52     uvm_config_db #(virtual ALSU_if)::set(null, "uvm_test_top" , "ALSU_if" ,ALSU_inst_if) ;
53     uvm_config_db #(virtual shift_reg_if)::set(null, "uvm_test_top" , "SHIFTER_if" ,SHIFTER_inst_if) ;
54     run_test("ALSU_test");
55   end
56 endmodule
```

3. ALSU_if:

```
● ○ ●
1 interface ALSU_if (clk);
2   input clk ;
3   logic rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
4   logic signed [1:0] cin ;
5   logic [2:0] opcode;
6   logic signed [2:0] A, B;
7   logic [15:0] leds;
8   logic signed [5:0] out;
9 endinterface
```

4. ALSU_test:

```
 1 package ALSU_test_pkg;
 2   import ALSU_env_pkg::*;
 3   import shift_reg_env_pkg::*;
 4   import ALSU_config_pkg::*;
 5   import shift_reg_config_pkg::*;
 6   import ALSU_seq_1_pkg::*;
 7   import ALSU_seq_2_pkg::*;
 8   import ALSU_reset_seq_pkg::*;
 9   import uvm_pkg::*;
10   `include "uvm_macros.svh"
11
12 class ALSU_test extends uvm_test;
13   `uvm_component_utils(ALSU_test)
14   ALSU_env env ;
15   shift_reg_env env_shift;
16   ALSU_config ALSU_cfg ;
17   shift_reg_config shift_reg_cfg;
18   ALSU_seq_1 seq_1;
19   ALSU_seq_2 seq_2;
20   ALSU_seq_reset rst_seq;
21
22   function new (string name = "ALSU_test" , uvm_component parent = null);
23     super.new (name , parent) ;
24   endfunction
25
26   function void build_phase (uvm_phase phase);
27     super.build_phase(phase) ;
28     env = ALSU_env::type_id::create("env",this);
29     env_shift = shift_reg_env::type_id::create("env_shift",this);
30     ALSU_cfg = ALSU_config::type_id::create("ALSU_cfg");
31     shift_reg_cfg = shift_reg_config::type_id::create("shift_reg_cfg");
32     seq_1 = ALSU_seq_1::type_id::create("seq_1");
33     seq_2 = ALSU_seq_2::type_id::create("seq_2");
34     rst_seq = ALSU_seq_reset::type_id::create("rst_seq");
35
36     if (!uvm_config_db #(virtual ALSU_if)::get(this, "" , "ALSU_if" , ALSU_cfg.ALSU_vif))
37       `uvm_fatal("build_phase" , "TEST - unable to get the virtual intrface of ALSU");
38
39     if (!uvm_config_db #(virtual shift_reg_if)::get(this, "" , "SHIFTER_if" , shift_reg_cfg.shift_reg_vif))
40       `uvm_fatal("build_phase" , "TEST - unable to get the virtual intrface of shift_reg");
41
42     ALSU_cfg.is_active = UVM_ACTIVE;
43     shift_reg_cfg.is_active = UVM_PASSIVE;
44
45     uvm_config_db #(ALSU_config)::set(this , "*" , "CFG" , ALSU_cfg);
46     uvm_config_db #(shift_reg_config)::set(this , "*" , "shift_CFG" , shift_reg_cfg);
47
48   endfunction
49
50   task run_phase (uvm_phase phase);
51     super.run_phase(phase);
52     phase.raise_objection(this);
53     `uvm_info("run_phase", "reset asserted" , UVM_MEDIUM)
54     rst_seq.start(env.agt.sqr);
55     `uvm_info("run_phase", "reset deasserted" , UVM_MEDIUM)
56
57     `uvm_info("run_phase", "seq_1 stimulis generated started " , UVM_MEDIUM)
58     seq_1.start(env.agt.sqr);
59     `uvm_info("run_phase", "seq_1 stimulis generated ended" , UVM_MEDIUM)
60
61     `uvm_info("run_phase", "seq_2 stimulis generated started" , UVM_MEDIUM)
62     seq_2.start(env.agt.sqr);
63     `uvm_info("run_phase", "seq_2 stimulis generated ended" , UVM_MEDIUM)
64     phase.drop_objection(this);
65   endtask: run_phase
66 endclass
67
68 endpackage
```

5. ALSU_configuration:

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

6. ALSU_seq_item

```
1 package ALSU_item_pkg;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4
5 parameter MAXPOS = 3 ;
6 parameter MAXNEG = -4 ;
7 typedef enum logic [2:0] {
8   OR, XOR, ADD, MULT, SHIFT, ROTATE, INVALID_6, INVALID_7
9 } opcode_t ;
10
11 class ALSU_item extends uvm_sequence_item;
12   `uvm_object_utils(ALSU_item)
13   rand logic rst ;
14   rand logic signed [2:0] A ;
15   rand logic signed [2:0] B ;
16   rand logic signed [1:0] cin ;
17   rand logic serial_in ;
18   rand logic red_op_A ;
19   rand logic red_op_B ;
20   rand logic bypass_A ;
21   rand logic bypass_B ;
22   rand logic direction;
23   rand opcode_t opcode ;
24   rand opcode_t [6] opcode_arr ;
25   logic [15:0] leds;
26   logic signed [5:0] out;
27
28   function new(string name = "ALSU_item");
29     super.new(name);
30   endfunction
31
32   function string convert2string();
33     return $sformatf("%s reset = %b , A = %b , B = %b , cin = %b , serial_in = %b , red_op_A = %b , red_op_B = %b ,
34                      bypass_A = %b , bypass_B = %b , direction = %b , opcode = %b " , super.convert2string() ,
35                      rst , A,B,cin , serial_in , red_op_A , red_op_B , bypass_A , bypass_B , direction , opcode);
36   endfunction
37
38   function string convert2string_stimulus();
39     return $sformatf("reset = %b , A = %b , B = %b , cin = %b , serial_in = %b , red_op_A = %b , red_op_B = %b ,
40                      bypass_A = %b , bypass_B = %b , direction = %b , opcode = %b , leds = %b , out = %d " ,
41                      rst , A,B,cin , serial_in , red_op_A , red_op_B , bypass_A , bypass_B , direction , opcode , leds , out);
42   endfunction
```

```
1      constraint Arethmatic {
2          if (opcode == ADD || opcode == MULT) {
3              A dist {-4:/25, 0:/25, 3:/25, -3:/5, -2:/5, -1:/5, 1:/5, 2:/5};
4              B dist {-4:/25, 0:/25, 3:/25, -3:/5, -2:/5, -1:/5, 1:/5, 2:/5};
5          }
6      }
7
8      constraint reduction_A {
9          if ( (opcode inside {XOR , OR}) && red_op_A ) {
10             A dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
11                     3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
12             B dist {0:/100} ;
13         }
14     }
15
16     constraint reduction_B {
17         if ( (opcode inside {XOR , OR}) && red_op_B ) {
18             B dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
19                     3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
20             A dist {0:/100} ;
21         }
22     }
23
24     constraint OPCODE {
25         opcode dist { [0:5] :/ 95, [6:7] :/ 5 };
26     }
27
28     constraint BYPASS {
29         bypass_A dist {0:/95 , 1:/5} ;
30         bypass_B dist {0:/95 , 1:/5} ;
31     }
32
33     constraint RESET {
34         rst dist {0:/99 , 1:/1} ;
35     }
36
37     constraint RED_OP {
38         if (opcode == OR || opcode == XOR ) {
39             red_op_A dist {0:/20 , 1:/80} ;
40             red_op_B dist {0:/30 , 1:/70} ;
41         }
42         else {
43             red_op_A dist {0:/95 , 1:/5} ;
44             red_op_B dist {0:/95 , 1:/5} ;
45         }
46     }
47
48     constraint OPCODE_array {
49         foreach (opcode_arr[i])
50             opcode_arr[i] inside {OR, XOR, ADD, MULT, SHIFT, ROTATE};
51
52         foreach (opcode_arr[i])
53             foreach (opcode_arr[j])
54                 if (i != j) opcode_arr[i] != opcode_arr[j];
55     }
56
57     endclass
58 endpackage
```

7. ALSU_reset_sequence:



```
1 package ALSU_reset_seq_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import ALSU_item_pkg::*;
5   class ALSU_seq_reset extends uvm_sequence #(ALSU_item);
6     `uvm_object_utils(ALSU_seq_reset)
7     ALSU_item seq_item ;
8
9     function new(string name = "ALSU_seq_reset");
10      super.new(name);
11    endfunction
12
13    task body;
14      seq_item = ALSU_item ::type_id::create("seq_item");
15      start_item(seq_item);
16      seq_item.rst = 1 ;
17      finish_item(seq_item);
18    endtask
19  endclass
20 endpackage
```

8. ALSU_sequence1



```
1 package ALSU_seq_1_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import ALSU_item_pkg::*;
5   class ALSU_seq_1 extends uvm_sequence #(ALSU_item);
6     `uvm_object_utils(ALSU_seq_1)
7     ALSU_item seq_item ;
8
9     function new(string name = "ALSU_seq_1");
10      super.new(name);
11    endfunction
12
13    task body;
14      seq_item = ALSU_item ::type_id::create("seq_item");
15      seq_item.OPCODE_array.constraint_mode (0) ;
16      repeat (1000) begin
17        start_item(seq_item);
18        assert(seq_item.randomize);
19        finish_item(seq_item);
20      end
21    endtask
22  endclass
23 endpackage
```

9. ALSU_sequence2:

```
● ● ●  
1 package ALSU_seq_2_pkg;  
2     import uvm_pkg::*;  
3     `include "uvm_macros.svh"  
4     import ALSU_item_pkg::*;  
5     class ALSU_seq_2 extends uvm_sequence #(ALSU_item);  
6         `uvm_object_utils(ALSU_seq_2)  
7             ALSU_item seq_item ;  
8  
9         function new(string name = "ALSU_seq_2");  
10            super.new(name);  
11        endfunction  
12  
13        task body;  
14            seq_item = ALSU_item ::type_id::create("seq_item");  
15            seq_item.constraint_mode (0) ;  
16            seq_item.OPCODE_array.constraint_mode (1) ;  
17            repeat (1000) begin  
18                start_item(seq_item);  
19                assert(seq_item.randomize()) with {  
20                    rst == 0;  
21                    bypass_A == 0;  
22                    bypass_B == 0;  
23                    red_op_A == 0;  
24                    red_op_B == 0;  
25                };  
26                finish_item(seq_item);  
27            end  
28        endtask  
29    endclass  
30 endpackage
```

10.ALSU_sequencer

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

11. ALSU_env:

```
● ● ●
1 package ALSU_env_pkg;
2   import ALSU_scoreboard_pkg::*;
3   import ALSU_agent_pkg::*;
4   import ALSU_coverage_pkg::*;
5   import uvm_pkg::*;
6   `include "uvm_macros.svh"
7
8   class ALSU_env extends uvm_env;
9
10   `uvm_component_utils(ALSU_env)
11   ALSU_agent agt ;
12   ALSU_coverage cov ;
13   ALSU_scoreboard sb ;
14
15   function new (string name = "ALSU_env" , 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     agt = ALSU_agent::type_id::create("agt",this);
22     cov = ALSU_coverage::type_id::create("cov",this);
23     sb = ALSU_scoreboard::type_id::create("sb",this);
24   endfunction
25
26   function void connect_phase (uvm_phase phase);
27     super.connect_phase(phase) ;
28     agt.agt_ap.connect(sb.sb_export);
29     agt.agt_ap.connect(cov.cov_export);
30   endfunction
31 endclass
32 endpackage
```

12. ALSU_agent

```
 1 package ALSU_agent_pkg;
 2     import uvm_pkg::*;
 3     `include "uvm_macros.svh"
 4     import ALSU_sequencer_pkg::*;
 5     import ALSU_driver_pkg::*;
 6     import ALSU_config_pkg::*;
 7     import ALSU_monitor_pkg::*;
 8     import ALSU_item_pkg::*;

 9
10    class ALSU_agent extends uvm_agent;
11        `uvm_component_utils(ALSU_agent)
12        ALSU_sequencer sqr ;
13        ALSU_driver drv ;
14        ALSU_monitor mon;
15        ALSU_config ALSU_cfg;
16        uvm_analysis_port #(ALSU_item) agt_ap;
17
18        function new(string name = "ALSU_agent" , uvm_component parent = null );
19            super.new(name, parent);
20        endfunction
21
22        function void build_phase (uvm_phase phase);
23            super.build_phase(phase);
24            if (!uvm_config_db #(ALSU_config)::get(this, "", "CFG", ALSU_cfg)) begin
25                `uvm_fatal("build_phase", "ALSU_AGENT - unable to get configuration object");
26            end
27            mon = ALSU_monitor::type_id::create("mon",this);
28            agt_ap = new("agt_ap",this);
29            if (ALSU_cfg.is_active == UVM_ACTIVE) begin
30                drv = ALSU_driver::type_id::create("drv",this);
31                sqr = ALSU_sequencer::type_id::create("sqr",this);
32            end
33        endfunction
34
35        function void connect_phase(uvm_phase phase);
36            super.connect_phase(phase);
37            mon.ALSU_vif = ALSU_cfg.ALSU_vif;
38            mon.mon_ap.connect(agt_ap);
39            if (ALSU_cfg.is_active == UVM_ACTIVE) begin
40                drv.ALSU_vif = ALSU_cfg.ALSU_vif;
41                drv.seq_item_port.connect(sqr.seq_item_export);
42            end
43        endfunction
44
45    endclass
46 endpackage
```

13. ALSU_driver:

```
● ● ●

1 package ALSU_driver_pkg;
2     import uvm_pkg::*;
3     `include "uvm_macros.svh"
4     import ALSU_item_pkg::*;
5
6     class ALSU_driver extends uvm_driver #(ALSU_item);
7         `uvm_component_utils(ALSU_driver)
8         virtual ALSU_if ALSU_vif;
9         ALSU_item stim_seq_item;
10
11        function new (string name = "ALSU_driver", uvm_component parent = null);
12            super.new(name, parent);
13        endfunction
14
15        task run_phase(uvm_phase phase);
16            super.run_phase(phase);
17            forever begin
18                stim_seq_item = ALSU_item::type_id::create("stim_seq_item");
19                seq_item_port.get_next_item(stim_seq_item);
20                ALSU_vif.rst = stim_seq_item.rst ;
21                ALSU_vif.red_op_A = stim_seq_item.red_op_A;
22                ALSU_vif.red_op_B = stim_seq_item.red_op_B;
23                ALSU_vif.bypass_A = stim_seq_item.bypass_A;
24                ALSU_vif.bypass_B = stim_seq_item.bypass_B;
25                ALSU_vif.direction = stim_seq_item.direction;
26                ALSU_vif.serial_in = stim_seq_item.serial_in;
27                ALSU_vif.cin = stim_seq_item.cin;
28                ALSU_vif.A = stim_seq_item.A;
29                ALSU_vif.B = stim_seq_item.B;
30                ALSU_vif.opcode = stim_seq_item.opcode;
31                @(negedge ALSU_vif.clk);
32                seq_item_port.item_done();
33            end
34        endtask
35    endclass
36 endpackage
```

14. ALSU_monitor:

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

15. ALSU_scoreboard:

```
● ● ●
1 package ALSU_scoreboard_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import ALSU_item_pkg::*;

5
6   class ALSU_scoreboard extends uvm_scoreboard;
7     `uvm_component_utils(ALSU_scoreboard)
8     uvm_analysis_export #(ALSU_item) sb_export;
9     uvm_tlm_analysis_fifo #(ALSU_item) sb_fifo;
10    ALSU_item seq_item_sb;
11    logic signed [5:0] out_ref;
12    logic [15:0] leds_ref ;
13    bit invalid ;
14
15    ALSU_item ref_pipe[$];
16    ALSU_item tmp;
17
18    int error_count = 0 ;
19    int correct_count = 0 ;
20
21    function new (string name = "ALSU_scoreboard" , uvm_component parent = null);
22      super.new(name , parent) ;
23    endfunction
24
25    function void build_phase (uvm_phase phase);
26      super.build_phase(phase);
27      sb_export = new("sb_export",this);
28      sb_fifo = new("sb_fifo",this);
29    endfunction
30
31    function void connect_phase(uvm_phase phase);
32      super.connect_phase(phase);
33      sb_export.connect(sb_fifo.analysis_export);
34    endfunction
35
36    function void report_phase(uvm_phase phase);
37      super.report_phase(phase);
38      `uvm_info("SCOREBOARD", $sformatf(
39        "Simulation Summary: correct_count=
40        , error_count", correct_count,
41        ", error_count), UVM_NONE)
42    endfunction
```

```
● ● ●
1 task run_phase (uvm_phase phase);
2     super.run_phase(phase);
3     forever begin
4         tmp = ALSU_item::type_id::create("tmp");
5         sb_fifo.get(seq_item_sb);
6         seq_item_sb.copy(tmp);
7         ref_pipe.push_back(tmp);
8         if (ref_pipe.size() >= 2) begin
9             ALSU_item delayed_item;
10            delayed_item = ref_pipe.pop_front();
11            ref_seq_item_chk_model(delayed_item);
12            if (delayed_item.out != out_ref || delayed_item.leds != leds_ref) begin
13                `uvm_error("SCOREBOARD", $sformatf(
14                    "Comparison FAIL: opcode=
15                    , A= %0h, expected %0h,
16                    , B= %0h, delayed_item.out, delayed_item.leds, out_ref, leds_ref))
17                , got out= error_count++;
18                leds=%0h, expected %0h=
19                leds=%0h", correct_count++);
20            end
21        end
22    endtask : run_phase
23
24
25 task ref_seq_item_chk_model(ALSU_item seq_item_chk);
26     leds_ref = 0 ;
27     invalid = (seq_item_sb.opcode == 3'b110) ||
28                 (seq_item_sb.opcode == 3'b111) ||
29                 ((seq_item_sb.red_op_A || seq_item_sb.red_op_B) &&
30                  (seq_item_sb.opcode != 3'b000) && (seq_item_sb.opcode != 3'b001)) ;
31     if (seq_item_sb.rst) begin
32         out_ref = 0 ;
33     end
34     else if (seq_item_sb.bypass_A) out_ref = seq_item_sb.A;
35     else if (seq_item_sb.bypass_B) out_ref = seq_item_sb.B;
36     else if (invalid) out_ref = 0;
37     else if (seq_item_sb.opcode == 0) begin
38         if (seq_item_sb.red_op_A) out_ref = |seq_item_sb.A;
39         else if (seq_item_sb.red_op_B) out_ref = ^seq_item_sb.B ;
40         else out_ref = seq_item_sb.A | seq_item_sb.B ;
41     end
42     else if (seq_item_sb.opcode == 1) begin
43         if (seq_item_sb.red_op_A) out_ref = ^seq_item_sb.A ;
44         else if (seq_item_sb.red_op_B) out_ref = ^seq_item_sb.B ;
45         else out_ref = seq_item_sb.A ^ seq_item_sb.B ;
46     end
47     else if (seq_item_sb.opcode == 2) out_ref = seq_item_sb.A + seq_item_sb.B + seq_item_sb.cin ;
48     else if (seq_item_sb.opcode == 3) out_ref = seq_item_sb.A * seq_item_sb.B ;
49
50     if (seq_item_sb.rst) leds_ref = 0 ;
51     begin
52         if (invalid)
53             leds_ref = ~leds_ref;
54     end
55 endtask
56 endclass
57
58 endpackage
```

16. ALSU_coverage:

```
 1 package ALSU_coverage_pkg;
 2   import uvm_pkg::*;
 3   import ALSU_item_pkg::*;
 4   `include "uvm_macros.svh"
 5
 6   class ALSU_coverage extends uvm_component;
 7     `uvm_component_utils(ALSU_coverage)
 8     uvm_analysis_export #(ALSU_item) cov_export;
 9     uvm_tlm_analysis_fifo #(ALSU_item) cov_fifo;
10     ALSU_item seq_item_cov;
11
12   covergroup covcode;
13     A_cp : coverpoint seq_item_cov.A {
14       bins A_data_0          = { 0 }      ;
15       bins A_data_max        = {MAXPOS}  ;
16       bins A_data_min        = {MAXNEG}  ;
17       bins A_data_default    = default   ;
18       bins A_data_walkingones[] = {001,010,100} iff (seq_item_cov.red_op_A) ;
19     }
20     B_cp : coverpoint seq_item_cov.B {
21       bins B_data_0          = { 0 }      ;
22       bins B_data_max        = {MAXPOS}  ;
23       bins B_data_min        = {MAXNEG}  ;
24       bins B_data_default    = default   ;
25       bins B_data_walkingones[] = {001,010,100} iff (seq_item_cov.red_op_B) ;
26     }
27     ALU_CP : coverpoint seq_item_cov.opcode {
28       bins Bins_shift[]      = {SHIFT , ROTATE} ;
29       bins Bins_arith[]      = {ADD , MULT} ;
30       bins Bins_bitwise[]    = {OR , XOR} ;
31       bins Bins_invalid      = {INVALID_6 , INVALID_7} ;
32       bins trans   = (OR => XOR => ADD => MULT => SHIFT => ROTATE) ;
33     }
34     arith_A: cross A_cp , ALU_CP {
35       option.cross_auto_bin_max = 0 ;
36       bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_0) ;
37       bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_max) ;
38       bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_min) ;
39     }
40     arith_B: cross B_cp , ALU CP {
41       option.cross_auto_bin_max = 0 ;
42       bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_0) ;
43       bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_max) ;
44       bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_min) ;
45     }
46     cin_cp: coverpoint seq_item_cov.cin iff (seq_item_cov.opcode == ADD) {
47       bins zero_cin = {0} ;
48       bins one_cin  = {1} ;
49     }
50     direction_cp : coverpoint seq_item_cov.direction iff (seq_item_cov.opcode inside {SHIFT , ROTATE}) {
51       bins zero_direction = {0} ;
52       bins one_direction  = {1} ;
53     }
54     serial_in_cp : coverpoint seq_item_cov.serial_in iff (seq_item_cov.opcode == SHIFT) {
55       bins zero_serial_in = {0} ;
56       bins one_serial_in  = {1} ;
57     }
58     walking_one_A : cross ALU_CP , A_cp , B_cp {
59       option.cross_auto_bin_max = 0 ;
60       bins A_walkingones = binsof (ALU_CP.Bins_bitwise) && binsof (A_cp.A_data_walkingones) && binsof (B_cp.B_data_0) iff (seq_item_cov.red_op_A) ;
61     }
62     walking_one_B : cross ALU_CP , A_cp , B_cp {
63       option.cross_auto_bin_max = 0 ;
64       bins B_walkingones = binsof (ALU_CP.Bins_bitwise) && binsof (B_cp.B_data_walkingones) && binsof (A_cp.A_data_0) iff (seq_item_cov.red_op_B) ;
65     }
66     invalid_cp : coverpoint seq_item_cov.opcode iff (seq_item_cov.red_op_A || seq_item_cov.red_op_B) {
67       bins invalid_opcode = {ADD , MULT , SHIFT , ROTATE , INVALID_6 , INVALID_7} ;
68     }
69   endgroup

```

```
1      function new (string name = "ALSU_coverage" , uvm_component parent = null);
2          super.new(name , parent) ;
3          covcode = new() ;
4      endfunction
5
6      function void build_phase (uvm_phase phase);
7          super.build_phase(phase);
8          cov_export = new("cov_export",this);
9          cov_fifo   = new("cov_fifo",this);
10         endfunction
11
12         function void connect_phase(uvm_phase phase);
13             super.connect_phase(phase);
14             cov_export.connect(cov_fifo.analysis_export);
15         endfunction
16
17         task run_phase (uvm_phase phase);
18             super.run_phase(phase);
19             forever begin
20                 cov_fifo.get(seq_item_cov);
21                 covcode.sample();
22             end
23         endtask
24     endclass
25 endpackage
```

17.SHIFTER_design:

```
● ● ●  
1 module shift_reg (serial_in, direction, mode, datain, dataout);  
2 input serial_in, direction, mode;  
3 input [5:0] datain;  
4 output reg [5:0] dataout;  
5  
6 always @(*) begin  
7     if (mode) // rotate  
8         if (direction) // left  
9             dataout = {datain[4:0], datain[5]};  
10        else  
11            dataout = {datain[0], datain[5:1]};  
12    else // shift  
13        if (direction) // left  
14            dataout = {datain[4:0], serial_in};  
15        else  
16            dataout = {serial_in, datain[5:1]};  
17 end  
18 endmodule
```

18.SHIFTER_if:

```
● ● ●  
1 interface shift_reg_if ();  
2     logic reset;  
3     logic serial_in, direction, mode;  
4     logic [5:0] datain, dataout;  
5 endinterface
```

19.SHIFTER_test:

```
● ○ ●
1 package shift_reg_test_pkg;
2   import shift_reg_env_pkg::*;
3   import shift_reg_config_pkg::*;
4   import shift_reg_seq_pkg::*;
5   import uvm_pkg::*;
6   `include "uvm_macros.svh"
7
8 class shift_reg_test extends uvm_test;
9   `uvm_component_utils(shift_reg_test)
10  shift_reg_env env ;
11  shift_reg_config shift_reg_cfg ;
12  shift_reg_seq main_seq;
13
14 function new (string name = "shift_reg_test" , uvm_component parent = null);
15   super.new (name , parent) ;
16 endfunction
17
18 function void build_phase (uvm_phase phase);
19   super.build_phase(phase) ;
20   env = shift_reg_env::type_id::create("env",this);
21   shift_reg_cfg = shift_reg_config::type_id::create("shift_reg_cfg");
22   main_seq = shift_reg_seq::type_id::create("main_seq");
23
24 if (!uvm_config_db #(virtual shift_reg_if)::get(this, "" , "shift_reg_if" , shift_reg_cfg.shift_reg_vif))
25   `uvm_fatal("build_phase" , "TEST - unable to get the virtual intrface");
26
27 uvm_config_db #(shift_reg_config)::set(this , "" , "shift_CFG" , shift_reg_cfg);
28
29 endfunction
30
31 task run_phase (uvm_phase phase);
32   super.run_phase(phase);
33   phase.raise_objection(this);
34   `uvm_info("run_phase" , "stimulus generated" , UVM_MEDIUM)
35   main_seq.start(env.agt.sqr);
36   `uvm_info("run_phase" , "stimulus generated" , UVM_MEDIUM)
37   phase.drop_objection(this);
38 endtask: run_phase
39 endclass: shift_reg_test
40 endpackage
```

21 .SHIFTER_configuration:

```
● ○ ●
1 package shift_reg_config_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   class shift_reg_config extends uvm_object;
5     `uvm_object_utils(shift_reg_config)
6     virtual shift_reg_if shift_reg_vif ;
7     uvm_active_passive_enum is_active;
8     function new(string name = "shift_reg_config");
9       super.new(name) ;
10      endfunction
11    endclass
12  endpackage
```

22.SHIFTER_seq_item:

```
● ● ●
1 package shift_reg_item_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   class shift_reg_item extends uvm_sequence_item;
5     `uvm_object_utils(shift_reg_item)
6     rand bit reset ;
7     rand logic serial_in, direction, mode;
8     rand logic [5:0] datain ;
9     logic [5:0] dataout;
10
11    function new(string name = "shift_reg_item");
12      super.new(name);
13    endfunction
14
15    function string convert2string();
16      return $sformatf("%s reset = %b , serial_in = %b , direction = %b , mode = %b , data_in = %b " , super.convert2string() ,
17                        reset , serial_in ,direction ,mode , datain);
18    endfunction
19
20    function string convert2string_stimuls();
21      return $sformatf("reset = %b , serial_in = %b , direction = %b , mode = %b , datain = %b , dataout = %b " ,
22                        reset , serial_in ,direction ,mode , datain , dataout);
23    endfunction
24  endclass
25 endpackage
```

23.SHIFTER_sequence:

```
● ● ●
1 package shift_reg_seq_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import shift_reg_item_pkg::*;
5   class shift_reg_seq extends uvm_sequence #(shift_reg_item);
6     `uvm_object_utils(shift_reg_seq)
7     shift_reg_item seq_item ;
8
9     function new(string name = "shift_reg_seq");
10       super.new(name);
11     endfunction
12
13     task body;
14       seq_item = shift_reg_item ::type_id::create("seq_item");
15       repeat (2000) begin
16         start_item(seq_item);
17         assert(seq_item.randomize);
18         finish_item(seq_item);
19       end
20     endtask
21   endclass
22 endpackage
```

24. SHIFTER_sequencer:

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

25. SHIFTER_env:

```

1 package shift_reg_env_pkg;
2 import shift_reg_scoreboard_pkg::*;
3 import shift_reg_agent_pkg::*;
4 import shift_reg_coverage_pkg::*;
5 import uvm_pkg::*;
6 `include "uvm_macros.svh"
7
8 class shift_reg_env extends uvm_env;
9   `uvm_component_utils(shift_reg_env)
10  shift_reg_agent agt ;
11  shift_reg_coverage cov ;
12  shift_reg_scoreboard sb ;
13
14 function new (string name = "shift_reg_env" , uvm_component parent = null);
15   super.new(name , parent) ;
16 endfunction
17
18 function void build_phase (uvm_phase phase);
19   super.build_phase(phase) ;
20   agt = shift_reg_agent::type_id::create("agt",this);
21   cov = shift_reg_coverage::type_id::create("cov",this);
22   sb = shift_reg_scoreboard::type_id::create("sb",this);
23 endfunction
24
25 function void connect_phase (uvm_phase phase);
26   super.connect_phase(phase) ;
27   agt.agt_ap.connect(sb.sb_export);
28   agt.agt_ap.connect(cov.cov_export);
29 endfunction
30
31 endclass
32 endpackage

```

26.SHIFTER_agent:

```
● ● ●

1 package shift_reg_agent_pkg;
2     import uvm_pkg::*;
3     `include "uvm_macros.svh"
4     import shift_reg_sequencer_pkg::*;
5     import shift_reg_driver_pkg::*;
6     import shift_reg_config_pkg::*;
7     import shift_reg_monitor_pkg::*;
8     import shift_reg_item_pkg::*;
9
10    class shift_reg_agent extends uvm_agent;
11        `uvm_component_utils(shift_reg_agent)
12        shift_reg_sequencer sqr ;
13        shift_reg_driver drv ;
14        shift_reg_monitor mon;
15        shift_reg_config shift_cfg;
16        uvm_analysis_port #(shift_reg_item) agt_ap;
17
18        function new(string name = "shift_reg_agent" , uvm_component parent = null );
19            super.new(name, parent);
20        endfunction
21
22        function void build_phase (uvm_phase phase);
23            super.build_phase(phase);
24            if (!uvm_config_db #(shift_reg_config)::get(this, "", "shift_CFG", shift_cfg)) begin
25                `uvm_fatal("build_phase", "shift_reg_AGENT - unable to get configuration object");
26            end
27            mon = shift_reg_monitor::type_id::create("mon",this);
28            agt_ap = new("agt_ap",this);
29            if (shift_cfg.is_active == UVM_ACTIVE) begin
30                drv = shift_reg_driver::type_id::create("drv",this);
31                sqr = shift_reg_sequencer::type_id::create("sqr",this);
32            end
33        endfunction
34
35        function void connect_phase(uvm_phase phase);
36            super.connect_phase(phase);
37            mon.shift_reg_vif = shift_cfg.shift_reg_vif;
38            mon.mon_ap.connect(agt_ap);
39            if (shift_cfg.is_active == UVM_ACTIVE) begin
40                drv.seq_item_port.connect(sqr.seq_item_export);
41                drv.shift_reg_vif = shift_cfg.shift_reg_vif;
42            end
43        endfunction
44
45    endclass
46 endpackage
```

27. SHIFTER_driver:

```
● ● ●
1 package shift_reg_driver_pkg;
2   import uvm_pkg::*;
3   `include "uvm_macros.svh"
4   import shift_reg_item_pkg::*;
5
6   class shift_reg_driver extends uvm_driver #(shift_reg_item);
7     `uvm_component_utils(shift_reg_driver)
8     virtual shift_reg_if shift_reg_vif;
9     shift_reg_item stim_seq_item;
10
11    function new (string name = "shift_reg_driver", uvm_component parent = null);
12      super.new(name, parent);
13    endfunction
14
15    task run_phase(uvm_phase phase);
16      super.run_phase(phase);
17      forever begin
18        stim_seq_item = shift_reg_item::type_id::create("stim_seq_item");
19        seq_item_port.get_next_item(stim_seq_item);
20        shift_reg_vif.serial_in = stim_seq_item.serial_in;
21        shift_reg_vif.direction = stim_seq_item.direction;
22        shift_reg_vif.mode = stim_seq_item.mode;
23        shift_reg_vif.datain = stim_seq_item.datain;
24        #2;
25        seq_item_port.item_done();
26      end
27    endtask
28  endclass
29 endpackage
```

28. SHIFTER_monitor:

```
1 package shift_reg_monitor_pkg;
2     import uvm_pkg::*;
3     `include "uvm_macros.svh"
4     import shift_reg_item_pkg::*;
5
6     class shift_reg_monitor extends uvm_monitor;
7         `uvm_component_utils(shift_reg_monitor)
8         virtual shift_reg_if shift_reg_vif;
9         shift_reg_item rsp_seq_item;
10        uvm_analysis_port #(shift_reg_item) mon_ap;
11
12        function new (string name = "shift_reg_monitor", uvm_component parent = null);
13            super.new(name, parent);
14        endfunction
15
16        function void build_phase(uvm_phase phase);
17            super.build_phase(phase);
18            mon_ap = new("mon_ap",this);
19        endfunction
20
21        task run_phase(uvm_phase phase);
22            super.run_phase(phase);
23            forever begin
24                rsp_seq_item = shift_reg_item::type_id::create("rsp_seq_item");
25                rsp_seq_item.serial_in = shift_reg_vif.serial_in ;
26                rsp_seq_item.direction = shift_reg_vif.direction ;
27                rsp_seq_item.mode = shift_reg_vif.mode ;
28                rsp_seq_item.datain = shift_reg_vif.datain ;
29                rsp_seq_item.dataout = shift_reg_vif.dataout ;
30                #2;
31                mon_ap.write(rsp_seq_item);
32            end
33        endtask
34    endclass
35 endpackage
```

29. SHIFTER_scoreboard:



```
1 package shift_reg_scoreboard_pkg;
2     import uvm_pkg::*;
3     `include "uvm_macros.svh"
4     import shift_reg_item_pkg::*;
5
6     class shift_reg_scoreboard extends uvm_scoreboard;
7         `uvm_component_utils(shift_reg_scoreboard)
8         uvm_analysis_export #(shift_reg_item) sb_export;
9         uvm_tlm_analysis_fifo #(shift_reg_item) sb_fifo;
10        shift_reg_item seq_item_sb;
11        logic [5:0] dataout_ref;
12
13        int error_count = 0 ;
14        int correct_count = 0 ;
15
16        function new (string name = "shift_reg_scoreboard" , uvm_component parent = null);
17            super.new(name , parent) ;
18        endfunction
19
20        function void build_phase (uvm_phase phase);
21            super.build_phase(phase);
22            sb_export = new("sb_export",this);
23            sb_fifo = new("sb_fifo",this);
24        endfunction
25
26        function void connect_phase(uvm_phase phase);
27            super.connect_phase(phase);
28            sb_export.connect(sb_fifo.analysis_export);
29        endfunction
30
31        function void report_phase(uvm_phase phase);
32            super.report_phase(phase);
33            `uvm_info("SCOREBOARD", $sformatf(
34                "Simulation Summary shifter: correct_count=%
35                , error_count=%
36                , UVM_NONE")
37        endfunction
38
39        task run_phase (uvm_phase phase);
40            super.run_phase(phase);
41            forever begin
42                sb_fifo.get(seq_item_sb);
43                ref_seq_item_chk_model(seq_item_sb);
44                if (seq_item_sb.dataout != dataout_ref) begin
45                    `uvm_error("run_phase" , "comaprison fail");
46                    error_count++;
47                end
48                else correct_count++;
49            end
50        endtask: run_phase
51
52        task ref_seq_item_chk_model(shift_reg_item seq_item_chk);
53            if (seq_item_chk.reset)
54                dataout_ref <= 0;
55            else
56                if (seq_item_chk.mode)
57                    if (seq_item_chk.direction)
58                        dataout_ref = {seq_item_chk.datain[4:0], seq_item_chk.datain[5]};
59                    else
60                        dataout_ref = {seq_item_chk.datain[0], seq_item_chk.datain[5:1]};
61                else
62                    if (seq_item_chk.direction)
63                        dataout_ref = {seq_item_chk.datain[4:0], seq_item_chk.serial_in};
64                    else
65                        dataout_ref = {seq_item_chk.serial_in, seq_item_chk.datain[5:1]};
66            endtask
67
68        endclass
69
70    endpackage
```

30. SHIFTER_coverage:

```
 1 package shift_reg_coverage_pkg;
 2     import uvm_pkg::*;
 3     import shift_reg_item_pkg::*;
 4     `include "uvm_macros.svh"
 5
 6     class shift_reg_coverage extends uvm_component;
 7         `uvm_component_utils(shift_reg_coverage)
 8         uvm_analysis_export #(shift_reg_item) cov_export;
 9         uvm_tlm_analysis_fifo #(shift_reg_item) cov_fifo;
10         shift_reg_item seq_item_cov;
11
12         covergroup covcode;
13             cv1: coverpoint seq_item_cov.serial_in;
14             cv2: coverpoint seq_item_cov.direction;
15             cv3: coverpoint seq_item_cov.mode;
16         endgroup
17
18         function new (string name = "shift_reg_coverage" , uvm_component parent = null);
19             super.new(name , parent) ;
20             covcode = new() ;
21         endfunction
22
23         function void build_phase (uvm_phase phase);
24             super.build_phase(phase);
25             cov_export = new("cov_export",this);
26             cov_fifo   = new("cov_fifo",this);
27         endfunction
28
29         function void connect_phase(uvm_phase phase);
30             super.connect_phase(phase);
31             cov_export.connect(cov_fifo.analysis_export);
32         endfunction
33
34         task run_phase (uvm_phase phase);
35             super.run_phase(phase);
36             forever begin
37                 cov_fifo.get(seq_item_cov);
38                 covcode.sample();
39             end
40         endtask
41     endclass
42 endpackage
```

31.ALSU_assertion:

```
1 import uvm_pkg::*;
2 `include "uvm_macros.svh"
3
4 module ALSU_assertions #(
5   parameter INPUT_PRIORITY = "A"
6 ) (
7   input logic clk,
8   input logic rst,
9   input logic signed [5:0] out,
10  input logic [15:0] leds,
11  input logic [2:0] opcode,
12  input logic bypass_A,
13  input logic bypass_B,
14  input logic red_op_A,
15  input logic red_op_B,
16  input logic direction,
17  input logic signed [2:0] A,
18  input logic signed [2:0] B,
19  input logic [1:0] cin,
20  input logic serial_in
21 );
22
23 // Internal signal to track previous values for shift operations
24 logic signed [5:0] out_prev;
25 always @(posedge clk) begin
26   out_prev <= out;
27 end
28
29 // 1. After reset, outputs must be zero (with proper timing)
30 property reset_clears_outputs;
31   @(posedge clk) rst |=> (out == 0) and (leds == 0);
32 endproperty
33 assert property(reset_clears_outputs)
34   else `uvm_error("ASSERT", "Outputs not cleared after reset");
35
36 // 2. Invalid opcode (110=6, 111=7) -> out must be 0 after 2 cycles
37 property invalid_opcode_out_zero;
38   @(posedge clk) disable iff (rst)
39     (opcode inside {3'b110, 3'b111} && !bypass_A && !bypass_B) |=> ##1 (out == 0);
40 endproperty
41 assert property(invalid_opcode_out_zero)
42   else `uvm_error("ASSERT", "Invalid opcode did not force out=0");
43
44 // 3. Invalid opcode causes LEDs to toggle
45 property invalid_opcode_leds_toggle;
46   @(posedge clk) disable iff (rst)
47     (opcode inside {3'b110, 3'b111}) |=> ##1 (leds == ~$past(leds));
48 endproperty
49 assert property(invalid_opcode_leds_toggle)
50   else `uvm_error("ASSERT", "LEDs did not toggle on invalid opcode");
```

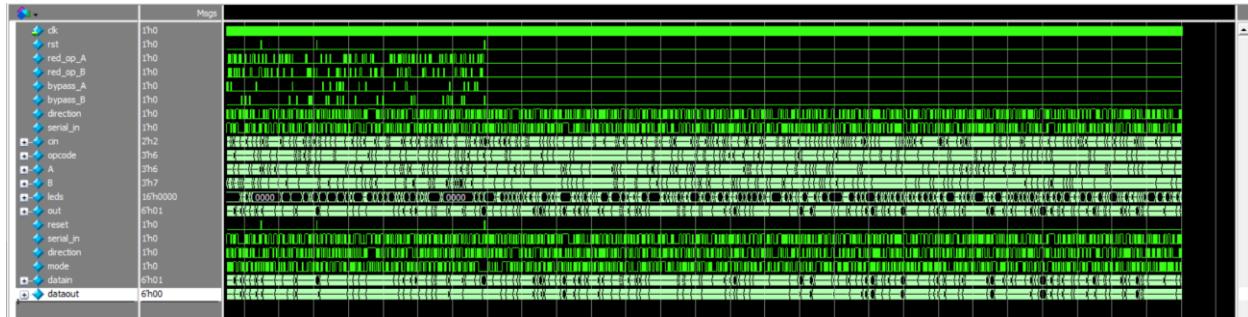
```

1 // 4. Valid opcode -> LEDs must be 0
2 property valid_opcode_leds_zero;
3     @(posedge clk) disable iff (rst)
4         !(opcode inside {3'b110, 3'b111}) &&
5             !((red_op_A | red_op_B) && (opcode[1] | opcode[2]))
6             |=> ###1 (leds == 0);
7 endproperty
8 assert property(valid_opcode_leds_zero)
9     else `uvm_error("ASSERT", "LEDs not zero for valid operation");
10
11 // 5. Bypass_A active (and bypass_B not active) -> out = A after 2 cycles
12 property bypass_A_correct;
13     @(posedge clk) disable iff (rst)
14         (bypass_A && !bypass_B) |-> ##2 (out == $past(A, 2));
15 endproperty
16 assert property(bypass_A_correct)
17     else `uvm_error("ASSERT", "Bypass A mismatch");
18
19 // 6. Bypass_B active (and bypass_A not active) -> out = B after 2 cycles
20 property bypass_B_correct;
21     @(posedge clk) disable iff (rst)
22         (bypass_B && !bypass_A) |-> ##2 (out == $past(B, 2));
23 endproperty
24 assert property(bypass_B_correct)
25     else `uvm_error("ASSERT", "Bypass B mismatch");
26
27 // 7. Both bypass_A and bypass_B active -> priority based on parameter
28 property both_bypass_priority_A;
29     @(posedge clk) disable iff (rst)
30         (bypass_A && bypass_B) |-> ##2 (out == $past(A, 2));
31 endproperty
32 assert property(both_bypass_priority_A)
33     else `uvm_error("ASSERT", "Both bypass active - priority A failed");
34
35 // 8. OR operation (opcode=000) without reduction
36 property or_operation;
37     @(posedge clk) disable iff (rst)
38         (opcode == 3'b000 && !red_op_A && !red_op_B && !bypass_A && !bypass_B)
39         |-> ##2 (out == ($past(A, 2) | $past(B, 2)));
40 endproperty
41 assert property(or_operation)
42     else `uvm_error("ASSERT", "OR operation failed");
43
44 // 9. XOR operation (opcode=001) without reduction
45 property xor_operation;
46     @(posedge clk) disable iff (rst)
47         (opcode == 3'b001 && !red_op_A && !red_op_B && !bypass_A && !bypass_B)
48         |-> ##2 (out == ($past(A, 2) ^ $past(B, 2)));
49 endproperty
50 assert property(xor_operation)
51     else `uvm_error("ASSERT", "XOR operation failed");
52
53 // 10. Reduction OR on A (opcode=000, red_op_A=1, red_op_B=0)
54 property reduction_or_A;
55     @(posedge clk) disable iff (rst)
56         (opcode == 3'b000 && red_op_A && !red_op_B && !bypass_A && !bypass_B)
57         |-> ##2 (out == |$past(A, 2));
58 endproperty
59 assert property(reduction_or_A)
60     else `uvm_error("ASSERT", "Reduction OR A failed");
61
62 // 11. Invalid red_op with non-bitwise opcode -> out = 0
63 property invalid_red_op;
64     @(posedge clk) disable iff (rst)
65         ((red_op_A | red_op_B) && (opcode[1] | opcode[2]) && !bypass_A && !bypass_B)
66         |-> ##2 (out == 0);
67 endproperty
68 assert property(invalid_red_op)
69     else `uvm_error("ASSERT", "Invalid red_op did not force out=0");
70
71
72 endmodule

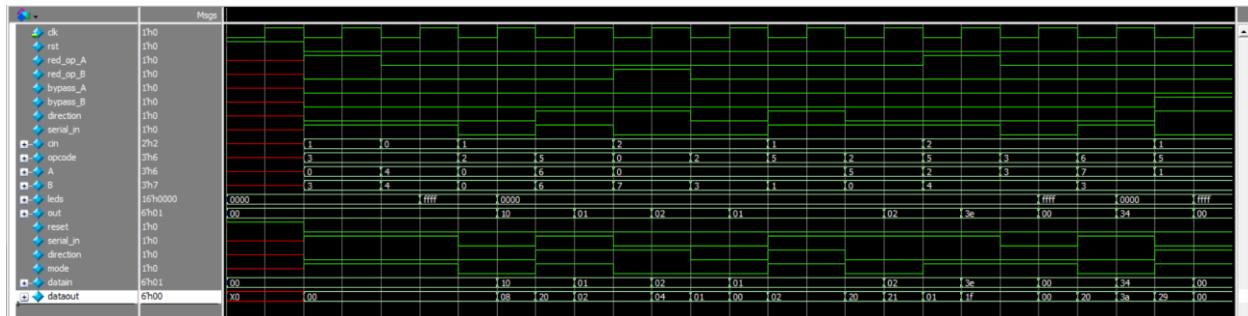
```

32. Questa_sim Waveform:

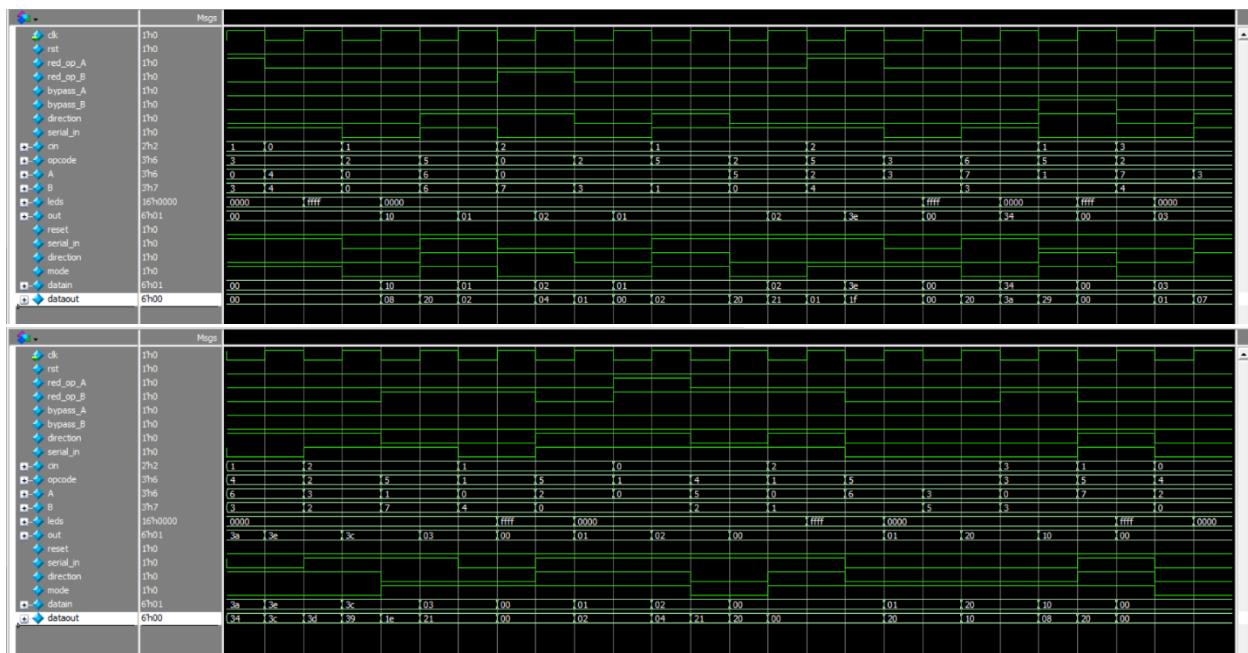
- Waveform showing whole driving data of two interfaces successful:

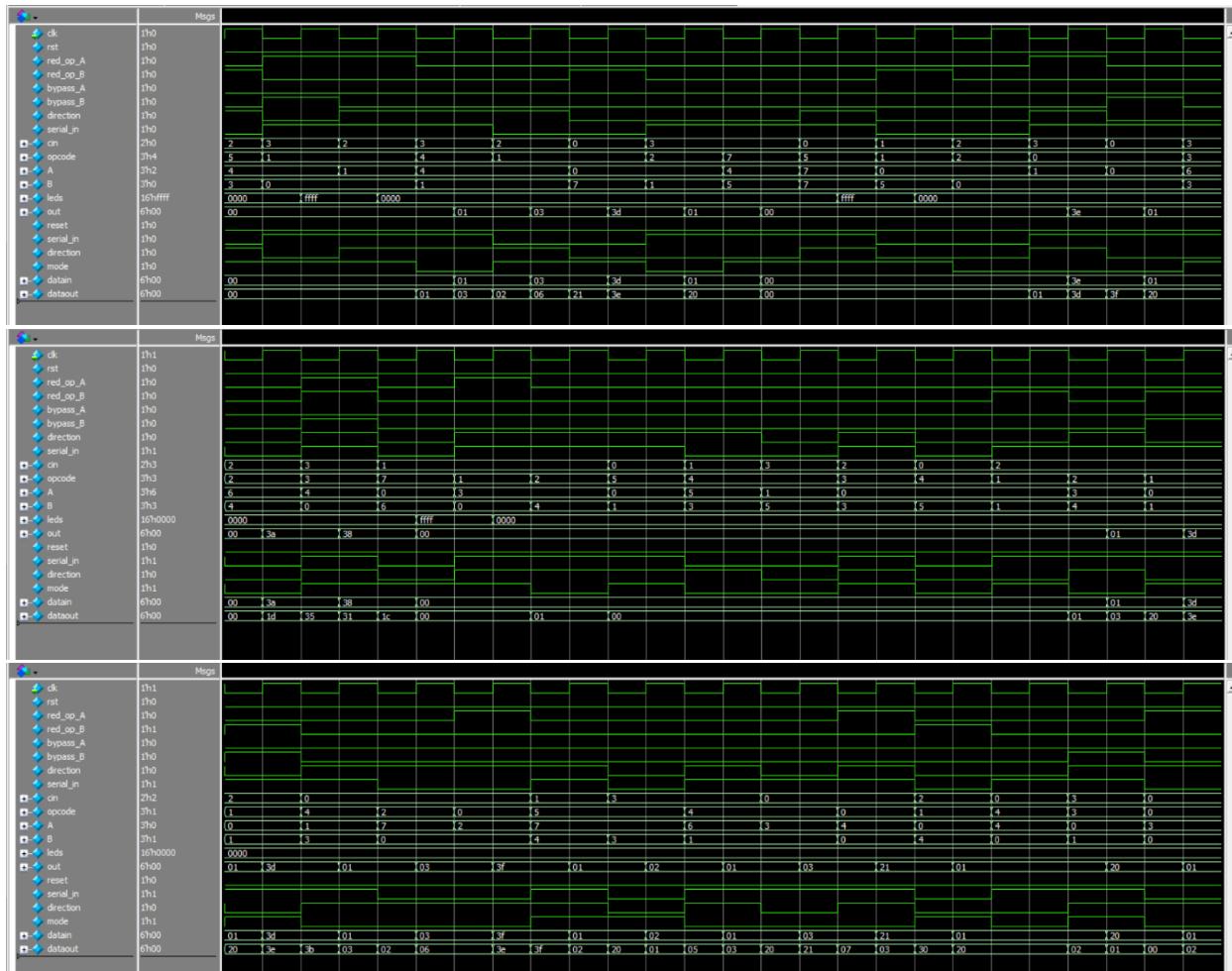


- Waveforms showing the reset behavior:

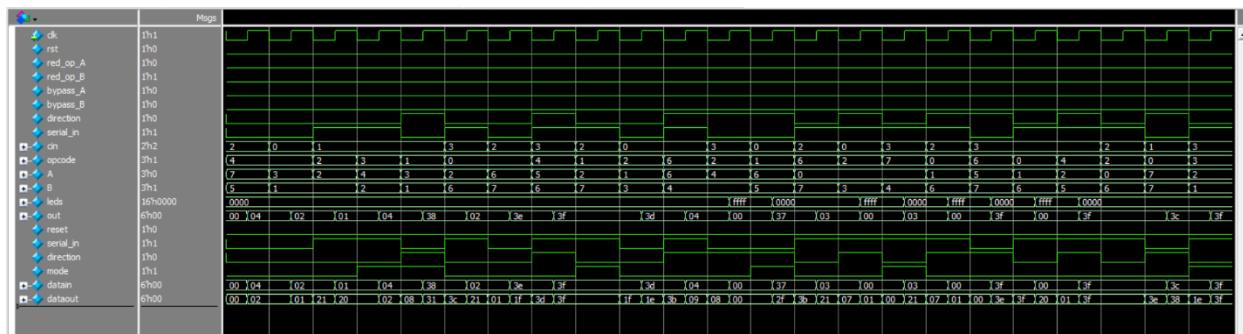


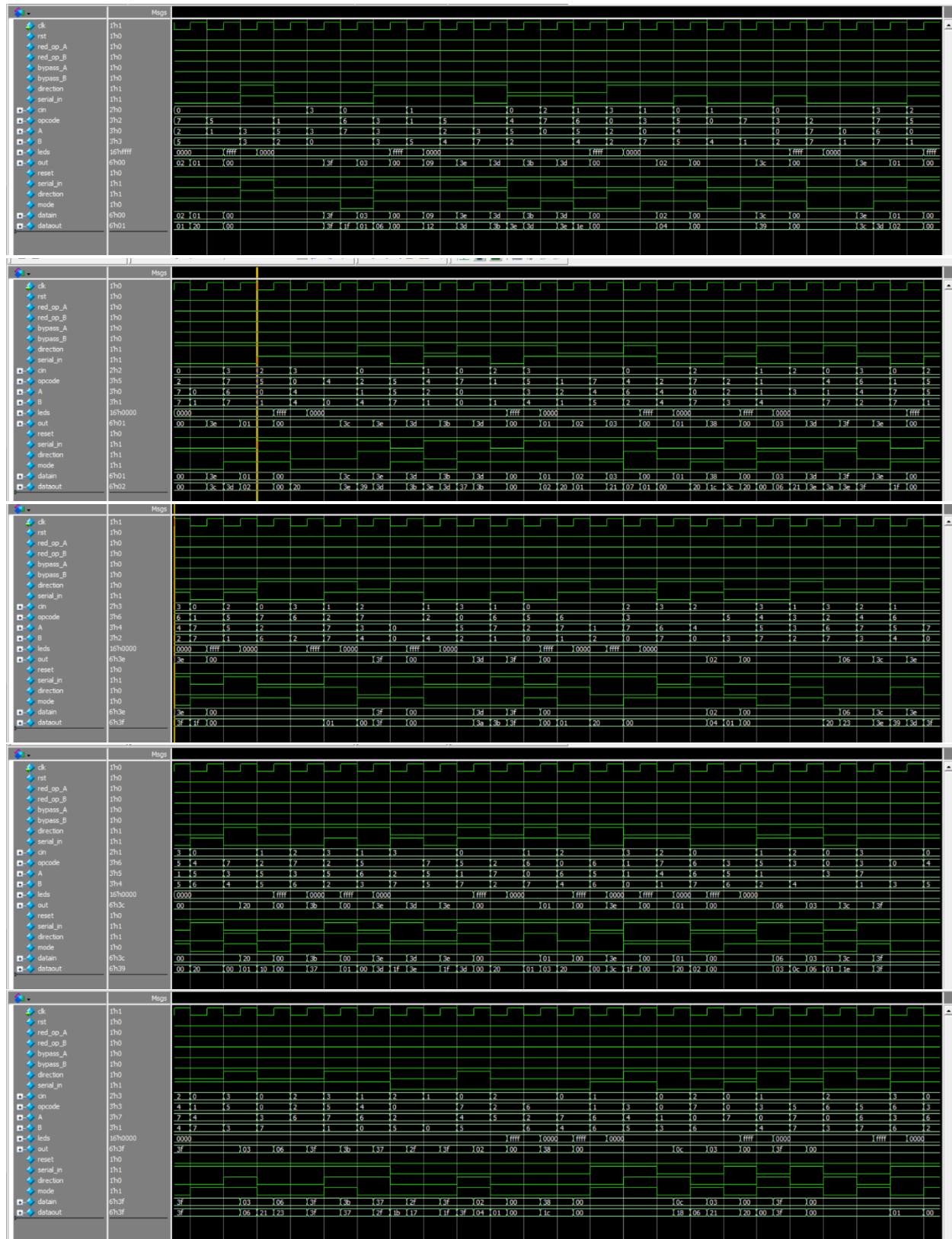
- Waveforms showing the behavior of the first sequence:



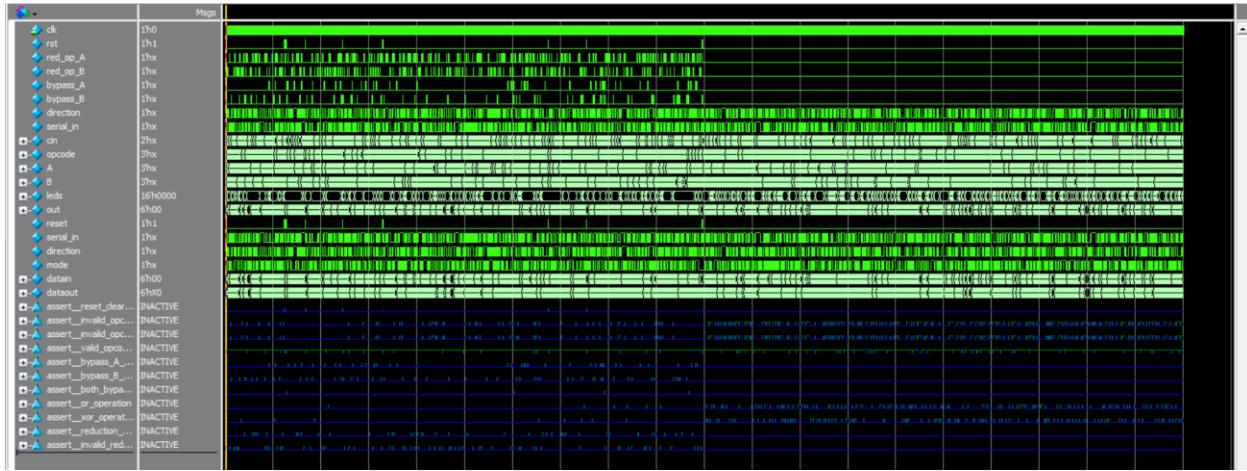


- **Waveforms showing the behavior of the second sequence:**

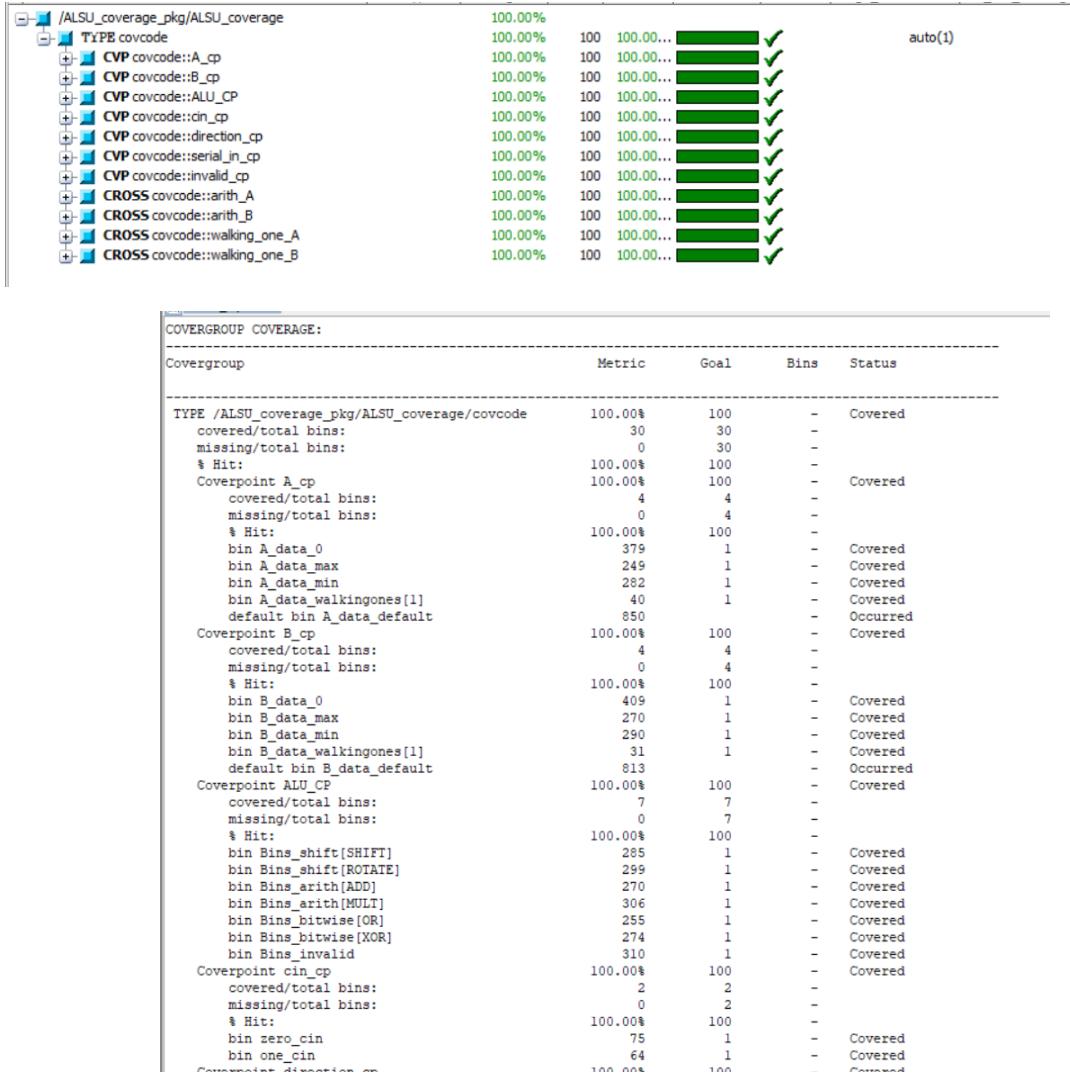




- Waveform showing no assertion error:



33. ALSU Coverage report:



	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
# Hit:	100.00%	100	-	
bin zero_direction	274	1	-	Covered
bin one_direction	310	1	-	Covered
Coverpoint serial_in_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
# Hit:	100.00%	100	-	
bin zero_serial_in	130	1	-	Covered
bin one_serial_in	155	1	-	Covered
Coverpoint invalid_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
# Hit:	100.00%	100	-	
bin invalid_opcode	77	1	-	Covered
Cross arith_A	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
# Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin arith_zero	100	1	-	Covered
bin arith_max	106	1	-	Covered
bin arith_min	111	1	-	Covered
Cross arith_B	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
# Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin arith_zero	113	1	-	Covered
bin arith_max	111	1	-	Covered
bin arith_min	120	1	-	Covered
Cross walking_one_A	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 A_walkingones	34	1	-	Covered
Cross walking_one_B	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 B_walkingones	30	1	-	Covered
TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1				

34.ALSU Functional coverage

```

# =====
# === Instance: /top/inst_DUT/inst_assertion
# === Design Unit: work.ALSU_assertions
# =====
#
# Assertion Coverage:
#   Assertions      11      11      0  100.00%
# -----
#   Name           File(Line)      Failure      Pass
#                  Count        Count
# -----
#   /top/inst_DUT/inst_assertion/assert_invalid_red_op
#       ALSU_assertion.sv(119)      0      1
#   /top/inst_DUT/inst_assertion/assert_reduction_or_A
#       ALSU_assertion.sv(110)      0      1
#   /top/inst_DUT/inst_assertion/assert_xor_operation
#       ALSU_assertion.sv(101)      0      1
#   /top/inst_DUT/inst_assertion/assert_or_operation
#       ALSU_assertion.sv(92)      0      1
#   /top/inst_DUT/inst_assertion/assert_both_bypass_priority_A
#       ALSU_assertion.sv(83)      0      1
#   /top/inst_DUT/inst_assertion/assert_bypass_B_correct
#       ALSU_assertion.sv(75)      0      1
#   /top/inst_DUT/inst_assertion/assert_bypass_A_correct
#       ALSU_assertion.sv(67)      0      1
#   /top/inst_DUT/inst_assertion/assert_valid_opcode_leds_zero
#       ALSU_assertion.sv(59)      0      1
#   /top/inst_DUT/inst_assertion/assert_invalid_opcode_leds_toggle
#       ALSU_assertion.sv(49)      0      1
#   /top/inst_DUT/inst_assertion/assert_invalid_opcode_out_zero
#       ALSU_assertion.sv(41)      0      1
#   /top/inst_DUT/inst_assertion/assert_reset_clears_outputs
#       ALSU_assertion.sv(33)      0      1
#

```

```

# =====
# === Instance: /ALSU_coverage_pkg
# === Design Unit: work.ALSU_coverage_pkg
# =====

#
# Covergroup Coverage:
#   Covergroups          1      na      na  100.00%
#     Coverpoints/Crosses 11      na      na      na
#     Covergroup Bins    30      30      0   100.00%
# -----
#   Covergroup           Metric   Goal   Bins   Status
#   TYPE /ALSU_coverage_pkg/ALSU_coverage/covcode 100.00% 100  -  Covered
#     covered/total bins: 30      30
#     missing/total bins: 0       30
#     % Hit: 100.00% 100
#     Coverpoint A_cp 100.00% 100  -  Covered
#       covered/total bins: 4       4
#       missing/total bins: 0       4
#       % Hit: 100.00% 100
#       bin A_data_0 379      1   -  Covered
#       bin A_data_max 249      1   -  Covered
#       bin A_data_min 282      1   -  Covered
#       bin A_data_walkingones[1] 40      1   -  Covered
#       default bin A_data_default 850      -   Occurred
#     Coverpoint B_cp 100.00% 100  -  Covered
#       covered/total bins: 4       4
#       missing/total bins: 0       4
#       % Hit: 100.00% 100
#       bin B_data_0 409      1   -  Covered
#       bin B_data_max 270      1   -  Covered
#       bin B_data_min 290      1   -  Covered
#       bin B_data_walkingones[1] 31      1   -  Covered
#       default bin B_data_default 813      -   Occurred
#     Coverpoint ALU_CP 100.00% 100  -  Covered
#       covered/total bins: 7       7
#       missing/total bins: 0       7
#       % Hit: 100.00% 100
#       bin Bins_shift[SHIFT] 285      1   -  Covered
#       bin Bins_shift[ROTATE] 299      1   -  Covered
#       bin Bins_arith[ADD] 270      1   -  Covered
#       bin Bins_arith[MULT] 306      1   -  Covered
#       bin Bins_bitwise[OR] 255      1   -  Covered
#       bin Bins_bitwise[XOR] 274      1   -  Covered
#       bin Bins_invalid 310      1   -  Covered
# =====
# === Instance: /ALSU_seq_2_pkg
# === Design Unit: work.ALSU_seq_2_pkg
# =====

#
# Assertion Coverage:
#   Assertions          1      1      0  100.00%
# -----
#   Name        File(Line)      Failure      Pass
#              Count        Count
#   -----
#   /ALSU_seq_2_pkg/ALSU_seq_2/body/#ublk#145558711#17/immed_19
#             ALSU_seq2.sv(19)      0          1
#   -

#
# === Instance: /ALSU_seq_1_pkg
# === Design Unit: work.ALSU_seq_1_pkg
# =====

#
# Assertion Coverage:
#   Assertions          1      1      0  100.00%
# -----
#   Name        File(Line)      Failure      Pass
#              Count        Count
#   -----
#   /ALSU_seq_1_pkg/ALSU_seq_1/body/#ublk#145624247#16/immed_18
#             ALSU_seq1.sv(18)      0          1
#   -

```

#	Coverpoint cin_cp	100.00%	100	-	Covered
#	covered/total bins:	2	2	-	
#	missing/total bins:	0	2	-	
#	% Hit:	100.00%	100	-	
#	bin zero_cin	75	1	-	Covered
#	bin one_cin	64	1	-	Covered
#	Coverpoint direction_cp	100.00%	100	-	Covered
#	covered/total bins:	2	2	-	
#	missing/total bins:	0	2	-	
#	% Hit:	100.00%	100	-	
#	bin zero_direction	274	1	-	Covered
#	bin one_direction	310	1	-	Covered
#	Coverpoint serial_in_cp	100.00%	100	-	Covered
#	covered/total bins:	2	2	-	
#	missing/total bins:	0	2	-	
#	% Hit:	100.00%	100	-	
#	bin zero_serial_in	130	1	-	Covered
#	bin one_serial_in	155	1	-	Covered
#	Coverpoint invalid_cp	100.00%	100	-	Covered
#	covered/total bins:	1	1	-	
#	missing/total bins:	0	1	-	
#	% Hit:	100.00%	100	-	
#	bin invalid_opcode	77	1	-	Covered
#	Cross arith_A	100.00%	100	-	Covered
#	covered/total bins:	3	3	-	
#	missing/total bins:	0	3	-	
#	% Hit:	100.00%	100	-	
#	Auto, Default and User Defined Bins:				
#	bin arith_zero	100	1	-	Covered
#	bin arith_max	106	1	-	Covered
#	bin arith_min	111	1	-	Covered
#	Cross arith_B	100.00%	100	-	Covered
#	covered/total bins:	3	3	-	
#	missing/total bins:	0	3	-	
#	% Hit:	100.00%	100	-	
#	Auto, Default and User Defined Bins:				
#	bin A_walkingones	113	1	-	Covered
#	bin B_walkingones	111	1	-	Covered
#	bin C_walkingones	120	1	-	Covered
#	Cross walking_one_A	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 A_walkingones	34	1	-	Covered
#	Cross walking_one_B	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 B_walkingones	30	1	-	Covered
#					
#	ASSERTION RESULTS:				
#	-----				
#	Name	File(Line)	Failure Count	Pass Count	
#					
#	-----				
#	/top/inst_DUT/inst_assertion/assert_invalid_red_op	ALSU_assertion.sv(119)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_reduction_or_A	ALSU_assertion.sv(110)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_xor_operation	ALSU_assertion.sv(101)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_or_operation	ALSU_assertion.sv(92)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_both_bypass_priority_A	ALSU_assertion.sv(83)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_bypass_B_correct	ALSU_assertion.sv(75)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_bypass_A_correct	ALSU_assertion.sv(67)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_valid_opcode_leds_zero	ALSU_assertion.sv(59)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_invalid_opcode_leds_toggle	ALSU_assertion.sv(49)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_invalid_opcode_out_zero	ALSU_assertion.sv(41)	0	1	
#					
#	/top/inst_DUT/inst_assertion/assert_reset_clears_outputs	ALSU_assertion.sv(33)	0	1	
#					
#	/ALSU_seq_2_pkg/ALSU_seq_2/body/#ublk#145558711#17/immed_19	ALSU_seq2.sv(19)	0	1	
#					
#	/ALSU_seq_1_pkg/ALSU_seq_1/body/#ublk#145624247#16/immed_18	ALSU_seq1.sv(18)	0	1	
#					
#	Total Coverage By Instance (filtered view): 100.00%				
#					
#	End time: 14:47:34 on Oct 02, 2025, Elapsed time: 0:00:00				

35.SHIFTER_coverage report:

	/shift_reg_coverage_pkg/shift_reg_coverage		100.00%		
	TYPE covcode		100.00%	100	100.00...
	CVP covcode::cv1		100.00%	100	100.00...
	CVP covcode::cv2		100.00%	100	100.00...
	CVP covcode::cv3		100.00%	100	100.00...
<hr/>					
=====					
== Instance: /shift_reg_coverage_pkg					
== Design Unit: work.shift_reg_coverage_pkg					
<hr/>					
Covergroup Coverage:					
Covergroups 1 na na 100.00%					
Coverpoints/Crosses 3 na na na					
Covergroup Bins 6 6 0 100.00%					
<hr/>					
Covergroup					
Metric Goal Bins Status					
<hr/>					
TYPE /shift_reg_coverage_pkg/shift_reg_coverage/covcode					
covered/total bins: 100.00% 100 - Covered					
covered/total bins: 6 6 -					
missing/total bins: 0 0 -					
% Hit: 100.00% 100 -					
Coverpoint cv1 100.00% 100 - Covered					
covered/total bins: 2 2 -					
missing/total bins: 0 2 -					
% Hit: 100.00% 100 -					
bin auto[0] 966 1 - Covered					
bin auto[1] 1033 1 - Covered					
Coverpoint cv2 100.00% 100 - Covered					
covered/total bins: 2 2 -					
missing/total bins: 0 2 -					
% Hit: 100.00% 100 -					
bin auto[0] 971 1 - Covered					
bin auto[1] 1028 1 - Covered					
Coverpoint cv3 100.00% 100 - Covered					
covered/total bins: 2 2 -					
missing/total bins: 0 2 -					
% Hit: 100.00% 100 -					
bin auto[0] 961 1 - Covered					
bin auto[1] 1038 1 - Covered					
<hr/>					
=====					
== Instance: /ALSU_coverage_pkg					
== Design Unit: work.ALSU_coverage_pkg					
<hr/>					
Covergroup Coverage:					
Covergroups 1 na na 100.00%					
Coverpoints/Crosses 11 na na na					
Covergroup Bins 30 30 0 100.00%					
<hr/>					
Covergroup					
Metric Goal Bins Status					
<hr/>					
TYPE /ALSU_coverage_pkg/ALSU_coverage/covcode					
covered/total bins: 100.00% 100 - Covered					
covered/total bins: 30 30 -					
missing/total bins: 0 30 -					
% Hit: 100.00% 100 -					
Coverpoint A_cp 100.00% 100 - Covered					
covered/total bins: 4 4 -					
missing/total bins: 0 4 -					
% Hit: 100.00% 100 -					
bin A_data_0 379 1 - Covered					
bin A_data_max 249 1 - Covered					
bin A_data_min 282 1 - Covered					
bin A_data_walkingones[1] 40 1 - Covered					
default bin A_data_default 850 - Occurred					
Coverpoint B_cp 100.00% 100 - Covered					
covered/total bins: 4 4 -					
missing/total bins: 0 4 -					
% Hit: 100.00% 100 -					
bin B_data_0 409 1 - Covered					
bin B_data_max 270 1 - Covered					
bin B_data_min 290 1 - Covered					
bin B_data_walkingones[1] 31 1 - Covered					
default bin B_data_default 813 - Occurred					
Coverpoint ALU_CP 100.00% 100 - Covered					
covered/total bins: 7 7 -					
missing/total bins: 0 7 -					
% Hit: 100.00% 100 -					
bin Bins_shift[SHIFT] 285 1 - Covered					
bin Bins_shift[ROTATE] 299 1 - Covered					
bin Bins_arith[ADD] 270 1 - Covered					
bin Bins_arith[MULTI] 306 1 - Covered					
bin Bins_bitwise[OR] 255 1 - Covered					
bin Bins_bitwise[XOR] 274 1 - Covered					
bin Bins_invalid 310 1 - Covered					
Coverpoint cin_cp 100.00% 100 - Covered					

Coverpoint cin_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero_cin	75	1	-	Covered
bin one_cin	64	1	-	Covered
Coverpoint direction_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero_direction	274	1	-	Covered
bin one_direction	310	1	-	Covered
Coverpoint serial_in_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero_serial_in	130	1	-	Covered
bin one_serial_in	155	1	-	Covered
Coverpoint invalid_cp	100.00%	100	-	Covered
covered/total bins:	1	1	-	
missing/total bins:	0	1	-	
% Hit:	100.00%	100	-	
bin invalid_opcode	77	1	-	Covered
Cross arith_A	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin arith_zero	100	1	-	Covered
bin arith_max	106	1	-	Covered
bin arith_min	111	1	-	Covered
Cross arith_B	100.00%	100	-	Covered
covered/total bins:	3	3	-	
missing/total bins:	0	3	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin arith_zero	113	1	-	Covered
bin arith_max	111	1	-	Covered
bin arith_min	120	1	-	Covered
Cross walking_one_A	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 A_walkingones	34	1	-	Covered
Cross walking_one_B	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 B_walkingones	30	1	-	Covered
TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 2				
Total Coverage By Instance (filtered view): 100.00%				

36.SHIFTER functional coverage :

# COVERGROUP COVERAGE:				
# -----				
# Covergroup		Metric	Goal	Bins
#				Status
# -----				
# TYPE /shift_reg_coverage_pkg/shift_reg_coverage/covcode	100.00%	100	-	Covered
# covered/total bins:	6	6	-	
# missing/total bins:	0	6	-	
# % Hit:	100.00%	100	-	
# Coverpoint cv1	100.00%	100	-	Covered
# covered/total bins:	2	2	-	
# missing/total bins:	0	2	-	
# % Hit:	100.00%	100	-	
# bin auto[0]	966	1	-	Covered
# bin auto[1]	1033	1	-	Covered
# Coverpoint cv2	100.00%	100	-	Covered
# covered/total bins:	2	2	-	
# missing/total bins:	0	2	-	
# % Hit:	100.00%	100	-	

```

# COVERGROUP COVERAGE:
#
#-----#
# Covergroup
#-----#
#   TYPE /shift_reg_coverage_pkg/shift_reg_coverage/covcode
#           Metric      Goal     Bins    Status
#           100.00%    100      -       Covered
#   covered/total bins:          6       6      -
#   missing/total bins:         0       6      -
#   % Hit:                      100.00% 100      -
#   Coverpoint cv1
#       covered/total bins:      100.00% 100      -       Covered
#       missing/total bins:      2       2      -
#       % Hit:                  100.00% 100      -
#       bin auto[0]              966      1      -       Covered
#       bin auto[1]              1033     1      -       Covered
#   Coverpoint cv2
#       covered/total bins:      100.00% 100      -       Covered
#       missing/total bins:      2       2      -
#       % Hit:                  100.00% 100      -
#       bin auto[0]              971      1      -       Covered
#       bin auto[1]              1028     1      -       Covered
#   Coverpoint cv3
#       covered/total bins:      100.00% 100      -       Covered
#       missing/total bins:      2       2      -
#       % Hit:                  100.00% 100      -
#       bin auto[0]              961      1      -       Covered
#       bin auto[1]              1038     1      -       Covered
#-----#
# Name             File(Line)        Failure  Pass
#                   Count        Count
# -----#
# /top/ALSU_inst_DUT/inst_assertion/assert_invalid_red_op
#                         ALSU/ALSU_assertion.sv(115)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_reduction_or_A
#                         ALSU/ALSU_assertion.sv(106)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_xor_operation
#                         ALSU/ALSU_assertion.sv(97)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_or_operation
#                         ALSU/ALSU_assertion.sv(88)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_both_bypass_priority_A
#                         ALSU/ALSU_assertion.sv(79)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_bypass_B_correct
#                         ALSU/ALSU_assertion.sv(71)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_bypass_A_correct
#                         ALSU/ALSU_assertion.sv(63)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_valid_opcode_leds_zero
#                         ALSU/ALSU_assertion.sv(55)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_invalid_opcode_leds_toggle
#                         ALSU/ALSU_assertion.sv(45)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_invalid_opcode_out_zero
#                         ALSU/ALSU_assertion.sv(37)
#                               0       1
# /top/ALSU_inst_DUT/inst_assertion/assert_reset_clears_outputs
#                         ALSU/ALSU_assertion.sv(29)
#                               0       1
# /ALSU_seq_2_pkg/ALSU_seq_2/body/#ublk#145558711#17/immed_19
#                         ALSU/ALSU_seq2.sv(19) 0       1
# /ALSU_seq_1_pkg/ALSU_seq_1/body/#ublk#145624247#16/immed_18
#                         ALSU/ALSU_seq1.sv(18) 0       1
#
# Total Coverage By Instance (filtered view): 100.00%
#
# End time: 19:04:09 on Oct 03,2025, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0

```

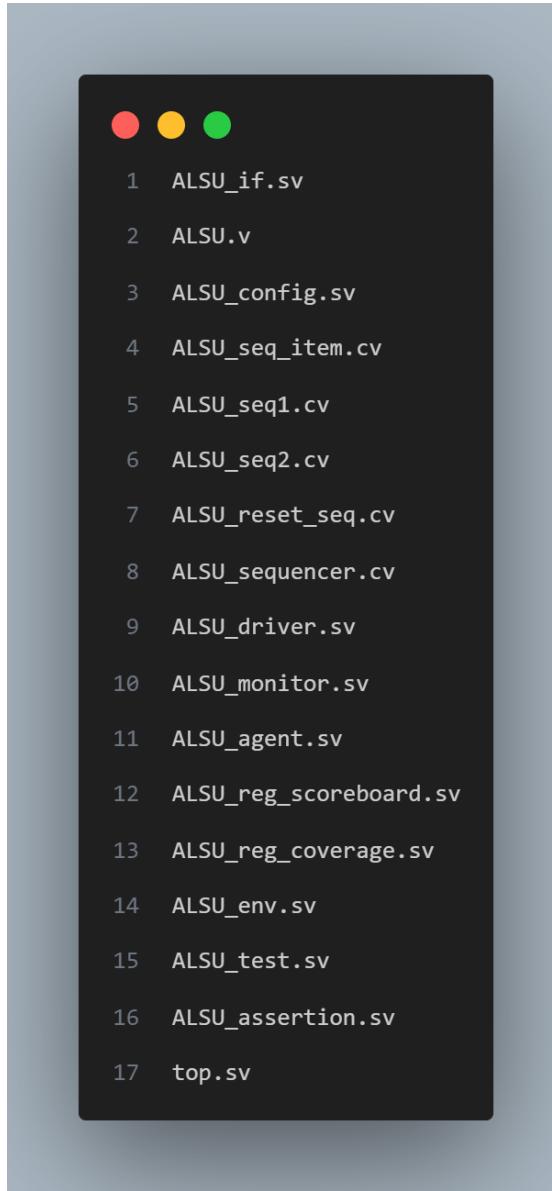
37.DO file:

```
● ● ●  
1 vlib work  
2 vlog -f SHIFTER/src_files_shifter.list  
3 vlog -f ALSU/src_files_ALSU.list  
4 vsim -voptargs+=acc work.top -classdebug -uvmcontrol=all  
5 add wave /top/inst_if/*  
6 coverage save ALSU.ucdb -onexit  
7 coverage save shift_reg.ucdb -onexit  
8 run -all
```

38 .SHIFTER source file:

```
● ● ●  
1 shift_reg_if.sv  
2 shift_reg.v  
3 shift_reg_config.sv  
4 shift_reg_seq_item.cv  
5 shift_reg_seq.cv  
6 shift_reg_reset_seq.cv  
7 shift_reg_sequencer.cv  
8 shift_reg_driver.sv  
9 shift_reg_monitor.sv  
10 shift_reg_agent.sv  
11 shitr_reg_scoreboard.sv  
12 shitr_reg_coverage.sv  
13 shift_reg_env.sv  
14 shift_reg_test.sv  
15 SHIFTER_top.sv
```

39. ALSU source file:



40. UVM report showing no errors :

```
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(53) @ 0: uvm_test_top [run_phase] reset asserted
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(53) @ 2: uvm_test_top [run_phase] reset deasserted
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(53) @ 2002: uvm_test_top [run_phase] seq_1 stimulus generated started
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(53) @ 2002: uvm_test_top [run_phase] seq_1 stimulus generated ended
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(61) @ 2002: uvm_test_top [run_phase] seq_2 stimulus generated started
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_test.sv(61) @ 4002: uvm_test_top [run_phase] seq_2 stimulus generated ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(126) @ 4002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/ALSU/ALSU_scoreboard.sv(38) @ 4002: uvm_test_top.env.ab [SCOREBOARD] Simulation Summary ALSU : correct_count=2000, error_count=0
# UVM_INFO E:/Digital_course/verification/session6/ASSIGNMENT2/SHIFTER/shifter_reg_scoreboard.sv(33) @ 4002: uvm_test_top.env_shift.ab [SCOREBOARD] Simulation Summary shifter: correct_count=2001, error_count=0
# ---- UVM Report Summary ---
# ** Report counts by severity
# UVM_INFO : 12
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# Report counts by id
# [Objects UVM] 2
# [RUNIT] 1
# [SCOREBOARD] 2
# [TEST_DONE] 1
# [run_phase] 6
# ** Note: $finish : C:/questasim64_2021.1/win64/.../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 4002 ns Iteration: 61 Instance: /top
# 1
```