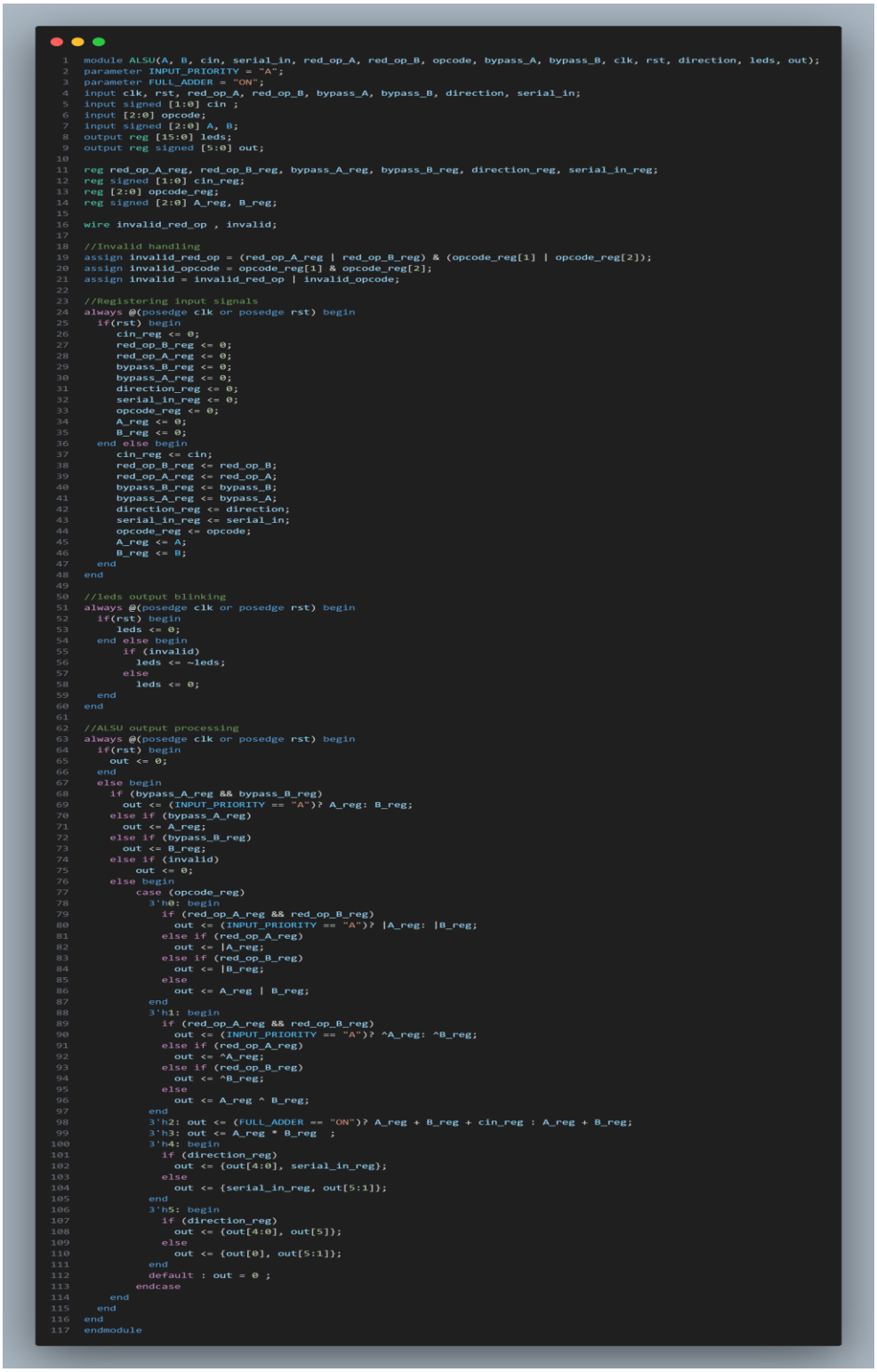
ALSU enverionment

1. Design



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

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

wire invalid_red_op , invalid;

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

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

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

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

2. Top :

```
import uvm_pkg::*;
`include "uvm_macros.svh"
import ALSU_test_pkg::*;

module top();
  bit clk ,reset ;
  initial begin
    forever
      #1 clk =~clk ;
  end
  ALSU_if inst_if (clk) ;
  ALSU inst_DUT (inst_if.A, inst_if.B, inst_if.cin, inst_if.serial_in,
                 inst_if.red_op_A, inst_if.red_op_B, inst_if.opcode,
                   inst_if.bypass_A, inst_if.bypass_B, inst_if.clk,
                   inst_if.rst, inst_if.direction, inst_if.leds, inst_if.out);
  bind inst_DUT ALSU_assertions inst_assertion  (
    .clk(inst_if.clk),
    .rst(inst_if.rst),
    .out(inst_if.out),
    .leds(inst_if.leds),
    .opcode(inst_if.opcode),
    .bypass_A(inst_if.bypass_A),
    .bypass_B(inst_if.bypass_B),
    .red_op_A(inst_if.red_op_A),
    .red_op_B(inst_if.red_op_B),
    .direction(inst_if.direction),
    .A(inst_if.A),
    .B(inst_if.B),
    .cin(inst_if.cin),
    .serial_in(inst_if.serial_in)
  );
  initial begin
    uvm_config_db #(virtual ALSU_if)::set(null, "uvm_test_top" , "ALSU_if" ,inst_if) ;
    run_test("ALSU_test");
  end
endmodule
```

3. ALSU_if:

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

4. ALSU_test:

```
package ALSU_test_pkg;
  import ALSU_env_pkg::*;
  import ALSU_config_pkg::*;
  import ALSU_reset_seq_pkg::*;
  import ALSU_seq_1_pkg::*;
  import ALSU_seq_2_pkg::*;
  import uvm_pkg::*;
  `include "uvm_macros.svh"

  class ALSU_test extends uvm_test;
    `uvm_component_utils(ALSU_test)
    ALSU_env env ;
    ALSU_config ALSU_cfg ;
    ALSU_seq_1 seq_1;
    ALSU_seq_2 seq_2;
    ALSU_seq_reset rst_seq;

      function new (string name = "ALSU_test" , uvm_component parent = null);
        super.new (name , parent) ;
      endfunction

    function void build_phase (uvm_phase phase);
      super.build_phase(phase) ;
      env = ALSU_env::type_id::create("env",this);
      ALSU_cfg = ALSU_config::type_id::create("ALSU_cfg");
      seq_1 = ALSU_seq_1::type_id::create("seq_1");
      seq_2 = ALSU_seq_2::type_id::create("seq_2");
      rst_seq = ALSU_seq_reset::type_id::create("rst_seq");

      if (!uvm_config_db #(virtual ALSU_if)::get(this, "" , "ALSU_if" , ALSU_cfg.ALSU_vif))
        `uvm_fatal("build_phase" , "TEST - unable to get the virtual intrface");

      uvm_config_db #(ALSU_config)::set(this , "*" , "CFG" , ALSU_cfg);

    endfunction

    task run_phase (uvm_phase phase);
      super.run_phase(phase);
      phase.raise_objection(this);
      `uvm_info("run_phase", "reset asserted" , UVM_MEDIUM)
      rst_seq.start(env.agt.sqr);
      `uvm_info("run_phase", "reset deasserted" , UVM_MEDIUM)

      `uvm_info("run_phase", "seq_1 stimulis generated started " , UVM_MEDIUM)
      seq_1.start(env.agt.sqr);
      `uvm_info("run_phase", "seq_1 stimulis generated ended" , UVM_MEDIUM)

      `uvm_info("run_phase", "seq_2 stimulis generated started" , UVM_MEDIUM)
      seq_2.start(env.agt.sqr);
      `uvm_info("run_phase", "seq_2 stimulis generated ended" , UVM_MEDIUM)
      phase.drop_objection(this);
    endtask: run_phase
  endclass

endpackage
```

5. ALSU_configuration:

```
package ALSU_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class ALSU_config extends uvm_object;
        `uvm_object_utils(ALSU_config)
        virtual ALSU_if ALSU_vif ;
        function new(string name = "ALSU_config");
            super.new(name) ;
        endfunction
    endclass
endpackage
```

6. ALSU_seq_item

```
package ALSU_item_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    parameter MAXPOS =  3 ;
    parameter MAXNEG = -4 ;
    typedef enum logic [2:0] {
        OR, XOR, ADD, MULT, SHIFT, ROTATE, INVALID_6, INVALID_7
    } opcode_t ;

    class ALSU_item extends uvm_sequence_item;
        `uvm_object_utils(ALSU_item)
        rand logic rst ;
        rand logic signed [2:0] A ;
        rand logic signed [2:0] B ;
        rand logic signed [1:0] cin ;
        rand logic serial_in ;
        rand logic red_op_A ;
        rand logic red_op_B ;
        rand logic bypass_A ;
        rand logic bypass_B ;
        rand logic direction;
        rand opcode_t opcode ;
        rand opcode_t [6] opcode_arr ;
        logic [15:0] leds;
        logic signed [5:0] out;

        function new(string name = "ALSU_item");
            super.new(name);
        endfunction

        function string convert2string();
            return $sformatf("%s reset = %b , A = %b , B = %b , cin = %b , serial_in = %b  , red_op_A = %b , red_op_B = %b ,
                              bypass_A = %b , bypass_B = %b , direction = %b , opcode = %b " , super.convert2string() ,
                             rst , A ,B ,cin , serial_in , red_op_A , red_op_B , bypass_A , bypass_B , direction , opcode);
        endfunction

        function string convert2string_stimuls();
            return $sformatf("reset = %b , A = %b , B = %b , cin = %b , serial_in = %b  , red_op_A = %b , red_op_B = %b ,
                              bypass_A = %b , bypass_B = %b , direction = %b , opcode = %b , leds = %b , out = %d  " ,
                             rst , A ,B ,cin , serial_in , red_op_A , red_op_B , bypass_A , bypass_B , direction , opcode , leds , out);
        endfunction
```

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

        constraint reduction_A {
            if ( (opcode inside {XOR , OR}) && red_op_A ) {
                A dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
                        3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
                B dist {0:/100} ;
            }
        }

        constraint reduction_B {
            if ( (opcode inside {XOR , OR}) && red_op_B ) {
                B dist {3'b001:/25, 3'b010:/25, 3'b100:/25,
                        3'b000:/5, 3'b011:/5, 3'b101:/5, 3'b110:/5, 3'b111:/5};
                A dist {0:/100} ;
            }
        }

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

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

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

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

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

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

    endclass
endpackage
```

7. ALSU_reset_sequence:

```
package ALSU_reset_seq_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;
    class ALSU_seq_reset extends uvm_sequence #(ALSU_item);
        `uvm_object_utils(ALSU_seq_reset)
        ALSU_item seq_item ;

        function new(string name = "ALSU_seq_reset");
            super.new(name);
        endfunction

        task body;
            seq_item = ALSU_item ::type_id::create("seq_item");
            start_item(seq_item);
            seq_item.rst = 1 ;
            finish_item(seq_item);
        endtask
    endclass
endpackage
```

8. ALSU_sequence1

```
package ALSU_seq_1_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;
    class ALSU_seq_1 extends uvm_sequence #(ALSU_item);
        `uvm_object_utils(ALSU_seq_1)
        ALSU_item seq_item ;

        function new(string name = "ALSU_seq_1");
            super.new(name);
        endfunction

        task body;
            seq_item = ALSU_item ::type_id::create("seq_item");
            seq_item.OPCODE_array.constraint_mode (0) ;
            repeat (1000) begin
            start_item(seq_item);
            assert(seq_item.randomize);
            finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

9. ALSU_sequence2:

```
package ALSU_seq_2_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;
    class ALSU_seq_2 extends uvm_sequence #(ALSU_item);
        `uvm_object_utils(ALSU_seq_2)
        ALSU_item seq_item ;

        function new(string name = "ALSU_seq_2");
            super.new(name);
        endfunction

        task body;
            seq_item = ALSU_item ::type_id::create("seq_item");
            seq_item.constraint_mode (0) ;
            seq_item.OPCODE_array.constraint_mode (1) ;
            repeat (1000) begin
            start_item(seq_item);
            assert(seq_item.randomize() with {
                rst == 0;
                bypass_A == 0;
                bypass_B == 0;
                red_op_A == 0;
                red_op_B == 0;
                });
            finish_item(seq_item);
            end
        endtask
    endclass
endpackage
```

10.ALSU_sequencer

```
package ALSU_sequencer_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;
    class ALSU_sequencer extends uvm_sequencer #(ALSU_item);
        `uvm_component_utils(ALSU_sequencer)

        function new(string name = "ALSU_sequencer" , uvm_component parent = null );
            super.new(name, parent);
        endfunction
    endclass
endpackage
```

11. ALSU_env:

```
package ALSU_env_pkg;
  import ALSU_scoreboard_pkg::*;
  import ALSU_agent_pkg::*;
  import ALSU_coverage_pkg::*;
  import uvm_pkg::*;
  `include "uvm_macros.svh"

  class ALSU_env extends uvm_env;

    `uvm_component_utils(ALSU_env)
    ALSU_agent agt ;
    ALSU_coverage cov ;
    ALSU_scoreboard sb ;

    function new (string name = "ALSU_env" , uvm_component parent = null);
      super.new(name , parent) ;
    endfunction

    function void build_phase (uvm_phase phase);
      super.build_phase(phase) ;
      agt = ALSU_agent::type_id::create("agt",this);
      cov = ALSU_coverage::type_id::create("cov",this);
      sb  = ALSU_scoreboard::type_id::create("sb",this);
    endfunction

    function void connect_phase (uvm_phase phase);
      super.connect_phase(phase) ;
      agt.agt_ap.connect(sb.sb_export);
      agt.agt_ap.connect(cov.cov_export);
    endfunction
  endclass
endpackage
```

12. ALSU_agent

```
package ALSU_agent_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_sequencer_pkg::*;
    import ALSU_driver_pkg::*;
    import ALSU_config_pkg::*;
    import ALSU_monitor_pkg::*;
    import ALSU_item_pkg::*;

    class ALSU_agent extends uvm_agent;
        `uvm_component_utils(ALSU_agent)
        ALSU_sequencer sqr ;
        ALSU_driver drv ;
        ALSU_monitor mon;
        ALSU_config ALSU_cfg;
        uvm_analysis_port #(ALSU_item) agt_ap;

        function new(string name = "ALSU_agent" , uvm_component parent = null );
            super.new(name, parent);
        endfunction

        function void build_phase (uvm_phase phase);
            super.build_phase(phase);
            if (!uvm_config_db #(ALSU_config)::get(this, "", "CFG", ALSU_cfg)) begin
                `uvm_fatal("build_phase", "AGENT - unable to get configuration object");
            end
            sqr = ALSU_sequencer::type_id::create("sqr",this);
            drv = ALSU_driver::type_id::create("drv",this);
            mon = ALSU_monitor::type_id::create("mon",this);
            agt_ap = new("agt_ap",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            drv.ALSU_vif = ALSU_cfg.ALSU_vif;
            mon.ALSU_vif = ALSU_cfg.ALSU_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
            mon.mon_ap.connect(agt_ap);
        endfunction
    endclass
endpackage
```

13. ALSU_driver:

```
package ALSU_driver_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;

    class ALSU_driver extends uvm_driver #(ALSU_item);
        `uvm_component_utils(ALSU_driver)
        virtual ALSU_if ALSU_vif;
        ALSU_item stim_seq_item;

        function new (string name = "ALSU_driver", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                stim_seq_item = ALSU_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item);
                ALSU_vif.rst = stim_seq_item.rst ;
                ALSU_vif.red_op_A = stim_seq_item.red_op_A;
                ALSU_vif.red_op_B = stim_seq_item.red_op_B;
                ALSU_vif.bypass_A = stim_seq_item.bypass_A;
                ALSU_vif.bypass_B = stim_seq_item.bypass_B;
                ALSU_vif.direction = stim_seq_item.direction;
                ALSU_vif.serial_in = stim_seq_item.serial_in;
                ALSU_vif.cin = stim_seq_item.cin;
                ALSU_vif.A = stim_seq_item.A;
                ALSU_vif.B = stim_seq_item.B;
                ALSU_vif.opcode = stim_seq_item.opcode;
                @(negedge ALSU_vif.clk);
                seq_item_port.item_done();
            end
        endtask
    endclass
endpackage
```

14. ALSU_monitor:

```
package ALSU_monitor_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;

    class ALSU_monitor extends uvm_monitor;
        `uvm_component_utils(ALSU_monitor)
        virtual ALSU_if ALSU_vif;
        ALSU_item rsp_seq_item;
        uvm_analysis_port #(ALSU_item) mon_ap;

        function new (string name = "ALSU_monitor", uvm_component parent = null);
            super.new(name, parent);
        endfunction

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            mon_ap = new("mon_ap",this);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                rsp_seq_item = ALSU_item::type_id::create("rsp_seq_item");
                rsp_seq_item.rst = ALSU_vif.rst ;
                rsp_seq_item.serial_in = ALSU_vif.serial_in ;
                rsp_seq_item.direction = ALSU_vif.direction ;
                rsp_seq_item.red_op_A = ALSU_vif.red_op_A ;
                rsp_seq_item.red_op_B = ALSU_vif.red_op_B ;
                rsp_seq_item.bypass_A = ALSU_vif.bypass_A ;
                rsp_seq_item.bypass_B = ALSU_vif.bypass_B ;
                rsp_seq_item.A = ALSU_vif.A ;
                rsp_seq_item.B = ALSU_vif.B ;
                rsp_seq_item.cin = ALSU_vif.cin ;
                rsp_seq_item.opcode = opcode_t' (ALSU_vif.opcode);
                rsp_seq_item.leds = ALSU_vif.leds ;
                rsp_seq_item.out = ALSU_vif.out ;
                @(negedge ALSU_vif.clk);
                mon_ap.write(rsp_seq_item);
            end
        endtask
    endclass
endpackage
```

15. ALSU_scoreboard:

```
package ALSU_scoreboard_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    import ALSU_item_pkg::*;

    class ALSU_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(ALSU_scoreboard)
        uvm_analysis_export #(ALSU_item) sb_export;
        uvm_tlm_analysis_fifo #(ALSU_item) sb_fifo;
        ALSU_item seq_item_sb;
        logic signed [5:0] out_ref;
        logic [15:0] leds_ref ;
        bit invalid ;

        ALSU_item ref_pipe[$];
        ALSU_item tmp;

        int error_count   = 0 ;
        int correct_count = 0 ;

        function new (string name = "ALSU_scoreboard" , uvm_component parent = null);
            super.new(name , parent) ;
        endfunction

        function void build_phase (uvm_phase phase);
            super.build_phase(phase);
            sb_export = new("sb_export",this);
            sb_fifo   = new("sb_fifo",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            sb_export.connect(sb_fifo.analysis_export);
        endfunction

        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("SCOREBOARD", $sformatf(
            "Simulation Summary: correct_count=
, error_countorrect_count, error_count
",          ), UVM_NONE)
        endfunction

```

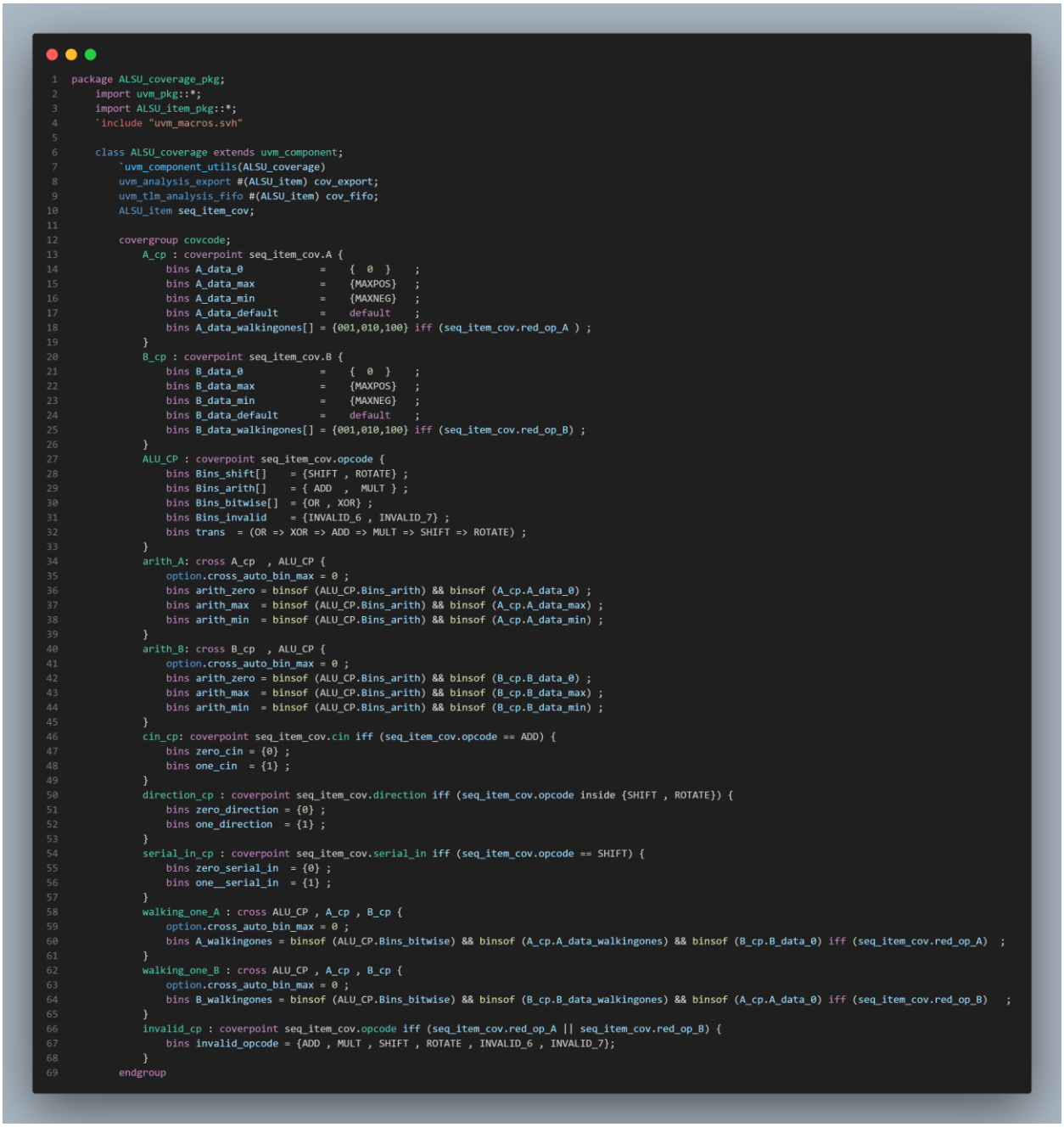
```
task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                tmp = ALSU_item::type_id::create("tmp");
                sb_fifo.get(seq_item_sb);
                seq_item_sb.copy(tmp);
                ref_pipe.push_back(tmp);
                if (ref_pipe.size() >= 2) begin
                    ALSU_item delayed_item;
                    delayed_item = ref_pipe.pop_front();
                    ref_seq_item_chk_model(delayed_item);
                    if (delayed_item.out != out_ref || delayed_item.leds != leds_ref) begin
                        `uvm_error("SCOREBOARD", $sformatf(
                                    "Comparison FAIL: opcode=
, A=                                    delayed_item.opcode, delayed_item.A, delayed_item.B,
, B=                                    delayed_item.out, delayed_item.leds, out_ref, leds_ref))
, got out=              error_count++;
 leds=%0h, expected out=
 leds=%0h",         else correct_count++;
                end
            end
        endtask : run_phase


        task ref_seq_item_chk_model(ALSU_item seq_item_chk);
            leds_ref = 0 ;
            invalid = (seq_item_sb.opcode == 3'b110) ||
                        (seq_item_sb.opcode == 3'b111) ||
                        ((seq_item_sb.red_op_A || seq_item_sb.red_op_B) &&
                        (seq_item_sb.opcode != 3'b000) && (seq_item_sb.opcode != 3'b001)) ;
            if (seq_item_sb.rst) begin
                out_ref  = 0 ;
            end
            else if (seq_item_sb.bypass_A) out_ref = seq_item_sb.A;
            else if (seq_item_sb.bypass_B) out_ref = seq_item_sb.B;
            else if (invalid) out_ref = 0;
            else if (seq_item_sb.opcode == 0) begin
                if (seq_item_sb.red_op_A) out_ref = |seq_item_sb.A ;
                else if (seq_item_sb.red_op_B) out_ref = |seq_item_sb.B ;
                else out_ref = seq_item_sb.A | seq_item_sb.B ;
            end
            else if (seq_item_sb.opcode == 1) begin
                if (seq_item_sb.red_op_A) out_ref = ^seq_item_sb.A ;
                else if (seq_item_sb.red_op_B) out_ref = ^seq_item_sb.B ;
                else out_ref = seq_item_sb.A ^ seq_item_sb.B ;
            end
            else if (seq_item_sb.opcode == 2) out_ref = seq_item_sb.A + seq_item_sb.B + seq_item_sb.cin ;
            else if (seq_item_sb.opcode == 3) out_ref = seq_item_sb.A * seq_item_sb.B ;
            else if (seq_item_sb.opcode == 4) begin
                if (seq_item_sb.direction) out_ref = {out_ref[4:0] , seq_item_sb.serial_in } ;
                else               out_ref = {seq_item_sb.serial_in , out_ref[5:1] } ;
            end
            else if (seq_item_sb.opcode == 5) begin
                if (seq_item_sb.direction) out_ref = {out_ref[4:0] , out_ref[5] } ;
                else               out_ref = {out_ref[0] , out_ref[5:1] } ;
            end

            if (seq_item_sb.rst) leds_ref = 0 ;
            begin
                if (invalid)
                leds_ref = ~leds_ref;
            end
        endtask
    endclass

endpackage
```

16. ALSU_coverage:



```
package ALSU_coverage_pkg;
    import uvm_pkg::*;
    import ALSU_item_pkg::*;
    `include "uvm_macros.svh"

    class ALSU_coverage extends uvm_component;
        `uvm_component_utils(ALSU_coverage)
        uvm_analysis_export #(ALSU_item) cov_export;
        uvm_tlm_analysis_fifo #(ALSU_item) cov_fifo;
        ALSU_item seq_item_cov;

        covergroup covcode;
            A_cp : coverpoint seq_item_cov.A {
                bins A_data_0             =    {  0  }    ;
                bins A_data_max           =    {MAXPOS}   ;
                bins A_data_min           =    {MAXNEG}   ;
                bins A_data_default       =    default    ;
                bins A_data_walkingones[] = {001,010,100} iff (seq_item_cov.red_op_A ) ;
            }
            B_cp : coverpoint seq_item_cov.B {
                bins B_data_0             =    {  0  }    ;
                bins B_data_max           =    {MAXPOS}   ;
                bins B_data_min           =    {MAXNEG}   ;
                bins B_data_default       =    default    ;
                bins B_data_walkingones[] = {001,010,100} iff (seq_item_cov.red_op_B) ;
            }
            ALU_CP : coverpoint seq_item_cov.opcode {
                bins Bins_shift[]    = {SHIFT , ROTATE} ;
                bins Bins_arith[]    = { ADD  ,  MULT } ;
                bins Bins_bitwise[]  = {OR , XOR} ;
                bins Bins_invalid    = {INVALID_6 , INVALID_7} ;
                bins trans  = (OR => XOR => ADD => MULT => SHIFT => ROTATE) ;
            }
            arith_A: cross A_cp  , ALU_CP {
                option.cross_auto_bin_max = 0 ;
                bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_0) ;
                bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_max) ;
                bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (A_cp.A_data_min) ;
            }
            arith_B: cross B_cp  , ALU_CP {
                option.cross_auto_bin_max = 0 ;
                bins arith_zero = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_0) ;
                bins arith_max  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_max) ;
                bins arith_min  = binsof (ALU_CP.Bins_arith) && binsof (B_cp.B_data_min) ;
            }
            cin_cp: coverpoint seq_item_cov.cin iff (seq_item_cov.opcode == ADD) {
                bins zero_cin = {0} ;
                bins one_cin  = {1} ;
            }
            direction_cp : coverpoint seq_item_cov.direction iff (seq_item_cov.opcode inside {SHIFT , ROTATE}) {
                bins zero_direction = {0} ;
                bins one_direction  = {1} ;
            }
            serial_in_cp : coverpoint seq_item_cov.serial_in iff (seq_item_cov.opcode == SHIFT) {
                bins zero_serial_in  = {0} ;
                bins one__serial_in  = {1} ;
            }
            walking_one_A : cross ALU_CP , A_cp , B_cp {
                option.cross_auto_bin_max = 0 ;
                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)  ;
            }
            walking_one_B : cross ALU_CP , A_cp , B_cp {
                option.cross_auto_bin_max = 0 ;
                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)   ;
            }
            invalid_cp : coverpoint seq_item_cov.opcode iff (seq_item_cov.red_op_A || seq_item_cov.red_op_B) {
                bins invalid_opcode = {ADD , MULT , SHIFT , ROTATE , INVALID_6 , INVALID_7};
            }
        endgroup
```

```
    function new (string name = "ALSU_coverage" , uvm_component parent = null);
            super.new(name , parent) ;
            covcode = new() ;
        endfunction

        function void build_phase (uvm_phase phase);
            super.build_phase(phase);
            cov_export = new("cov_export",this);
            cov_fifo   = new("cov_fifo",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            cov_export.connect(cov_fifo.analysis_export);
        endfunction

        task run_phase (uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item_cov);
                covcode.sample();
            end
        endtask
    endclass
endpackage
```

17.ALSU_assertion:

```
import uvm_pkg::*;
`include "uvm_macros.svh"

module ALSU_assertions #(
    parameter INPUT_PRIORITY = "A"
) (
    input logic clk,
    input logic rst,
    input logic signed [5:0] out,
    input logic [15:0] leds,
    input logic [2:0] opcode,
    input logic bypass_A,
    input logic bypass_B,
    input logic red_op_A,
    input logic red_op_B,
    input logic direction,
    input logic signed [2:0] A,
    input logic signed [2:0] B,
    input logic [1:0] cin,
    input logic serial_in
);

    // Internal signal to track previous values for shift operations
    logic signed [5:0] out_prev;
    always @(posedge clk) begin
        out_prev <= out;
    end

    // 1. After reset, outputs must be zero (with proper timing)
    property reset_clears_outputs;
        @(posedge clk) rst |=> (out == 0) and (leds == 0);
    endproperty
    assert property(reset_clears_outputs)
        else `uvm_error("ASSERT", "Outputs not cleared after reset");

    // 2. Invalid opcode (110=6, 111=7) -> out must be 0 after 2 cycles
    property invalid_opcode_out_zero;
        @(posedge clk) disable iff (rst)
            (opcode inside {3'b110, 3'b111} && !bypass_A && !bypass_B) |=> ##1  (out == 0);
    endproperty
    assert property(invalid_opcode_out_zero)
        else `uvm_error("ASSERT", "Invalid opcode did not force out=0");

    // 3. Invalid opcode causes LEDs to toggle
    property invalid_opcode_leds_toggle;
        @(posedge clk) disable iff (rst)
            (opcode inside {3'b110, 3'b111}) |=> ##1 (leds == ~$past(leds));
    endproperty
    assert property(invalid_opcode_leds_toggle)
        else `uvm_error("ASSERT", "LEDs did not toggle on invalid opcode");
```

```
// 4. Valid opcode -> LEDs must be 0
    property valid_opcode_leds_zero;
        @(posedge clk) disable iff (rst)
            !(opcode inside {3'b110, 3'b111}) &&
            !((red_op_A | red_op_B) && (opcode[1] | opcode[2]))
            |=> ##1 (leds == 0);
    endproperty
    assert property(valid_opcode_leds_zero)
        else `uvm_error("ASSERT", "LEDs not zero for valid operation");

    // 5. Bypass_A active (and bypass_B not active) -> out = A after 2 cycles
    property bypass_A_correct;
        @(posedge clk) disable iff (rst)
            (bypass_A && !bypass_B) |-> ##2 (out == $past(A, 2));
    endproperty
    assert property(bypass_A_correct)
        else `uvm_error("ASSERT", "Bypass A mismatch");

    // 6. Bypass_B active (and bypass_A not active) -> out = B after 2 cycles
    property bypass_B_correct;
        @(posedge clk) disable iff (rst)
            (bypass_B && !bypass_A) |-> ##2 (out == $past(B, 2));
    endproperty
    assert property(bypass_B_correct)
        else `uvm_error("ASSERT", "Bypass B mismatch");

    // 7. Both bypass_A and bypass_B active -> priority based on parameter
    property both_bypass_priority_A;
        @(posedge clk) disable iff (rst)
            (bypass_A && bypass_B) |-> ##2 (out == $past(A, 2));
    endproperty
    assert property(both_bypass_priority_A)
        else `uvm_error("ASSERT", "Both bypass active - priority A failed");

    // 8. OR operation (opcode=000) without reduction
    property or_operation;
        @(posedge clk) disable iff (rst)
            (opcode == 3'b000 && !red_op_A && !red_op_B && !bypass_A && !bypass_B)
            |-> ##2 (out == ($past(A, 2) | $past(B, 2)));
    endproperty
    assert property(or_operation)
        else `uvm_error("ASSERT", "OR operation failed");

    // 9. XOR operation (opcode=001) without reduction
    property xor_operation;
        @(posedge clk) disable iff (rst)
            (opcode == 3'b001 && !red_op_A && !red_op_B && !bypass_A && !bypass_B)
            |-> ##2 (out == ($past(A, 2) ^ $past(B, 2)));
    endproperty
    assert property(xor_operation)
        else `uvm_error("ASSERT", "XOR operation failed");

    // 10. Reduction OR on A (opcode=000, red_op_A=1, red_op_B=0)
    property reduction_or_A;
        @(posedge clk) disable iff (rst)
            (opcode == 3'b000 && red_op_A && !red_op_B && !bypass_A && !bypass_B)
            |-> ##2 (out == |$past(A, 2));
    endproperty
    assert property(reduction_or_A)
        else `uvm_error("ASSERT", "Reduction OR A failed");

    // 11. Invalid red_op with non-bitwise opcode -> out = 0
    property invalid_red_op;
        @(posedge clk) disable iff (rst)
            ((red_op_A | red_op_B) && (opcode[1] | opcode[2]) && !bypass_A && !bypass_B)
            |-> ##2 (out == 0);
    endproperty
    assert property(invalid_red_op)
        else `uvm_error("ASSERT", "Invalid red_op did not force out=0");


endmodule
```

18.DO file

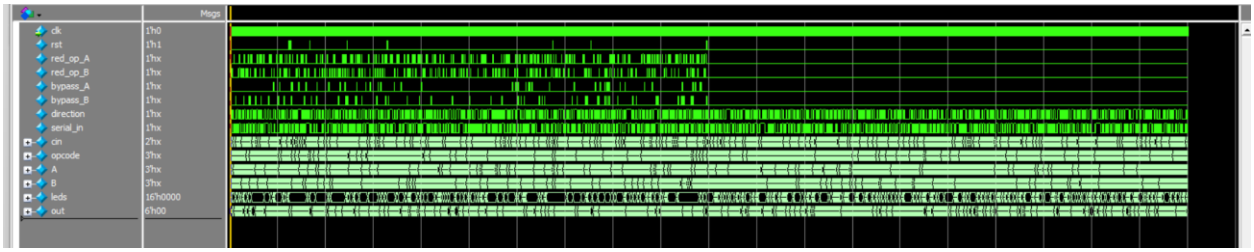
```
vlib work
vlog -f src_files.list
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all
add wave /top/inst_if/*
coverage save ALSU.ucdb -onexit
run -all
```

19.Source file

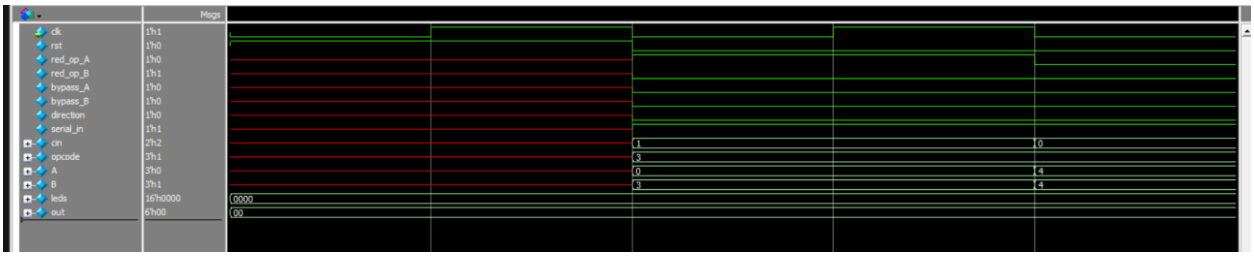
```
ALSU_if.sv
ALSU.v
ALSU_config.sv
ALSU_seq_item.sv
ALSU_seq1.sv
ALSU_seq2.sv
ALSU_reset_seq.sv
ALSU_sequencer.sv
ALSU_driver.sv
ALSU_monitor.sv
ALSU_scoreboard.sv
ALSU_coverage.sv
ALSU_agent.sv
ALSU_env.sv
ALSU_test.sv
ALSU_assertion.sv
top.sv
```

20. Questa_sim Waveform:

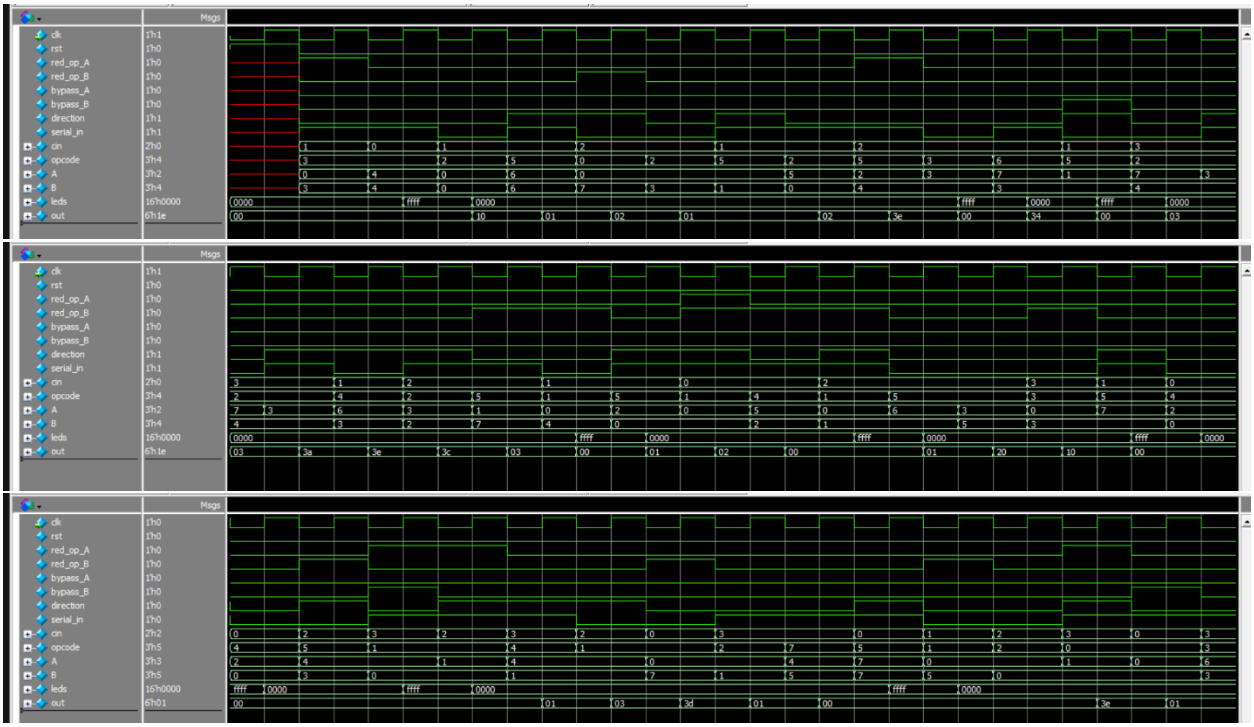
- **Waveform showing whole driving data successful:**

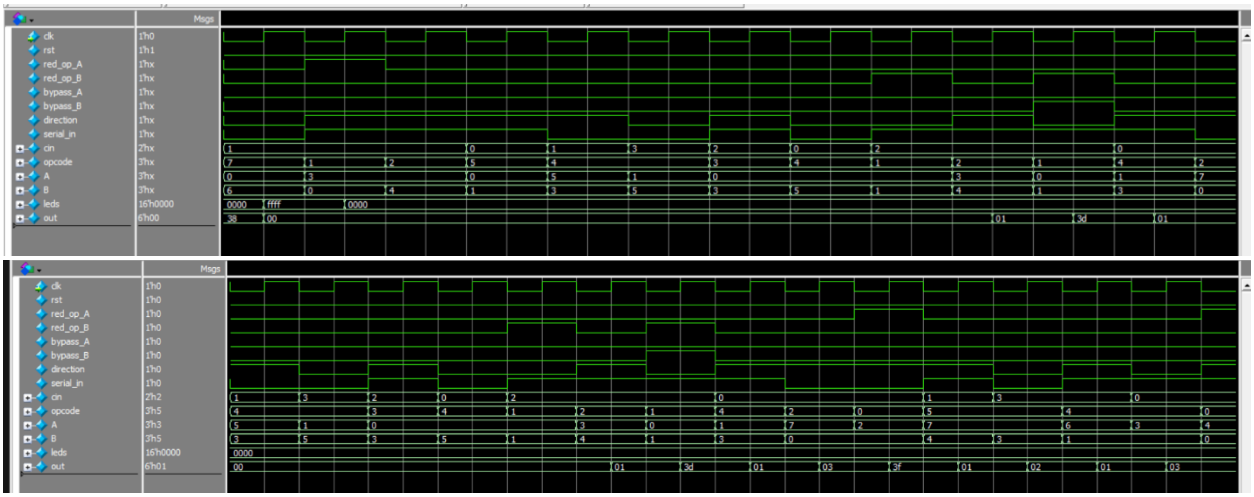


- **Waveforms showing the reset behavior:**

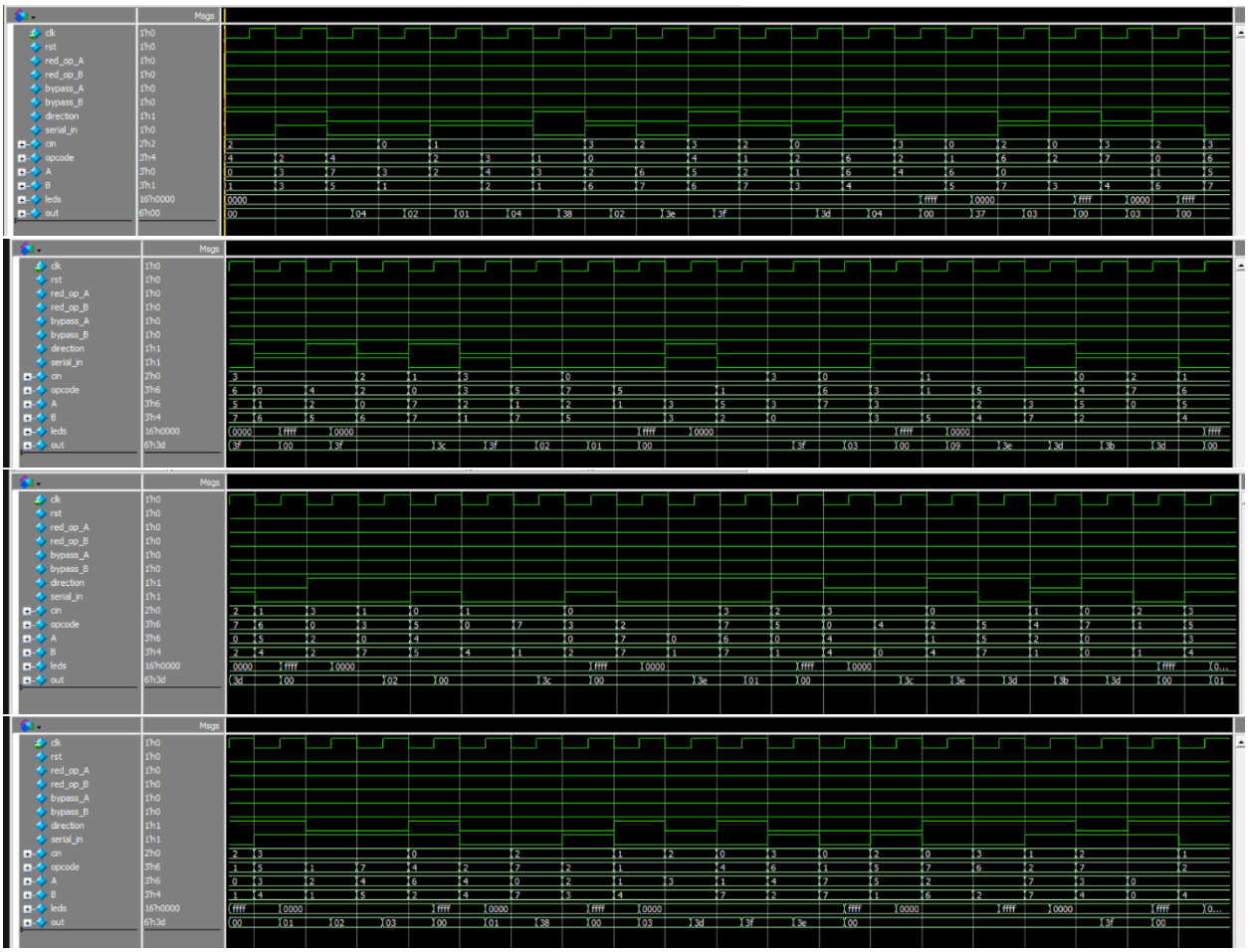


- **Waveforms showing the behavior of the first sequence:**



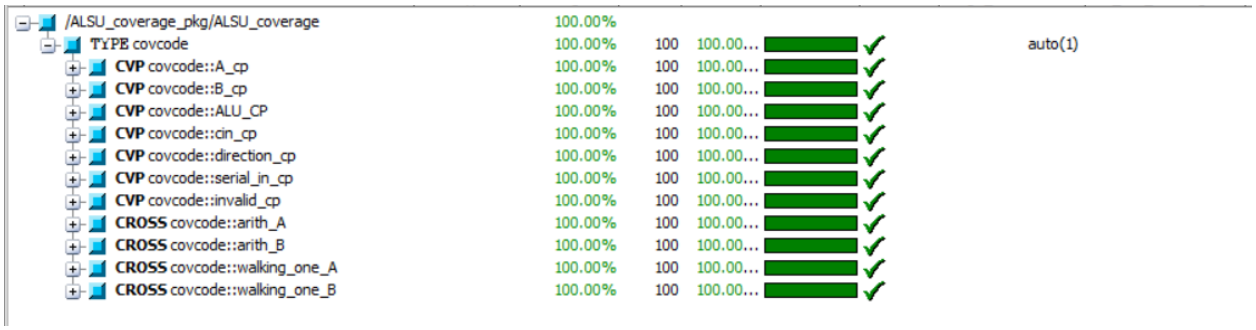


- **Waveforms showing the behavior of the second sequence:**



- **Waveform showing no assertion error:**

21.Coverage report:




```
COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------
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
    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: 1
```

22.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 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
#
# ASSERTION RESULTS:
# ---------------------------------------------------------------------
# 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
# /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
```

23. UVM report showing no errors :

```
# ****************************************************************************
# UVM_INFO ALSU_test.sv(42) @ 2: uvm_test_top [run_phase] reset deasserted
# UVM_INFO ALSU_test.sv(44) @ 2: uvm_test_top [run_phase] seq_1 stimulis generated started
# UVM_INFO ALSU_test.sv(46) @ 2002: uvm_test_top [run_phase] seq_1 stimulis generated ended
# UVM_INFO ALSU_test.sv(48) @ 2002: uvm_test_top [run_phase] seq_2 stimulis generated started
# UVM_INFO ALSU_test.sv(50) @ 4002: uvm_test_top [run_phase] seq_2 stimulis generated ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 4002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO ALSU_scoreboard.sv(38) @ 4002: uvm_test_top.env.sb [SCOREBOARD] Simulation Summary: correct_count=2000, error_count=0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :   11
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [SCOREBOARD]     1
# [TEST_DONE]     1
# [run_phase]     6
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
```