Assignment 6

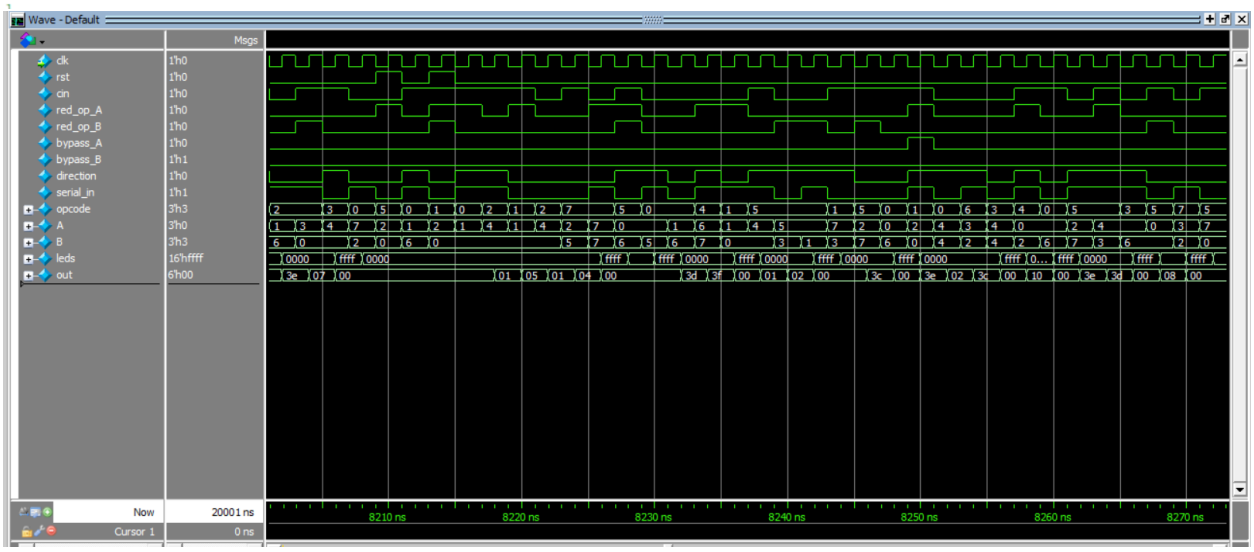
Submitted to: ENG. Kareem Waseem

Part1:

```
UVM_INFO alsu_test_pkg.sv(49) @ 20001: uvm_test_top [run_phase] Stimulus generation started
UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20001: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase

--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :    6
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[Questa UVM]     2
[RNTST]     1
[TEST_DONE]     1
[run_phase]     2
** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
   Time: 20001 ns  Iteration: 61  Instance: /top
```



Top:

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

module top();
    bit clk;
    initial begin
        clk=1;
        forever
            #1 clk=~clk;
    end

    alsu_if alsu_Vif (clk);
    ALSU #(.INPUT_PRIORITY(alsu_Vif.INPUT_PRIORITY), .FULL_ADDER(alsu_Vif.FULL_ADDER)) DUT (
        clk,
        alsu_Vif.A,
        alsu_Vif.B,
        alsu_Vif.cin,
        alsu_Vif.serial_in,
        alsu_Vif.red_op_A,
        alsu_Vif.red_op_B,
        alsu_Vif.opcode,
        alsu_Vif.bypass_A,
        alsu_Vif.bypass_B,
        alsu_Vif.rst,
        alsu_Vif.direction,
        alsu_Vif.leds,
        alsu_Vif.out);

    bind  ALSU alsu_SVA #(.INPUT_PRIORITY(alsu_Vif.INPUT_PRIORITY),.FULL_ADDER(alsu_Vif.FULL_ADDER)) m1 (
        clk,
        alsu_Vif.A,
        alsu_Vif.B,
        alsu_Vif.cin,
        alsu_Vif.serial_in,
        alsu_Vif.red_op_A,
        alsu_Vif.red_op_B,
        alsu_Vif.opcode,
        alsu_Vif.bypass_A,
        alsu_Vif.bypass_B,
        alsu_Vif.rst,
        alsu_Vif.direction,
        alsu_Vif.leds,
        alsu_Vif.out);

    initial begin
        uvm_config_db #(virtual alsu_if)::set(null, "uvm_test_top", "alsu_Vif", alsu_Vif);
        run_test("alsu_test");
    end
endmodule
```

Interface:

```
interface alsu_if (clk);

  parameter INPUT_PRIORITY = "A";
  parameter FULL_ADDER = "ON";
  input clk;
  logic rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
  logic [2:0] opcode;
  logic signed [2:0] A, B;
  logic signed [15:0] leds;
  logic signed [5:0] out;

endinterface
```

Config:

```
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

```

Test:

```
package alsu_test_pkg;
    import alsu_env_pkg::*;
    import alsu_driver_pkg::*;
    import alsu_config_pkg::*;
    import alsu_main_sequence_pkg::*;
    import alsu_reset_sequence_pkg::*;
    import alsu_agent_pkg::*;
    import MySequencer_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;
        virtual alsu_if alsu_Vif;
        alsu_reset_sequence reset_seq;
        alsu_main_sequence main_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");
            reset_seq = alsu_reset_sequence::type_id::create("reset_seq");
            main_seq = alsu_main_sequence::type_id::create("main_seq");

            if (!uvm_config_db #(virtual alsu_if)::get(this,"","alsu_Vif",alsu_cfg.alsu_Vif)) begin
                        `uvm_fatal("build_phase","Test - unable to get the virtual interface");
             end

            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);

            reset_seq.start(env.agt.sqr);
            `uvm_info("run_phase","Reset asserted", UVM_LOW)

            main_seq.start(env.agt.sqr);
            `uvm_info("run_phase","Stimulus generation started", UVM_LOW)

            phase.drop_objection(this);
        endtask
    endclass
endpackage
```

Env:

```
package alsu_env_pkg;
    import alsu_agent_pkg::*;
    //import alsu_scoreboard_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_scoreboard sb;
        alsu_coverage cov;

        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);
            //sb=alsu_scoreboard::type_id::create("sb",this);
            cov=alsu_coverage::type_id::create("cov",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 : connect_phase

    endclass
endpackage

```

Agent:

```
package alsu_agent_pkg;
    import alsu_driver_pkg::*;
    import MySequencer_pkg::*;
    import alsu_config_pkg::*;
    import alsu_monitor_pkg::*;
    import alsu_sequence_item_pkg::*;

    import uvm_pkg::*;
    `include "uvm_macros.svh"
    class alsu_agent extends uvm_agent ;
        `uvm_component_utils(alsu_agent)
        MySequencer sqr;
        alsu_driver drv;
        alsu_monitor mon;
        alsu_config alsu_cfg;
        uvm_analysis_port #(alsu_sequence_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);
            drv = alsu_driver::type_id::create("drv",this);
            sqr = MySequencer::type_id::create("sqr",this);
            mon = alsu_monitor::type_id::create("mon",this);
            agt_ap =new("agt_ap",this);
            if (!uvm_config_db #(alsu_config)::get(this,"","CFG",alsu_cfg)) begin
                `uvm_fatal("build_phase","Driver - unable to get configuration object");
             end
        endfunction : build_phase

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

Driver:

```
package alsu_driver_pkg;
  import alsu_config_pkg::*;
  import alsu_sequence_item_pkg::*;
  import alsu_main_sequence_pkg::*;
  import alsu_reset_sequence_pkg::*;

  import uvm_pkg::*;
  `include "uvm_macros.svh"
  class alsu_driver extends uvm_driver #(alsu_sequence_item);
    `uvm_component_utils(alsu_driver);
    virtual alsu_if alsu_Vif;
    alsu_sequence_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_sequence_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.cin=stim_seq_item.cin;
        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.opcode=stim_seq_item.opcode;
        alsu_Vif.A=stim_seq_item.A;
        alsu_Vif.B=stim_seq_item.B;
        @(negedge alsu_Vif.clk);
        seq_item_port.item_done();
        `uvm_info("run_phase",stim_seq_item.convert2string_stimulus(), UVM_HIGH)
      end
    endtask
  endclass
endpackage
```

Monitor:

```
package alsu_monitor_pkg;
    import alsu_config_pkg::*;
    import alsu_sequence_item_pkg::*;
    import alsu_shared_pkg::*;

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

    class alsu_monitor extends uvm_monitor ;
        `uvm_component_utils(alsu_monitor);
        virtual alsu_if alsu_Vif;
        alsu_sequence_item rsp_seq_item;
        uvm_analysis_port #(alsu_sequence_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_sequence_item::type_id::create("rsp_seq_item");
                @(negedge alsu_Vif.clk);

                rsp_seq_item.rst=alsu_Vif.rst;
                rsp_seq_item.cin=alsu_Vif.cin;
                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.direction=alsu_Vif.direction;
                rsp_seq_item.serial_in=alsu_Vif.serial_in;
                rsp_seq_item.opcode=opcode_e'(alsu_Vif.opcode);
                rsp_seq_item.A=alsu_Vif.A;
                rsp_seq_item.B=alsu_Vif.B;
                rsp_seq_item.leds=alsu_Vif.leds;
                rsp_seq_item.out=alsu_Vif.out;

                mon_ap.write(rsp_seq_item);
                `uvm_info("run_phase",rsp_seq_item.convert2string_stimulus(), UVM_HIGH)
            end
        endtask

    endclass
endpackage
```

Sequence Item:

```
package alsu_sequence_item_pkg;

    import alsu_shared_pkg::*;
    import uvm_pkg::*;

    `include "uvm_macros.svh"
    class alsu_sequence_item extends uvm_sequence_item ;
        `uvm_object_utils(alsu_sequence_item)

        rand reg_e A_constrained,B_constrained;
        rand bit cin;
        rand bit serial_in;
        rand bit direction;
        rand bit signed [2:0] A, B;
        rand bit red_op_A;
        rand bit red_op_B;
        rand opcode_e opcode;
        rand bit rst;
        rand bit bypass_A, bypass_B;
        rand bit signed [2:0] A_rem_rand, B_rem_rand;
        bit [2:0] ones [] ='{3'b001, 3'b010, 3'b100};
        rand bit [2:0] only_ones, no_only_ones;
        logic signed [15:0] leds;
        logic signed [5:0] out;

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

        constraint reset {
            rst dist {0:/98, 1:/2};
        }

        constraint A_and_B {
            A_rem_rand != MAXPOS || 0 || MAXNEG;
            B_rem_rand != MAXPOS || 0 || MAXNEG;
            only_ones inside {ones};
            !(no_only_ones inside {ones});

            if (opcode inside {OR,XOR}){
                if(red_op_A){
                    B==0;
                    A dist {only_ones:/90, no_only_ones:/10};

                }
                else if (red_op_B) {
                    A==0;
                    B dist {only_ones:/90, no_only_ones:/10};
                    }
```

```
            }
            else {
                red_op_A dist {1:/20, 0:/80};
                red_op_B dist {1:/20, 0:/80};
                if(opcode inside {ADD, MULT}){
                    A dist {A_constrained:/80, A_rem_rand:/20};
                    B dist {B_constrained:/80, B_rem_rand:/20};
                    }

                }
            }

        constraint opcode_values
        {
            opcode dist {[OR:ROTATE]:/80, [INVALID_6:INVALID_7]:/20};
        }


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

        constraint cin_values {cin dist {1:/50,0:/50};}

        constraint serial_in_values {serial_in dist {1:/50,0:/50};}

        constraint direction_values {direction dist {1:/50,0:/50};}


        function string convert2string();
            return $sformatf("%s rst_rand = 0b%0b,  cin_rand = 0b%0b, red_op_A_rand = 0b%0b, red_op_B_rand = 0b%0b, bypass_A_rand = 0b%0b, bypass_B_rand =0b%0b
                              , direction_rand =0b%0b, serial_in_rand =0b%0b, opcode_rand =%s, A_rand =0b%0b, B_rand =0b%0b, leds =0b%0b, out =0b%0b",
             super.convert2string(), rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in
            , opcode, A, B, leds, out);
        endfunction : convert2string

        function string convert2string_stimulus();
            return $sformatf("%s rst_rand = 0b%0b,  cin_rand = 0b%0b, red_op_A_rand = 0b%0b, red_op_B_rand =0b%0b, bypass_A_rand = 0b%0b, bypass_B_rand =0b%0b
                              , direction_rand =0b%0b, serial_in_rand =0b%0b, opcode_rand =%s, A_rand =0b%0b, B_rand =0b%0b",
             super.convert2string(), rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in
            , opcode, A, B);
        endfunction : convert2string_stimulus
    endclass


endpackage
```

Sequencer:

```
package MySequencer_pkg;
import alsu_sequence_item_pkg::*;

import uvm_pkg::*;
`include "uvm_macros.svh"
class MySequencer extends uvm_sequencer #(alsu_sequence_item);
    `uvm_component_utils(MySequencer)

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

endclass : MySequencer
endpackage : MySequencer_pkg
```

Reset sequence:

```
1   package alsu_reset_sequence_pkg;
2       import uvm_pkg::*;
3       import alsu_sequence_item_pkg::*;
4       import alsu_shared_pkg::*;
5
6       `include "uvm_macros.svh"
7       class alsu_reset_sequence extends uvm_sequence #(alsu_sequence_item) ;
8           `uvm_object_utils(alsu_reset_sequence)
9
10          alsu_sequence_item seq_item;
11
12          function new(string name = "alsu_reset_sequence");
13              super.new(name);
14          endfunction
15
16          task body;
17              seq_item = alsu_sequence_item::type_id::create("seq_item");
18              start_item(seq_item);
19              seq_item.rst=1;
20              seq_item.serial_in=0;
21              seq_item.opcode=opcode_e'(0);
22              seq_item.direction=0;
23              seq_item.bypass_A=0;
24              seq_item.bypass_B=0;
25              seq_item.red_op_A=0;
26              seq_item.red_op_B=0;
27              seq_item.cin=0;
28              seq_item.A=0;
29              seq_item.B=0;
30              finish_item(seq_item);
31          endtask : body
32      endclass
33
34
35  endpackage
36
```

Main Sequence:

```
1   package alsu_main_sequence_pkg;
2       import uvm_pkg::*;
3       import alsu_sequence_item_pkg::*;
4       `include "uvm_macros.svh"
5       class alsu_main_sequence extends uvm_sequence #(alsu_sequence_item) ;
6           `uvm_object_utils(alsu_main_sequence)
7
8           alsu_sequence_item seq_item;
9
10          function new(string name = "alsu_main_sequence");
11              super.new(name);
12          endfunction
13
14          task body;
15              repeat(10_000) begin
16                  seq_item=alsu_sequence_item::type_id::create("seq_item");
17                  start_item(seq_item);
18                  assert(seq_item.randomize());
19                  finish_item(seq_item);
20              end
21          endtask : body
22      endclass
23  endpackage
```

Coverage:

```
package alsu_coverage_pkg;
    import uvm_pkg::*;
    import alsu_shared_pkg::*;
    import alsu_sequence_item_pkg::*;
    `include "uvm_macros.svh"
    class alsu_coverage extends uvm_component;
        `uvm_component_utils(alsu_coverage)
        uvm_analysis_export #(alsu_sequence_item) cov_export;
        uvm_tlm_analysis_fifo #(alsu_sequence_item) cov_fifo;
        alsu_sequence_item seq_item_cov;

        // Covergroups
        covergroup cvr_gp();

            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;
            }
            A_Walking: coverpoint seq_item_cov.A iff(seq_item_cov.red_op_A)
            {
                bins A_data_ones[] = {1,2,-4};
            }

            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;
            }
            B_Walking: coverpoint seq_item_cov.B iff(seq_item_cov.red_op_B & !seq_item_cov.red_op_A)
            {
                bins B_data_ones[] = {1,2,-4};
            }

            ALU_cvp: coverpoint seq_item_cov.opcode
            {
                bins Bins_shift [] = {[SHIFT:ROTATE]};
                bins Bins_arith [] = {[ADD:MULT]};
                illegal_bins Bins_invalid [] = {[INVALID_6:INVALID_7]};
                bins Bins_trans = (OR => XOR => ADD => MULT => SHIFT => ROTATE);
            }

            opcode_bitwise_cp: coverpoint seq_item_cov.opcode{
                bins bins_bitwise[] = {OR, XOR};
            }

```

```
            cross_ARITH_PERM: cross A_cp, B_cp, ALU_cvp{
                ignore_bins ig_bins_shift = binsof(ALU_cvp.Bins_shift);
                ignore_bins ig_bins_trans = binsof(ALU_cvp.Bins_trans);
            }


            cross_SHIFT_opcode: cross seq_item_cov.direction, ALU_cvp{
                ignore_bins ig_bins_all = !binsof(ALU_cvp.Bins_shift);
            }

            cross_ARITH_CIN: cross seq_item_cov.cin, ALU_cvp{
                ignore_bins ig_bins_shift = binsof(ALU_cvp.Bins_shift);
                ignore_bins ig_bins_trans = binsof(ALU_cvp.Bins_trans);
            }

        endgroup

        function new(string name = "alsu_coverage", uvm_component parent = null);
            super.new(name,parent);
            cvr_gp=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 : build_phase

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                cov_fifo.get(seq_item_cov);
                cvr_gp.sample();
            end
        endtask : run_phase


    endclass
endpackage
```

Assertions:

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

    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";

    wire invalid_red_op, invalid_opcode, invalid;

    assign invalid_red_op = (red_op_A | red_op_B) & (opcode[1] | opcode[2]);
    assign invalid_opcode = opcode[1] & opcode[2];
    assign invalid = invalid_red_op | invalid_opcode;

    property p_1;
        @(posedge clk) disable iff (rst) invalid |-> ##2 (leds == (~$past(leds)));
    endproperty

    property p_2;
        @(posedge clk) disable iff (rst) invalid |-> ##2 (out == 6'b0);
    endproperty

    property p_3;
        @(posedge clk) disable iff (rst)
            (bypass_A && bypass_B && !invalid) |-> ##2(out == ((INPUT_PRIORITY == "A") ? $past(A,2) : $past(B,2)));
    endproperty

    property p_4;
        @(posedge clk) disable iff (rst)
            (bypass_A && !bypass_B && !invalid) |-> ##2 (out == $past(A,2));
    endproperty

    property p_5;
        @(posedge clk) disable iff (rst)
            (bypass_B && !bypass_A && !invalid) |-> ##2 (out == $past(B,2));
    endproperty
```

```
    property p_6;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h0 && !invalid && !bypass_A && !bypass_B && red_op_A && red_op_B) |-> ##2 (out == ((INPUT_PRIORITY == "A") ? |$past(A,2) : |$past(B,2)));
    endproperty

    property p_7;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h0 && !invalid && !bypass_A && !bypass_B && red_op_A && !red_op_B) |-> ##2 (out == |$past(A,2));
    endproperty

    property p_8;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h0 && !invalid && !bypass_A && !bypass_B && !red_op_A && red_op_B) |-> ##2 (out == |$past(B,2));
    endproperty

    property p_9;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h0 && !invalid && !bypass_A && !bypass_B && !red_op_A && !red_op_B) |-> ##2 (out == ($past(B,2) | $past(A,2)));
    endproperty

    property p_A;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h1 && !invalid && !bypass_A && !bypass_B && red_op_A && red_op_B) |-> ##2 (out == ((INPUT_PRIORITY == "A") ? ^$past(A,2) : ^$past(B,2)));
    endproperty

    property p_B;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h1 && !invalid && !bypass_A && !bypass_B && red_op_A && !red_op_B) |-> ##2 (out == ^$past(A,2));
    endproperty

    property p_C;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h1 && !invalid && !bypass_A && !bypass_B && !red_op_A && red_op_B) |-> ##2 (out == ^$past(B,2));
    endproperty

    property p_D;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h1 && !invalid && !bypass_A && !bypass_B && !red_op_A && !red_op_B) |-> ##2 (out == ($past(B,2) ^ $past(A,2)));
    endproperty


    property p_E;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h2 && !invalid && !bypass_A && !bypass_B) |-> ##2 (out == ((FULL_ADDER == "ON") ? ($past(A,2) + $past(B,2) + $past(cin,2)) : ($past(A,2) + $past(B,2))));
    endproperty

    property p_F;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h3 && !invalid && !bypass_A && !bypass_B) |-> ##2 (out == $past(A,2) * $past(B,2));
    endproperty
```

```
    property p_G;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h4 && !invalid && !bypass_A && !bypass_B && direction) |-> ##2 (out == ({$past(out[4:0]), $past(serial_in,2)}));
    endproperty

    property p_H;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h4 && !invalid && !bypass_A && !bypass_B && !direction) |-> ##2 (out == ({$past(serial_in,2), $past(out[5:1])}));
    endproperty

    property p_I;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h5 && !invalid && !bypass_A && !bypass_B && direction) |-> ##2 (out == ({$past(out[4:0]), $past(out[5])}));
    endproperty

    property p_J;
        @(posedge clk) disable iff (rst)
            (opcode == 3'h5 && !invalid && !bypass_A && !bypass_B && !direction) |-> ##2 (out == ({$past(out[0]), $past(out[5:1])}));
    endproperty



        AP: assert property (p_1) else $display("p_1 failed");
        BP: assert property (p_2) else $display("p_2 failed");
        CP: assert property (p_3) else $display("p_3 failed");
        DP: assert property (p_4) else $display("p_4 failed");
        EP: assert property (p_5) else $display("p_5 failed");
        FP: assert property (p_6) else $display("p_6 failed");
        GP: assert property (p_7) else $display("p_7 failed");
        HP: assert property (p_8) else $display("p_8 failed");
        IP: assert property (p_9) else $display("p_9 failed");
        JP: assert property (p_A) else $display("p_A failed");
        KP: assert property (p_B) else $display("p_B failed");
        LP: assert property (p_C) else $display("p_C failed");
        MP: assert property (p_D) else $display("p_D failed");
        NP: assert property (p_E) else $display("p_E failed");
        OP: assert property (p_F) else $display("p_F failed");
        PP: assert property (p_G) else $display("p_G failed");
        QP: assert property (p_H) else $display("p_H failed");
        RP: assert property (p_I) else $display("p_I failed");
        SP: assert property (p_J) else $display("p_J failed");
```

```


            Ac: cover property (p_1)  $display("p_1 pass");
            Bc: cover property (p_2)  $display("p_2 pass");
            Cc: cover property (p_3)  $display("p_3 pass");
            Dc: cover property (p_4)  $display("p_4 pass");
            Ec: cover property (p_5)  $display("p_5 pass");
            Fc: cover property (p_6)  $display("p_6 pass");
            Gc: cover property (p_7)  $display("p_7 pass");
            Hc: cover property (p_8)  $display("p_8 pass");
            Ic: cover property (p_9)  $display("p_9 pass");
            Jc: cover property (p_A)  $display("p_A pass");
            Kc: cover property (p_B)  $display("p_B pass");
            Lc: cover property (p_C)  $display("p_C pass");
            Mc: cover property (p_D)  $display("p_D pass");
            Nc: cover property (p_E)  $display("p_E pass");
            Oc: cover property (p_F)  $display("p_F pass");
            Pc: cover property (p_G)  $display("p_G pass");
            Qc: cover property (p_H)  $display("p_H pass");
            Rc: cover property (p_I)  $display("p_I pass");
            Sc: cover property (p_J)  $display("p_J pass");


endmodule
```

Shared package:

```
package alsu_shared_pkg;

    typedef enum logic [2:0] {OR = 3'b000,XOR = 3'b001,ADD = 3'b010,MULT = 3'b011,SHIFT = 3'b100,
                              ROTATE = 3'b101,INVALID_6 = 3'b110,INVALID_7 = 3'b111} opcode_e;

    typedef enum {MAXPOS = 3, ZERO = 0, MAXNEG = -4} reg_e;

endpackage : alsu_shared_pkg
```

Do:

```
vlib work
vlog -f alsu.list.txt +cover -covercells
vsim -voptargs=+acc work.top -cover
add wave /top/alsu_Vif/*
coverage save top.ucdb -onexit
run -all
```

File:

```
ALSU.v
alsu_config_pkg.sv
alsu_if.sv
alsu_main_sequence_pkg.sv
alsu_reset_sequence_pkg.sv
alsu_sequence_item_pkg.sv
MySequencer_pkg.sv
alsu_env_pkg.sv
alsu_test_pkg.sv
alsu_driver_pkg.sv
alsu_agent_pkg.sv
alsu_monitor_pkg.sv
alsu_coverage_pkg.sv
SVA.sv
top.sv
```

Code coverage:

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Me
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/...	Immediate	SVA	on	0	0	-	
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/i...	Immediate	SVA	on	0	0	-	
/alsu_main_sequence_pkg::alsu_main_sequence::body/#ublk...	Immediate	SVA	on	0	1	-	
/top/DUT/m1/AP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/BP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/CP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/DP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/EP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/FP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/GP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/HP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/IP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/JP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/KP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/LP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/MP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/NP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/OP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/PP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/QP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/RP	Concurrent	SVA	on	0	1	-	
/top/DUT/m1/SP	Concurrent	SVA	on	0	1	-	

Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/alsu_coverage_pkg/alsu_coverage		100.00%							
TYPE cvr_gp		100.00%	100	100.00...			auto(0)		
CVP cvr_gp::A_cp		100.00%	100	100.00...					
CVP cvr_gp::A_Walking		100.00%	100	100.00...					
CVP cvr_gp::B_cp		100.00%	100	100.00...					
CVP cvr_gp::B_Walking		100.00%	100	100.00...					
CVP cvr_gp::ALU_cvp		100.00%	100	100.00...					
CVP cvr_gp::opcode_bitwise_cp		100.00%	100	100.00...					
CVP cvr_gp::{#seq_item_cov.cin__0#}		100.00%	100	100.00...					
CVP cvr_gp::{#seq_item_cov.direction__1#}		100.00%	100	100.00...					
CROSS cvr_gp::cross_ARITH_PERM		100.00%	100	100.00...					
CROSS cvr_gp::cross_SHIFT_opcode		100.00%	100	100.00...					
CROSS cvr_gp::cross_ARITH_CIN		100.00%	100	100.00...					
INST \/alsu_coverage_pkg::alsu_coverage::cvr_gp		100.00%	100	100.00...					0
CVP A_cp		100.00%	100	100.00...					
CVP A_Walking		100.00%	100	100.00...					
CVP B_cp		100.00%	100	100.00...					
CVP B_Walking		100.00%	100	100.00...					
CVP ALU_cvp		100.00%	100	100.00...					
CVP opcode_bitwise_cp		100.00%	100	100.00...					
CVP #seq_item_cov.cin__0#		100.00%	100	100.00...					
CVP #seq_item_cov.direction__1#		100.00%	100	100.00...					
CROSS cross_ARITH_PERM		100.00%	100	100.00...					
CROSS cross_SHIFT_opcode		100.00%	100	100.00...					
CROSS cross_ARITH_CIN		100.00%	100	100.00...					

```
================================================================================
=== Instance: /top/alsu_Vif
=== Design Unit: work.alsu_if
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         80        80         0   100.00%

                                Toggle Details
```

```
Assertion Coverage:
    Assertions                      19        19         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/top/DUT/m1/AP       SVA.sv(121)                         0          1
/top/DUT/m1/BP       SVA.sv(122)                         0          1
/top/DUT/m1/CP       SVA.sv(123)                         0          1
/top/DUT/m1/DP       SVA.sv(124)                         0          1
/top/DUT/m1/EP       SVA.sv(125)                         0          1
/top/DUT/m1/FP       SVA.sv(126)                         0          1
/top/DUT/m1/GP       SVA.sv(127)                         0          1
/top/DUT/m1/HP       SVA.sv(128)                         0          1
/top/DUT/m1/IP       SVA.sv(129)                         0          1
/top/DUT/m1/JP       SVA.sv(130)                         0          1
/top/DUT/m1/KP       SVA.sv(131)                         0          1
/top/DUT/m1/LP       SVA.sv(132)                         0          1
/top/DUT/m1/MP       SVA.sv(133)                         0          1
/top/DUT/m1/NP       SVA.sv(134)                         0          1
/top/DUT/m1/OP       SVA.sv(135)                         0          1
/top/DUT/m1/PP       SVA.sv(136)                         0          1
/top/DUT/m1/QP       SVA.sv(137)                         0          1
/top/DUT/m1/RP       SVA.sv(138)                         0          1
/top/DUT/m1/SP       SVA.sv(139)                         0          1

Directive Coverage:
    Directives                      19        19         0   100.00%
```

```
Expression Coverage:
    Enabled Coverage              Bins    Covered    Misses  Coverage
    ----------------              ----    -------    ------  --------
    Expressions                      8          8         0   100.00%
```

```
 === Instance: /top/DUT
 === Design Unit: work.ALSU
 ================================================================================
 Branch Coverage:
     Enabled Coverage              Bins      Hits    Misses  Coverage
     ----------------              ----      ----    ------  --------
     Branches                        32        32         0   100.00%

 ================================Branch Details================================
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      48        48         0   100.00%

================================Statement Details================================
```

```
729 ▼ Toggle Coverage:
730       Enabled Coverage              Bins      Hits    Misses  Coverage
731       ----------------              ----      ----    ------  --------
732       Toggles                        118       118         0   100.00%
733
734   ------------------------------Toggle Details------------------------------
```

```
2840                bin <A_data_0,B_data_max,Bins_arith[MULT]>
2841                                                      122          1          -    Covered
2842                bin <A_data_0,B_data_max,Bins_arith[ADD]>
2843                                                      132          1          -    Covered
2844                bin <A_data_0,B_data_0,Bins_arith[MULT]>
2845                                                      120          1          -    Covered
2846                bin <A_data_0,B_data_0,Bins_arith[ADD]>
2847                                                      109          1          -    Covered
2848            Illegal and Ignore Bins:
2849                ignore_bin ig_bins_trans                0                     -    ZERO
2850                ignore_bin ig_bins_shift              367                     -    Occurred
2851        Cross cross_SHIFT_opcode                  100.00%        100          -    Covered
2852            covered/total bins:                         4          4          -
2853            missing/total bins:                         0          4          -
2854            % Hit:                                100.00%        100          -
2855            Auto, Default and User Defined Bins:
2856                bin <auto[1],Bins_shift[ROTATE]>      664          1          -    Covered
2857                bin <auto[1],Bins_shift[SHIFT]>       643          1          -    Covered
2858                bin <auto[0],Bins_shift[ROTATE]>      655          1          -    Covered
2859                bin <auto[0],Bins_shift[SHIFT]>       661          1          -    Covered
2860            Illegal and Ignore Bins:
2861                ignore_bin ig_bins_all               2729                     -    Occurred
2862        Cross cross_ARITH_CIN                     100.00%        100          -    Covered
2863            covered/total bins:                         4          4          -
2864            missing/total bins:                         0          4          -
2865            % Hit:                                100.00%        100          -
2866            Auto, Default and User Defined Bins:
2867                bin <auto[1],Bins_arith[MULT]>        675          1          -    Covered
2868                bin <auto[1],Bins_arith[ADD]>         672          1          -    Covered
2869                bin <auto[0],Bins_arith[MULT]>        700          1          -    Covered
2870                bin <auto[0],Bins_arith[ADD]>         681          1          -    Covered
2871            Illegal and Ignore Bins:
2872                ignore_bin ig_bins_trans                1                     -    Occurred
2873                ignore_bin ig_bins_shift             2623                     -    Occurred
2874
2875    TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

```
Name                                    Design Design   Lang File(Line)      Hits Status
                                        Unit   UnitType
------------------------------------------------------------------------------------------
/top/DUT/m1/Ac                          alsu_SVA Verilog  SVA  SVA.sv(142)     3724 Covered
/top/DUT/m1/Bc                          alsu_SVA Verilog  SVA  SVA.sv(143)     3724 Covered
/top/DUT/m1/Cc                          alsu_SVA Verilog  SVA  SVA.sv(144)       12 Covered
/top/DUT/m1/Dc                          alsu_SVA Verilog  SVA  SVA.sv(145)      270 Covered
/top/DUT/m1/Ec                          alsu_SVA Verilog  SVA  SVA.sv(146)      257 Covered
/top/DUT/m1/Fc                          alsu_SVA Verilog  SVA  SVA.sv(147)      177 Covered
/top/DUT/m1/Gc                          alsu_SVA Verilog  SVA  SVA.sv(148)      108 Covered
/top/DUT/m1/Hc                          alsu_SVA Verilog  SVA  SVA.sv(149)      114 Covered
/top/DUT/m1/Ic                          alsu_SVA Verilog  SVA  SVA.sv(150)      709 Covered
/top/DUT/m1/Jc                          alsu_SVA Verilog  SVA  SVA.sv(151)      147 Covered
/top/DUT/m1/Kc                          alsu_SVA Verilog  SVA  SVA.sv(152)      108 Covered
/top/DUT/m1/Lc                          alsu_SVA Verilog  SVA  SVA.sv(153)      136 Covered
/top/DUT/m1/Mc                          alsu_SVA Verilog  SVA  SVA.sv(154)      712 Covered
/top/DUT/m1/Nc                          alsu_SVA Verilog  SVA  SVA.sv(155)      745 Covered
/top/DUT/m1/Oc                          alsu_SVA Verilog  SVA  SVA.sv(156)      744 Covered
/top/DUT/m1/Pc                          alsu_SVA Verilog  SVA  SVA.sv(157)      361 Covered
/top/DUT/m1/Qc                          alsu_SVA Verilog  SVA  SVA.sv(158)      358 Covered
/top/DUT/m1/Rc                          alsu_SVA Verilog  SVA  SVA.sv(159)      353 Covered
/top/DUT/m1/Sc                          alsu_SVA Verilog  SVA  SVA.sv(160)      350 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 19
```

Name	Design Unit	Design UnitType	Lang	File(Line)	Hits	Status
/top/DUT/m1/Ac	alsu_SVA	Verilog	SVA	SVA.sv(142)	3724	Covered
/top/DUT/m1/Bc	alsu_SVA	Verilog	SVA	SVA.sv(143)	3724	Covered
/top/DUT/m1/Cc	alsu_SVA	Verilog	SVA	SVA.sv(144)	12	Covered
/top/DUT/m1/Dc	alsu_SVA	Verilog	SVA	SVA.sv(145)	270	Covered
/top/DUT/m1/Ec	alsu_SVA	Verilog	SVA	SVA.sv(146)	257	Covered
/top/DUT/m1/Fc	alsu_SVA	Verilog	SVA	SVA.sv(147)	177	Covered
/top/DUT/m1/Gc	alsu_SVA	Verilog	SVA	SVA.sv(148)	108	Covered
/top/DUT/m1/Hc	alsu_SVA	Verilog	SVA	SVA.sv(149)	114	Covered
/top/DUT/m1/Ic	alsu_SVA	Verilog	SVA	SVA.sv(150)	709	Covered
/top/DUT/m1/Jc	alsu_SVA	Verilog	SVA	SVA.sv(151)	147	Covered
/top/DUT/m1/Kc	alsu_SVA	Verilog	SVA	SVA.sv(152)	108	Covered
/top/DUT/m1/Lc	alsu_SVA	Verilog	SVA	SVA.sv(153)	136	Covered
/top/DUT/m1/Mc	alsu_SVA	Verilog	SVA	SVA.sv(154)	712	Covered
/top/DUT/m1/Nc	alsu_SVA	Verilog	SVA	SVA.sv(155)	745	Covered
/top/DUT/m1/Oc	alsu_SVA	Verilog	SVA	SVA.sv(156)	744	Covered
/top/DUT/m1/Pc	alsu_SVA	Verilog	SVA	SVA.sv(157)	361	Covered
/top/DUT/m1/Qc	alsu_SVA	Verilog	SVA	SVA.sv(158)	358	Covered
/top/DUT/m1/Rc	alsu_SVA	Verilog	SVA	SVA.sv(159)	353	Covered
/top/DUT/m1/Sc	alsu_SVA	Verilog	SVA	SVA.sv(160)	350	Covered

TOTAL DIRECTIVE COVERAGE: 100.00% COVERS: 19