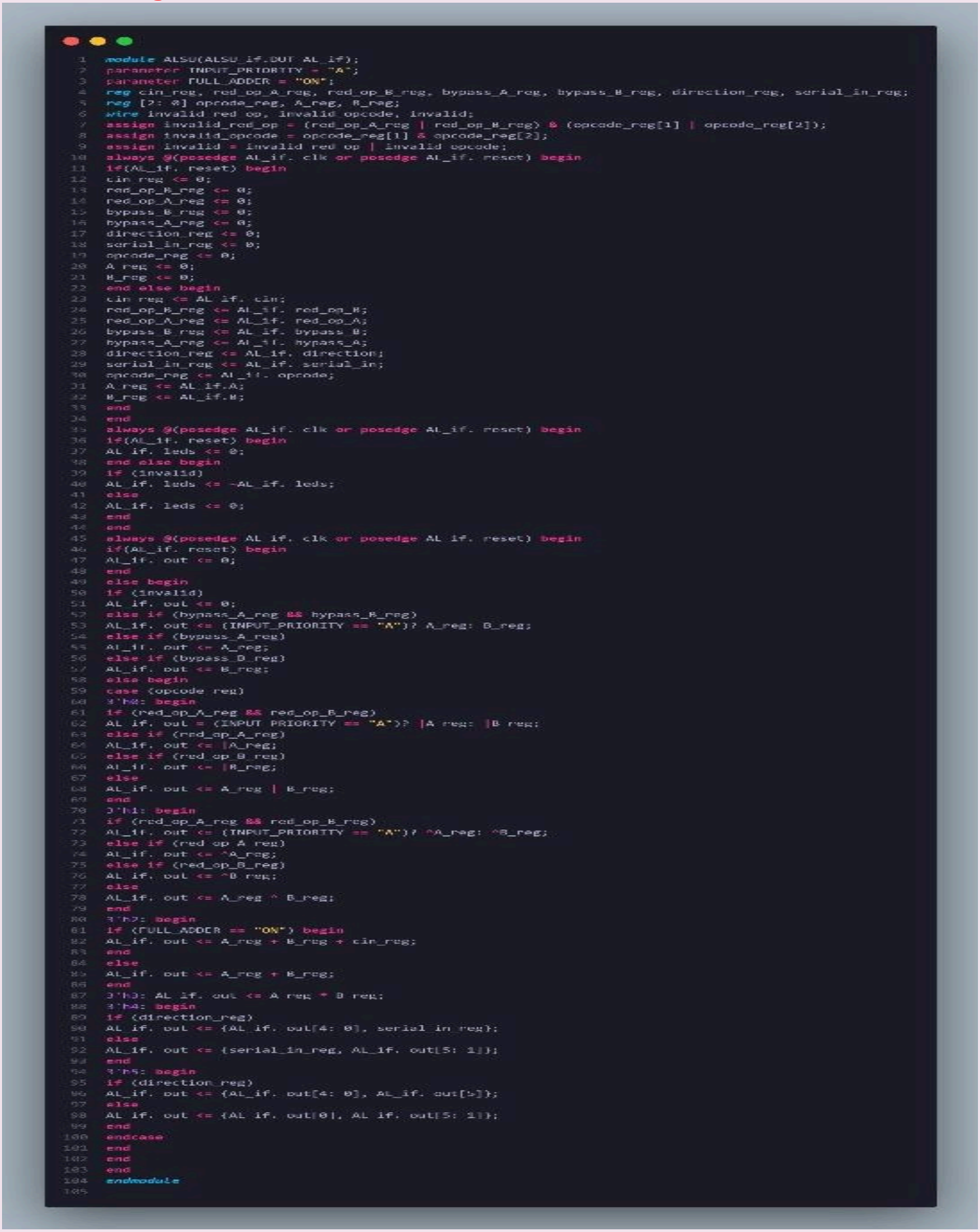
MARIAM MOHAMED MARZOUK

ASSIGNMENT_6

I. RTL design



```
module ALSU(ALSU_if.DUT AL_if);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";
reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg [2: 0] opcode_reg, A_reg, B_reg;
wire invalid_red_op, invalid_opcode, invalid;
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;
always @(posedge AL_if. clk or posedge AL_if. reset) begin
if(AL_if. reset) 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 <= AL_if. cin;
red_op_B_reg <= AL_if. red_op_B;
red_op_A_reg <= AL_if. red_op_A;
bypass_B_reg <= AL_if. bypass_B;
bypass_A_reg <= AL_if. bypass_A;
direction_reg <= AL_if. direction;
serial_in_reg <= AL_if. serial_in;
opcode_reg <= AL_if. opcode;
A_reg <= AL_if.A;
B_reg <= AL_if.B;
end
end
always @(posedge AL_if. clk or posedge AL_if. reset) begin
if(AL_if. reset) begin
AL_if. leds <= 0;
end else begin
if (invalid)
AL_if. leds <= ~AL_if. leds;
else
AL_if. leds <= 0;
end
end
always @(posedge AL_if. clk or posedge AL_if. reset) begin
if(AL_if. reset) begin
AL_if. out <= 0;
end
else begin
if (invalid)
AL_if. out <= 0;
else if (bypass_A_reg && bypass_B_reg)
AL_if. out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
else if (bypass_A_reg)
AL_if. out <= A_reg;
else if (bypass_B_reg)
AL_if. out <= B_reg;
else begin
case (opcode_reg)
3'b0: begin
if (red_op_A_reg && red_op_B_reg)
AL_if. out = (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
else if (red_op_A_reg)
AL_if. out <= |A_reg;
else if (red_op_B_reg)
AL_if. out <= |B_reg;
else
AL_if. out <= A_reg | B_reg;
end
3'b1: begin
if (red_op_A_reg && red_op_B_reg)
AL_if. out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
else if (red_op_A_reg)
AL_if. out <= ^A_reg;
else if (red_op_B_reg)
AL_if. out <= ^B_reg;
else
AL_if. out <= A_reg ^ B_reg;
end
3'b2: begin
if (FULL_ADDER == "ON") begin
AL_if. out <= A_reg + B_reg + cin_reg;
end
else
AL_if. out <= A_reg + B_reg;
end
3'b3: AL_if. out <= A_reg * B_reg;
3'b4: begin
if (direction_reg)
AL_if. out <= {AL_if. out[4: 0], serial_in_reg};
else
AL_if. out <= {serial_in_reg, AL_if. out[5: 1]};
end
3'b5: begin
if (direction_reg)
AL_if. out <= {AL_if. out[4: 0], AL_if. out[5]};
else
AL_if. out <= {AL_if. out[0], AL_if. out[5: 1]};
end
endcase
end
end
end
endmodule
```

II. Top module

```
import pack_test::*;;
import uvm_pkg::*;
`include "uvm_macros.svh"
module TOP ();
bit clk;
initial begin
clk = 0;
forever begin
#1 clk = ~clk;
end
end
ALSU_if al_if (clk);
ALSU DUT (al_if);
ALSU_Gold GOLD(al_if);
bind ALSU SVA assertions (al_if.DUT);
initial begin
uvm_config_db#(virtual ALSU_if):: set(null,"uvm_test_top","ALSU_if", al_if);
run_test("test");
end
endmodule
```

III. Interface module

```
interface ALSU_if (clk);
input bit clk;
logic reset, 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 [15: 0] leds, leds_G;
logic [5: 0] out;
logic [5: 0] out_G;
modport DUT (input clk, reset, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in, opcode,A, B, output out, leds);
modport DUT_GOLD (input clk, reset, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in, opcode,A, B, output out_G, leds_G);
endinterface
```

IV. Golden Model

```
module ALSU_Gold (ALSU_if.DUT_GOLD AL_if);
parameter INPUT_PRIORIT = "A";
parameter FULL_ADDER = "ON";
reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
reg [2: 0] opcode_reg, A_reg, B_reg;
always @(posedge AL_if. clk or posedge AL_if. reset) begin
if(AL_if. reset) 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 <= AL_if. cin;
red_op_B_reg <= AL_if. red_op_B;
red_op_A_reg <= AL_if. red_op_A;
bypass_B_reg <= AL_if. bypass_B;
bypass_A_reg <= AL_if. bypass_A;
direction_reg <= AL_if. direction;
serial_in_reg <= AL_if. serial_in;
opcode_reg <= AL_if. opcode;
A_reg <= AL_if. A;
B_reg <= AL_if. B;
end
end
always @(posedge AL_if. clk or posedge AL_if. reset) begin
if(AL_if. reset)begin
AL_if. out_G <= 6'h00;
AL_if. leds_G <= 16'h0000;
end
else begin
case (opcode_reg)
3'b000: begin
if (red_op_A_reg && red_op_B_reg)
AL_if. out_G <= |A_reg;
else if (red_op_A_reg)
AL_if. out_G <= |A_reg;
else if (red_op_B_reg)
AL_if. out_G <= |B_reg;
else
AL_if. out_G <= A_reg|B_reg;
if (bypass_A_reg && bypass_B_reg ) begin
AL_if. out_G <= A_reg;
end
else if (bypass_A_reg) begin
AL_if. out_G <= A_reg;
end
```

```
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
end
3'b001: begin
if (red_op_A_reg && red_op_B_reg)
ALU_if.out_G <= ^A_reg;
else if (red_op_A_reg)
ALU_if.out_G <= ^A_reg;
else if (red_op_B_reg)
ALU_if.out_G <= ^B_reg;
else
ALU_if.out_G <= A_reg^B_reg;
if (bypass_A_reg && bypass_B_reg ) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_A_reg) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
end
3'b010: begin
if (red_op_A_reg || red_op_B_reg)begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
else if (bypass_A_reg && bypass_B_reg ) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_A_reg) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
else begin
ALU_if.out_G <= A_reg + B_reg + cin_reg;
end
end
3'b011: begin
if (red_op_A_reg || red_op_B_reg) begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
else if (bypass_A_reg && bypass_B_reg ) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_A_reg) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
else begin
ALU_if.out_G <= A_reg * B_reg;
end
end
3'b100: begin
if (red_op_A_reg || red_op_B_reg)begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
else if (bypass_A_reg && bypass_B_reg ) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_A_reg) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
else begin
if (direction_reg) begin
ALU_if.out_G <= {ALU_if.out_G[4:0], serial_in_reg};
end
else
ALU_if.out_G <= {serial_in_reg, ALU_if.out_G[5:1]};
end
end
3'b101: begin
if (red_op_A_reg || red_op_B_reg)begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
else if (bypass_A_reg && bypass_B_reg ) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_A_reg) begin
ALU_if.out_G <= A_reg;
end
else if (bypass_B_reg) begin
ALU_if.out_G <= B_reg;
end
else begin
if (direction_reg) begin
ALU_if.out_G <= {ALU_if.out_G[4:0], ALU_if.out_G[5]};
end
else
ALU_if.out_G <= {ALU_if.out_G[0], ALU_if.out_G[5:1]};
end
end
3'b110: begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
3'b111: begin
ALU_if.out_G = 0;
ALU_if.leds_G = ~ALU_if.leds_G;
end
endcase
end
end
endmodule //ALU
```

V. Sequence item

```
package ALSU_seq_item_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"
    typedef enum {OR, XOR, ADD, MULT, SHIFT, ROTATE, INVALID_6, INVALID_7 } Opcode_e;
    localparam MAXPOS=3, MAXNEG=-4, ZERO=0;
    class ALSU_seq_item extends uvm_sequence_item;
        `uvm_object_utils(ALSU_seq_item);
        rand logic signed [2:0] A, B;
        rand Opcode_e opcode;
        rand logic rst;
        rand logic cin,red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
        logic signed [5:0] out;
        logic signed [5:0] out_golden;
        logic [15:0] leds;
        logic [15:0] leds_golden;
        function  new(string name="ALSU_seq_item");
            super.new(name);
        endfunction : new
        virtual function string convert2string();
            return
            $sformatf(" %s  reset=%0b , A=%0b , B=%0b, opcode=%0b , cin=%0h red_op_A=%0b red_op_A=%0b bypass_A=%0b bypass_B=0%b direction=%0b serial_in=%0b",
            super.convert2string(),rst, A, B, opcode, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in);
        endfunction: convert2string
        virtual function string convert2string_stimulus();
            return
            $sformatf("reset=%0b , A=%0b , B=%0b, opcode=%0b , cin=%0h red_op_A=%0b red_op_A=%0b, bypass_A=%0b bypass_B=0%b direction=%0b serial_in=%0b",
            rst, A,B, opcode, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in);
        endfunction: convert2string_stimulus
        //ALSU_4
        constraint reset {
            rst dist {0:=99 , 1:=1};
        }
        constraint ADD_MULT {
            if( opcode==MULT || opcode==ADD )
            {
                A dist { MAXPOS:=30 , ZERO:=30 , MAXNEG:=30 , [-MAXPOS:-1]:=10, [1:2]:=10 };
                B dist { MAXPOS:=30 , ZERO:=30 , MAXNEG:=30 , [-MAXPOS:-1]:=10, [1:2]:=10 };}}
            constraint OR_XOR {
                if( ( opcode==OR || opcode==XOR) && red_op_A )
                {
                    A dist { 2:=90, -4:=90, [-3:0]:=10, 3:=30 };
                    B==ZERO;
                }
                    if( ( opcode==OR || opcode==XOR) && red_op_B )
                    {
                        B dist{ 2:=90, -4:=90 , [-3:0]:=10, 3:=30     };
                        A==ZERO;
                    }}
            //ALSU_3
            constraint Opcode_Invalid_Cases{
                opcode dist { [0:5]:=90 , [6:7]:=10 };
            }
            //ALSU_4
            constraint Bypass_A_B{
                    bypass_B dist {ZERO:=80, 1:=20};
                    bypass_A dist {ZERO:=80, 1:=20};
            }
        endclass : ALSU_seq_item
    endpackage : ALSU_seq_item_pkg
```

VI. Sequence

```
package pack_seq;
`include "uvm_macros.svh"
import uvm_pkg::*;
import pack_seq_item::*;
class shift_reg_reset_sequence extends uvm_sequence #(shift_reg_seq_item);
`uvm_object_utils(shift_reg_reset_sequence);
shift_reg_seq_item seq_item;
function new(string name = "shift_reg_reset_sequence");
super. new(name);
endfunction
task body;
seq_item = shift_reg_seq_item::type_id:: create("seq_item");
start_item(seq_item);
seq_item. reset = 1;
seq_item. serial_in = 0;
seq_item. red_op_B = 0;
seq_item. red_op_A = 0;
seq_item. bypass_A = 0;
seq_item. bypass_B = 0;
seq_item. cin = 0;
seq_item. direction = 0;
seq_item.A = 0;
seq_item. B = 0;
seq_item. opcode = 0;
finish_item(seq_item);
endtask
endclass
class shift_reg_main_sequence extends uvm_sequence #(shift_reg_seq_item);
`uvm_object_utils(shift_reg_main_sequence);
shift_reg_seq_item seq_item;
function new(string name = "shift_reg_main_sequence");
super. new(name);
endfunction
task body;
repeat(1000) begin
seq_item = shift_reg_seq_item::type_id:: create("seq_item");
start_item(seq_item);
assert (seq_item. randomize());
finish_item(seq_item);
end
endtask
endclass
class direct_test_sequence extends uvm_sequence #(shift_reg_seq_item);
`uvm_object_utils(direct_test_sequence);
shift_reg_seq_item seq_item;
function new(string name = "direct_test_sequence");
super. new(name);
endfunction
task body;
seq_item = shift_reg_seq_item::type_id:: create("seq_item");
start_item(seq_item);
seq_item. A = 50;
seq_item. B = 50;
#1;
seq_item. opcode = 3'b000;
#2;
seq_item. opcode = 3'b001;
#2;
seq_item. opcode = 3'b010;
#2;
seq_item. opcode = 3'b011;
#2;
seq_item. opcode = 3'b100;
#2;
seq_item. opcode = 3'b101;
#2;
finish_item(seq_item);
endtask
endclass
endpackage

```

VII. Scoreboard

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

  class scoreboard extends uvm_scoreboard;
    `uvm_component_utils(scoreboard)

    uvm_analysis_export #(shift_reg_seq_item) sb_export;
    uvm_tlm_analysis_fifo #(shift_reg_seq_item) sb_fifo;
    shift_reg_seq_item seq_item_sb;
    int error_count = 0;
    int correct_count = 0;

    function new(string name = "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

    task run_phase(uvm_phase phase);
      super.run_phase(phase);
      forever begin
        sb_fifo.get(seq_item_sb);
        if (seq_item_sb.out != seq_item_sb.out_G) begin
          `uvm_error("run_phase", $sformatf("comparison failed, transaction received by the DUT: %s", seq_item_sb.convert2string()));
          error_count++;
        end else begin
          `uvm_info("run_phase", $sformatf("correct output: %s", seq_item_sb.convert2string()), UVM_LOW);
          correct_count++;
        end
      end
    endtask

    function void report_phase(uvm_phase phase);
      super.report_phase(phase);
      `uvm_info("report_phase", $sformatf("Total successful counts:
", correct_count), UVM_MEDIUM); $sformatf("Total failed counts:
", error_count), UVM_MEDIUM);
    endfunction
  endclass
endpackage
```

VIII. Coverage collector

```
package sh_coverage;
`include "uvm_macros.svh"
import uvm_pkg::*;
import pack_seq_item::*;
class ccoverage extends uvm_component;
`uvm_component_utils(ccoverage);
uvm_analysis_export #(shift_reg_seq_item) cov_export;
uvm_tlm_analysis_fifo #(shift_reg_seq_item) cov_fifo;
shift_reg_seq_item seq_item_cov;
covergroup cvg;
A_cp: coverpoint seq_item_cov.A {
      bins data_0 = {ZERO};
      bins data_max = {MAXPOS};
      bins data_min = {MAXNEG};
      bins data_walkingones = {3'b001, 3'b010, 3'b100};
      bins data_default = default;}
    B_cp: coverpoint seq_item_cov.B {
      bins data_0 = {ZERO};
      bins data_max = {MAXPOS};
      bins data_min = {MAXNEG};
      bins data_walkingones = {3'b001, 3'b010, 3'b100};
      bins data_default = default;}
    ALU_cp: coverpoint seq_item_cov.opcode{
      bins Bins_shift[] = {SHIFT, ROTATE};
      bins Bins_arith[] = {ADD, MULT};
      bins Bins_bitwise[] = {OR, XOR};
      illegal_bins Bins_invalid = {INVALID_6, INVALID_7};
    }
    red_A_cp: coverpoint seq_item_cov.red_op_A {
      bins Bins_red_A = {0, 1};}
    red_B_cp: coverpoint seq_item_cov.red_op_B{
      bins Bins_red_B = {0, 1};}
    cin_cp: coverpoint seq_item_cov.cin {
      bins Bins_Cin = {0, 1};}
    serial_in_cp: coverpoint seq_item_cov.serial_in {
      bins Bins_serial_in = {0, 1};}
    direction_cp: coverpoint seq_item_cov.direction {
      bins Bins_Direction = {0, 1};}
    ALU_cross1: cross A_cp, B_cp, ALU_cp {
      bins cross_ab_arith = binsof(ALU_cp.Bins_arith) && binsof(A_cp) intersect {ZERO, MAXNEG, MAXPOS}
        && binsof(B_cp) intersect {ZERO, MAXNEG, MAXPOS};
    ignore_bins ignore_ab_arith=! binsof(ALU_cp.Bins_arith) && binsof(A_cp) intersect {ZERO, MAXNEG, MAXPOS}&& binsof(B_cp) intersect {ZERO, MAXNEG, MAXPOS} ;
   }
    ALU_cross2: cross ALU_cp, cin_cp {
      bins cross_cin_arith = binsof(ALU_cp.Bins_arith) && binsof(cin_cp.Bins_Cin);}
    ALU_cross3: cross ALU_cp, serial_in_cp, direction_cp {
    bins cross_serial_shift = binsof(ALU_cp.Bins_shift) && binsof(serial_in_cp.Bins_serial_in);
    bins cross_shift_direction = binsof(ALU_cp.Bins_shift) && binsof(direction_cp.Bins_Direction);
    ignore_bins ignore_serial_shift = !binsof(ALU_cp.Bins_shift);
    ignore_bins ignore_shift_direction = !binsof(ALU_cp.Bins_shift);}
    ALU_cross4: cross A_cp, B_cp, ALU_cp, red_A_cp, red_B_cp {
      bins cross_bitwise_redA = binsof(ALU_cp.Bins_bitwise) && binsof(red_A_cp.Bins_red_A[1])
        && binsof(A_cp.data_walkingones) && binsof(B_cp.data_0);
      bins cross_bitwise_redB = binsof(ALU_cp.Bins_bitwise) && binsof(red_B_cp.Bins_red_B[1])
        && binsof(B_cp.data_walkingones) && binsof(A_cp.data_0);
      illegal_bins cross_red_A_not_bitwise
        = binsof(ALU_cp) intersect {ADD, MULT, SHIFT, ROTATE} && binsof(red_A_cp.Bins_red_A[1]);
      illegal_bins cross_red_B_not_bitwise
        = binsof(ALU_cp) intersect {ADD, MULT, SHIFT, ROTATE} && binsof(red_B_cp.Bins_red_B[1]);}
endgroup
function new(string name = "ccoverage", uvm_component parent = null);
super.new(name, parent);
cvg = 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);
cvg.sample();
end
endtask
endclass
endpackage
```

IX. SVA Module

```
module ALSU_assertions (
    input logic signed [2:0] A, B,
    input logic clk, rst, cin, serial_in, red_op_A, red_op_B,
    input logic [2:0] opcode,
    input logic bypass_A, bypass_B, direction,
    input logic [15:0] leds,
    input logic signed [5:0] out
);
    logic invalid;
    assign invalid = (opcode == 7 || opcode == 6 || ((opcode != 1 && opcode != 0) && (red_op_B || red_op_A)));

    // Reset check
    always_comb if (rst) assert final(out == 0 && leds == 0);

    // Combined properties for operations
    property check_operations;
        @(posedge clk) disable iff (rst)
        (invalid || bypass_A || bypass_B || red_op_A || red_op_B || opcode == 0 || opcode == 1) |->
        ##2 (
            (bypass_A ? out == A :
            bypass_B ? out == B :
            red_op_A ? out == |A :
            red_op_B ? out == |B :
            opcode == 0 ? out == (A | B) :
            opcode == 1 ? out == (A ^ B) :
            opcode == 2 ? out[2:0] == ($past(A, 2) + $past(B, 2) + $past(cin, 2)) :
            opcode == 3 ? out == A * B :
            opcode == 4 ? out == {direction ? $past(out[4:0]) : serial_in, $past(out[5:1])} :  // Shift/Rotate
            out == {serial_in, $past(out[4:0])}  // Rotate
            )
        );
    endproperty

    // Assertions and coverage for operations
    assert_operations_ap: assert property(check_operations);
    cover_operations_cp: cover property(check_operations);

    // Invalid opcode check
    property invalid_opcode;
        @(posedge clk) disable iff (rst) (opcode == 6 || opcode == 7) |-> ##2 (out == 0 && leds == ~ $past(leds));
    endproperty

    // Assertion and coverage for invalid opcode
    assert_invalid_ap: assert property(invalid_opcode);
    cover_invalid_cp: cover property(invalid_opcode);
endmodule : ALSU_assertions


```

X. Driver module

```
package pack_driver;
`include "uvm_macros.svh"
import uvm_pkg::*;
import pack_object::*;
import pack_sequencer::*;
import pack_seq_item::*;
class DRIVER extends uvm_driver #(shift_reg_seq_item );
`uvm_component_utils(DRIVER);
virtual ALSU_if sh_vif;
shift_reg_seq_item stim_seq_item;
function new(string name = "DRIVER", uvm_component parent = null);
super. new(name, parent);
endfunction //new()
task run_phase (uvm_phase phase);
super. run_phase(phase);
forever begin
stim_seq_item = shift_reg_seq_item::type_id:: create("stim_seq_item");
seq_item_port. get_next_item(stim_seq_item);
sh_vif. reset = stim_seq_item. reset;
sh_vif. direction = stim_seq_item. direction;
sh_vif. cin = stim_seq_item. cin;
sh_vif. serial_in = stim_seq_item. serial_in;
sh_vif. bypass_A = stim_seq_item. bypass_A;
sh_vif. bypass_B = stim_seq_item. bypass_B;
sh_vif. red_op_A = stim_seq_item. red_op_A;
sh_vif. red_op_B = stim_seq_item. red_op_B;
sh_vif. A = stim_seq_item. A;
sh_vif. B = stim_seq_item. B;
sh_vif. opcode = stim_seq_item. opcode;
@(negedge sh_vif. clk);
stim_seq_item. out = sh_vif. out;
stim_seq_item. leds = sh_vif. leds;
stim_seq_item. out_G = sh_vif. out_G;
stim_seq_item. leds_G = sh_vif. leds_G;
 seq_item_port. item_done();
`uvm_info("run_phase", stim_seq_item. convert2string_stimulus(),UVM_HIGH);
end
endtask
endclass
endpackage
```

XI. Agent module

```
package pack_agent;
`include "uvm_macros.svh"
import uvm_pkg::*;
import pack_driver::*;
import pack_sequencer::*;
import pack_seq_item::*;
import pack_mon::*;
import pack_object::*;
class sh_agent extends uvm_agent;
`uvm_component_utils(sh_agent);
object alsu_config_obj_test;
DRIVER alsu_dr;
MYSequencer sqr;
MONITOR mon;
uvm_analysis_port #(shift_reg_seq_item) agt_ap;
function new(string name = "sh_agent", uvm_component parent = null);
super. new(name, parent);
endfunction //new()
function void build_phase (uvm_phase phase);
super. build_phase(phase);
if (!uvm_config_db #(object):: get(this,"","CGO", alsu_config_obj_test)) begin
`uvm_fatal("build_phase","DRIVER unable to get the virtual interface");
end
if (alsu_config_obj_test.sel_mod == UVM_ACTIVE) begin
alsu_dr = DRIVER::type_id:: create("alsu_dr",this);
sqr = MYSequencer::type_id:: create("sqr",this);
end
mon = MONITOR::type_id:: create("mon",this);
agt_ap = new("agt_ap",this);
endfunction
function void connect_phase(uvm_phase phase);
super. connect_phase(phase);
if (alsu_config_obj_test. sel_mod == UVM_ACTIVE) begin
alsu_dr.seq_item_port.connect(sqr. seq_item_export);
alsu_dr.sh_vif = alsu_config_obj_test.alsu_config_vif;
end
mon.sh_vif = alsu_config_obj_test. alsu_config_vif;
mon. mon_ap. connect(agt_ap);
endfunction
endclass
endpackage
```

XII. Environment module

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

XIII. Monitor

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

XIV. Sequencer

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

XV. Configuration

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

XVI. Test bench

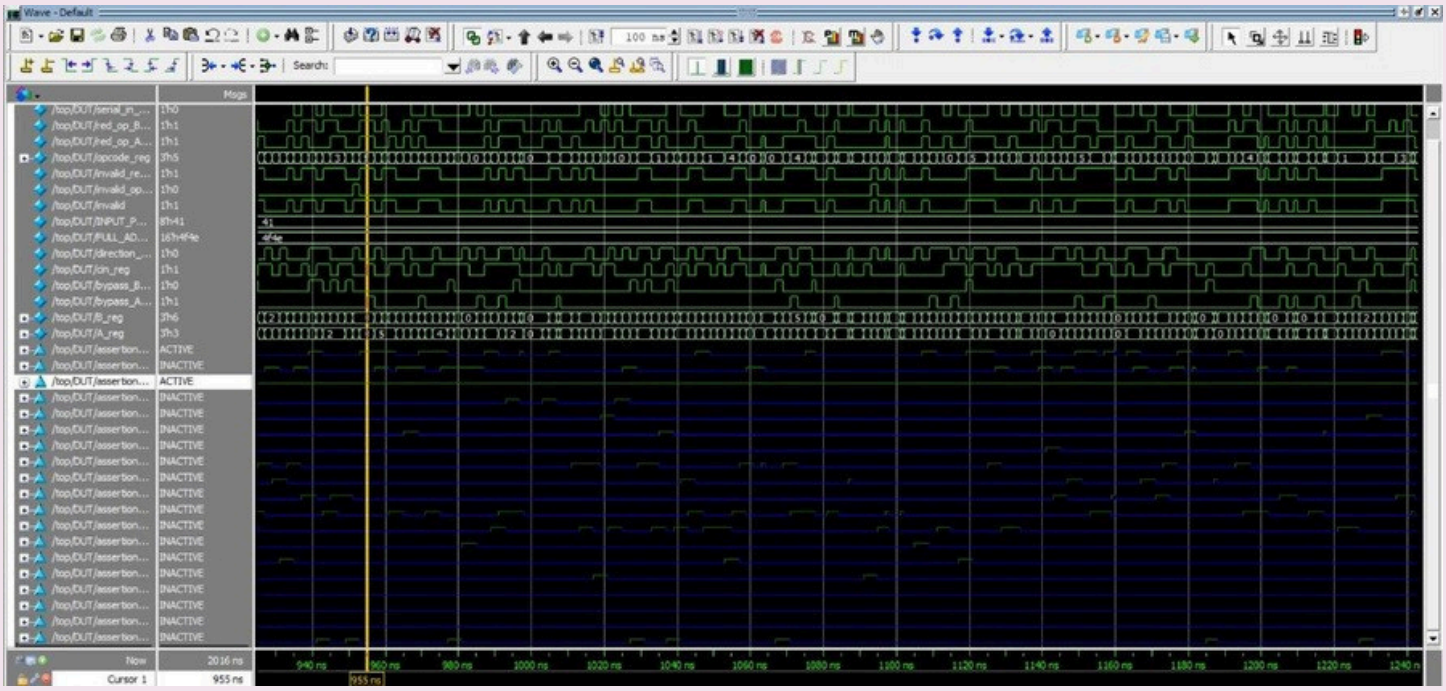
```
package pack_test;
`include "uvm_macros.svh"
import uvm_pkg::*;
import pack_env::*;
import pack_object::*;
import pack_seq::*;
class test extends uvm_test;
`uvm_component_utils(test)
alsu_env env;
object alsu_config_obj_test;
shift_reg_main_sequence main_seq;
shift_reg_reset_sequence reset_seq;
direct_test_sequence dirc_seq;
function new (string name = "test", uvm_component parent = null);
super . new(name, parent);
endfunction //new()
function void build_phase (uvm_phase phase);
super . build_phase(phase);
env = alsu_env :: type_id :: create("env",this);
alsu_config_obj_test = object :: type_id :: create ("alsu_config_obj_test");
main_seq = shift_reg_main_sequence::type_id:: create("main_seq");
reset_seq = shift_reg_reset_sequence::type_id:: create("reset_seq");
dirc_seq = direct_test_sequence::type_id:: create("dirc_seq");
if(! uvm_config_db #(virtual ALSU_if):: get(this,"","ALSU_if", alsu_config_obj_test. alsu_config_vif))begin
`uvm_fatal("build_phase","the test unable to get the virtual interface");
end
alsu_config_obj_test. sel_mod = UVM_ACTIVE;
uvm_config_db #(object) :: set (this,"*","CGO", alsu_config_obj_test);
endfunction
task run_phase (uvm_phase phase);
super . run_phase (phase);
phase . raise_objection(this);
//reset sequence
`uvm_info("run_phase","reset asserted",UVM_LOW);
reset_seq. start(env. agt. sqr);
`uvm_info("run_phase","reset deasserted",UVM_LOW);
//main sequence
`uvm_info("run_phase","stimulus generated started",UVM_LOW);
main_seq. start(env. agt. sqr);
`uvm_info("run_phase","stimulus generated ended",UVM_LOW);
//direct test sequence
`uvm_info("run_phase","stimulus generated started",UVM_LOW);
dirc_seq. start(env. agt. sqr);
`uvm_info("run_phase","stimulus generated ended",UVM_LOW);
phase . drop_objection(this);
endtask //run_phase
endclass //test
endpackage
```

XVII. Print

```
--- UVM Report Summary ---

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

XVIII. Waveform



XIX. Functional Coverage report

```
Coverage Report by instance with details

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

Directive Coverage:
    Directives                      19       19        0   100.00%

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

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

Covergroup Coverage:
    Covergroups                      1       na       na   100.00%
        Coverpoints/Crosses         12       na       na        na
            Covergroup Bins         71       71        0   100.00%
-----------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

-----------------------------------------------------------------------------------------------------
 TYPE /sh_coverage/ccoverage/cvg                   100.00%        100          -    Covered
    covered/total bins:                                 71         71          -
    missing/total bins:                                  0         71          -
    % Hit:                                          100.00%        100          -
    Coverpoint A_cp                                 100.00%        100          -    Covered
        covered/total bins:                               3          3          -
        missing/total bins:                               0          3          -
        % Hit:                                      100.00%        100          -
        bin data_0                                      188          1          -    Covered
        bin data_max                                    157          1          -    Covered
        bin data_walkingones                            354          1          -    Covered
        default bin data_default                        309                     -    Occurred
    Coverpoint B_cp                                 100.00%        100          -    Covered
        covered/total bins:                               3          3          -
        missing/total bins:                               0          3          -
        % Hit:                                      100.00%        100          -
        bin data_0                                      240          1          -    Covered
        bin data_max                                    154          1          -    Covered
        bin data_walkingones                            318          1          -    Covered
        default bin data_default                        296                     -    Occurred
    Coverpoint ALU_cp                               100.00%        100          -    Covered
        covered/total bins:                               6          6          -
        missing/total bins:                               0          6          -
        % Hit:                                      100.00%        100          -
        illegal_bin Bins_invalid                         32                     -    Occurred
        bin Bins_shift[4]                               170          1          -    Covered
        bin Bins_shift[5]                               173          1          -    Covered
        bin Bins_arith[2]                               156          1          -    Covered
        bin Bins_arith[3]                               137          1          -    Covered
        bin Bins_bitwise[0]                             172          1          -    Covered
        bin Bins_bitwise[1]                             168          1          -    Covered
    Coverpoint red_A_cp                             100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
        bin Bins_red_A                                 1008          1          -    Covered
    Coverpoint red_B_cp                             100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
        bin Bins_red_B                                 1008          1          -    Covered
    Coverpoint cin_cp                               100.00%        100          -    Covered
        covered/total bins:                               1          1          -
        missing/total bins:                               0          1          -
        % Hit:                                      100.00%        100          -
        bin Bins_Cin                                   1008          1          -    Covered
```

```
    % Hit:                                 100.00%        100          -
    bin Bins_red_A                            1008          1          -    Covered
Coverpoint red_B_cp                        100.00%        100          -    Covered
    covered/total bins:                           1          1          -
    missing/total bins:                           0          1          -
    % Hit:                                 100.00%        100          -
    bin Bins_red_B                            1008          1          -    Covered
Coverpoint cin_cp                          100.00%        100          -    Covered
    covered/total bins:                           1          1          -
    missing/total bins:                           0          1          -
    % Hit:                                 100.00%        100          -
    bin Bins_Cin                              1008          1          -    Covered
Coverpoint serial_in_cp                    100.00%        100          -    Covered
    covered/total bins:                           1          1          -
    missing/total bins:                           0          1          -
    % Hit:                                 100.00%        100          -
    bin Bins_serial_in                        1008          1          -    Covered
Coverpoint direction_cp                    100.00%        100          -    Covered
    covered/total bins:                           1          1          -
    missing/total bins:                           0          1          -
    % Hit:                                 100.00%        100          -
    bin Bins_Direction                        1008          1          -    Covered
Cross ALU_cross1                           100.00%        100          -    Covered
    covered/total bins:                          31         31          -
    missing/total bins:                           0         31          -
    % Hit:                                 100.00%        100          -
    Auto, Default and User Defined Bins:
        bin cross_ab_arith                     100          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_bitwise[1]>
                                                21          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_bitwise[0]>
                                                21          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_arith[3]>
                                                 3          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_shift[5]>
                                                26          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_arith[2]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_shift[4]>
                                                26          1          -    Covered
        bin <data_max,data_walkingones,Bins_bitwise[1]>
                                                 3          1          -    Covered
        bin <data_0,data_walkingones,Bins_bitwise[1]>
                                                15          1          -    Covered
        bin <data_max,data_walkingones,Bins_bitwise[0]>
                                                 2          1          -    Covered
        bin <data_0,data_walkingones,Bins_bitwise[0]>
                                                19          1          -    Covered
        bin <data_max,data_walkingones,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_0,data_walkingones,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_max,data_walkingones,Bins_shift[5]>
                                                 7          1          -    Covered
        bin <data_0,data_walkingones,Bins_shift[5]>
                                                10          1          -    Covered
        bin <data_max,data_walkingones,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_0,data_walkingones,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_max,data_walkingones,Bins_shift[4]>
                                                10          1          -    Covered
        bin <data_0,data_walkingones,Bins_shift[4]>
                                                 9          1          -    Covered
        bin <data_walkingones,data_max,Bins_bitwise[1]>
                                                 3          1          -    Covered
        bin <data_walkingones,data_0,Bins_bitwise[1]>
                                                49          1          -    Covered
        bin <data_walkingones,data_max,Bins_bitwise[0]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_0,Bins_bitwise[0]>
                                                33          1          -    Covered
        bin <data_walkingones,data_max,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_arith[3]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_max,Bins_shift[5]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_shift[5]>
                                                12          1          -    Covered
        bin <data_walkingones,data_max,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_arith[2]>
                                                 9          1          -    Covered
        bin <data_walkingones,data_max,Bins_shift[4]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_0,Bins_shift[4]>
                                                 6          1          -    Covered
    Illegal and Ignore Bins:
        ignore_bin ignore_ab_arith              38                     -    Occurred
Cross ALU_cross2                           100.00%        100          -    Covered
    covered/total bins:                           5          5          -
    missing/total bins:                           0          5          -
    % Hit:                                 100.00%        100          -
    Auto, Default and User Defined Bins:
        bin cross_cin_arith                    293          1          -    Covered
```

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

COVERGROUP COVERAGE:
----------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

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

```
    % Hit:                                100.00%        100          -
    bin Bins_red_B                           1008          1          -    Covered
Coverpoint cin_cp                         100.00%        100          -    Covered
    covered/total bins:                          1          1          -
    missing/total bins:                          0          1          -
    % Hit:                                100.00%        100          -
    bin Bins_Cin                             1008          1          -    Covered
Coverpoint serial_in_cp                   100.00%        100          -    Covered
    covered/total bins:                          1          1          -
    missing/total bins:                          0          1          -
    % Hit:                                100.00%        100          -
    bin Bins_serial_in                       1008          1          -    Covered
Coverpoint direction_cp                   100.00%        100          -    Covered
    covered/total bins:                          1          1          -
    missing/total bins:                          0          1          -
    % Hit:                                100.00%        100          -
    bin Bins_Direction                       1008          1          -    Covered
Cross ALU_cross1                          100.00%        100          -    Covered
    covered/total bins:                         31         31          -
    missing/total bins:                          0         31          -
    % Hit:                                100.00%        100          -
    Auto, Default and User Defined Bins:
        bin cross_ab_arith                     100          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_bitwise[1]>
                                                21          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_bitwise[0]>
                                                21          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_arith[3]>
                                                 3          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_shift[5]>
                                                26          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_arith[2]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_walkingones,Bins_shift[4]>
                                                26          1          -    Covered
        bin <data_max,data_walkingones,Bins_bitwise[1]>
                                                 3          1          -    Covered
        bin <data_0,data_walkingones,Bins_bitwise[1]>
                                                15          1          -    Covered
        bin <data_max,data_walkingones,Bins_bitwise[0]>
                                                 2          1          -    Covered
        bin <data_0,data_walkingones,Bins_bitwise[0]>
                                                19          1          -    Covered
        bin <data_max,data_walkingones,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_0,data_walkingones,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_max,data_walkingones,Bins_shift[5]>
                                                 7          1          -    Covered
        bin <data_0,data_walkingones,Bins_shift[5]>
                                                10          1          -    Covered
        bin <data_max,data_walkingones,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_0,data_walkingones,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_max,data_walkingones,Bins_shift[4]>
                                                10          1          -    Covered
        bin <data_0,data_walkingones,Bins_shift[4]>
                                                 9          1          -    Covered
        bin <data_walkingones,data_max,Bins_bitwise[1]>
                                                 3          1          -    Covered
        bin <data_walkingones,data_0,Bins_bitwise[1]>
                                                49          1          -    Covered
        bin <data_walkingones,data_max,Bins_bitwise[0]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_0,Bins_bitwise[0]>
                                                33          1          -    Covered
        bin <data_walkingones,data_max,Bins_arith[3]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_arith[3]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_max,Bins_shift[5]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_shift[5]>
                                                12          1          -    Covered
        bin <data_walkingones,data_max,Bins_arith[2]>
                                                 6          1          -    Covered
        bin <data_walkingones,data_0,Bins_arith[2]>
                                                 9          1          -    Covered
        bin <data_walkingones,data_max,Bins_shift[4]>
                                                 7          1          -    Covered
        bin <data_walkingones,data_0,Bins_shift[4]>
                                                 6          1          -    Covered
    Illegal and Ignore Bins:
        ignore_bin ignore_ab_arith              38                     -    Occurred
Cross ALU_cross2                          100.00%        100          -    Covered
    covered/total bins:                          5          5          -
    missing/total bins:                          0          5          -
    % Hit:                                100.00%        100          -
    Auto, Default and User Defined Bins:
        bin cross_cin_arith                    293          1          -    Covered
        bin <Bins_bitwise[1],Bins_Cin>         168          1          -    Covered
        bin <Bins_bitwise[0],Bins_Cin>         172          1          -    Covered
        bin <Bins_shift[5],Bins_Cin>           173          1          -    Covered
        bin <Bins_shift[4],Bins_Cin>           170          1          -    Covered
Cross ALU_cross3                          100.00%        100          -    Covered
```

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

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1

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

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 19

Total Coverage By Instance (filtered view): 100.00%
```