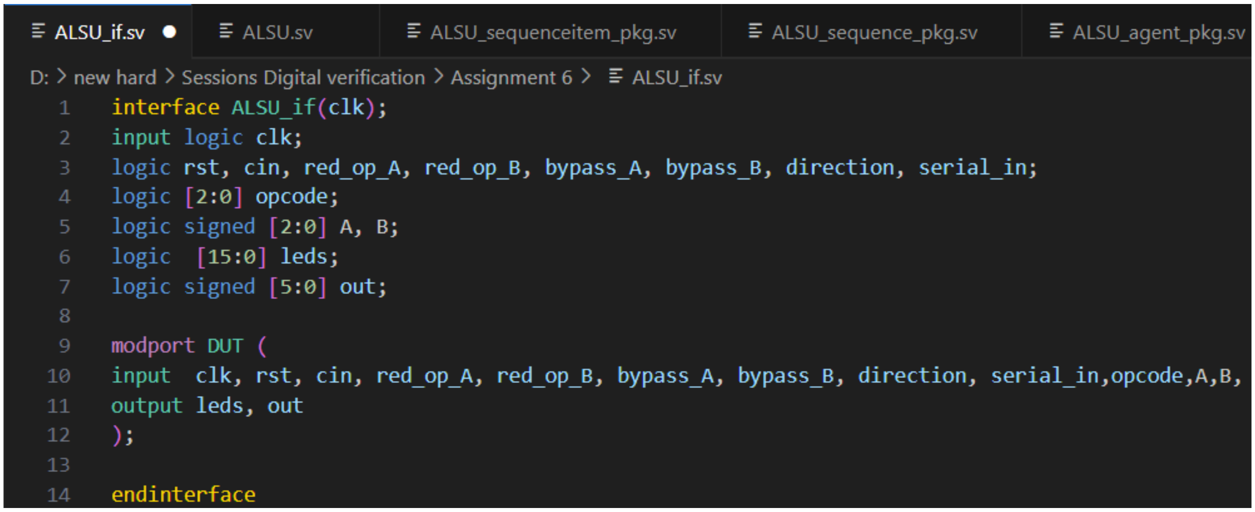
Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

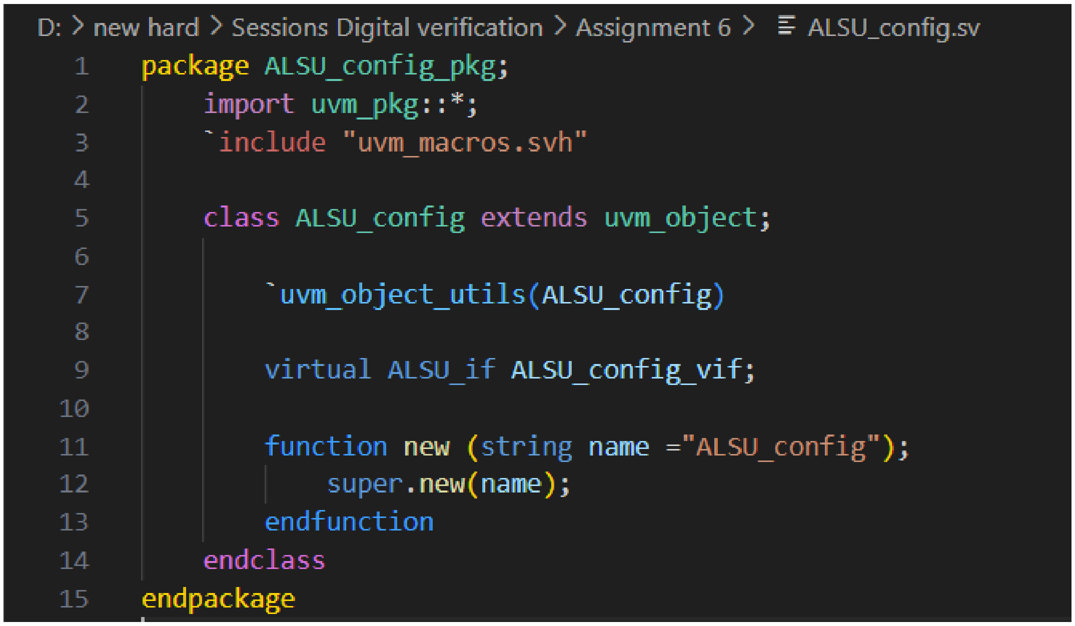
ALSU Design: -

Interface code:



```
interface ALSU_if(clk);
input logic 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  [15:0] leds;
logic signed [5:0] out;

modport DUT (
input  clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in,opcode,A,B,
output leds, out
);

endinterface
```

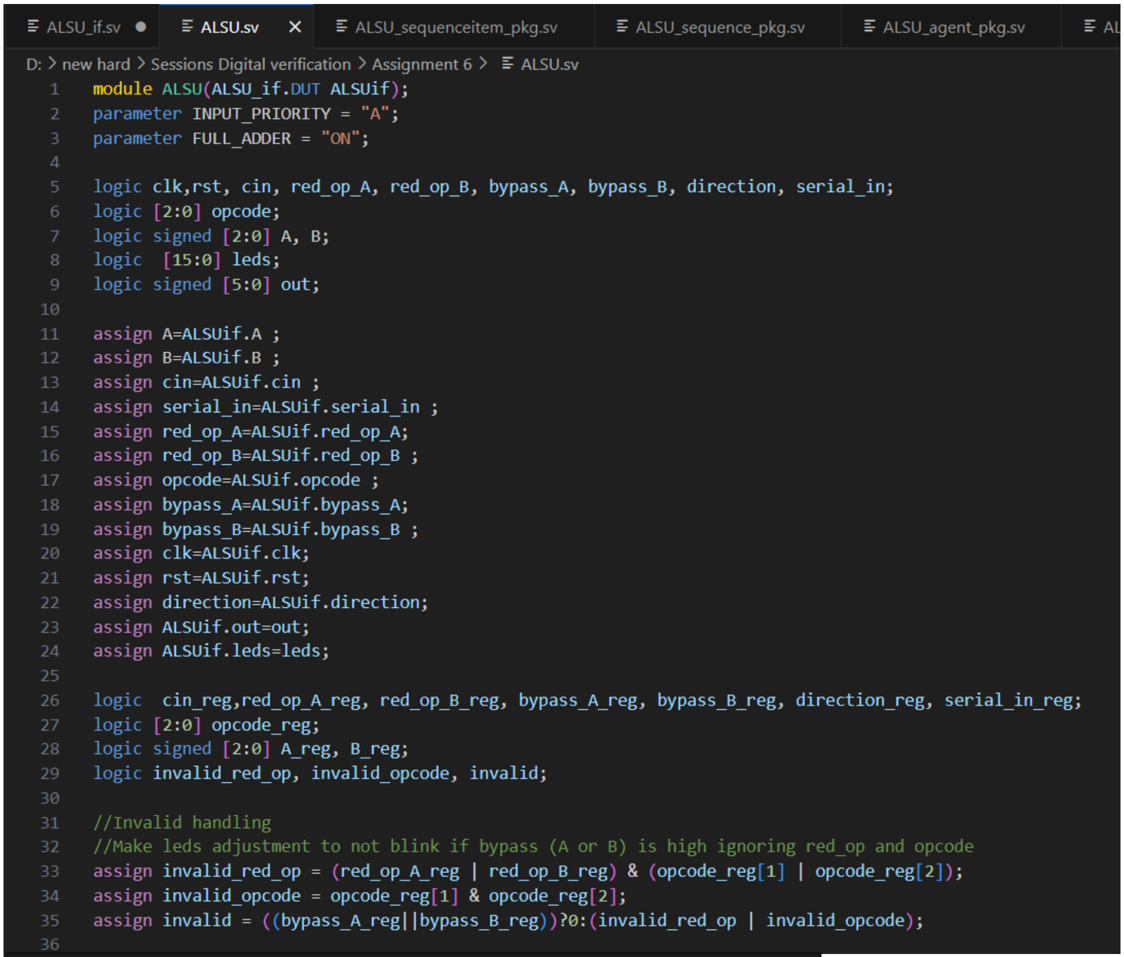
Configuration object code:


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_config.sv

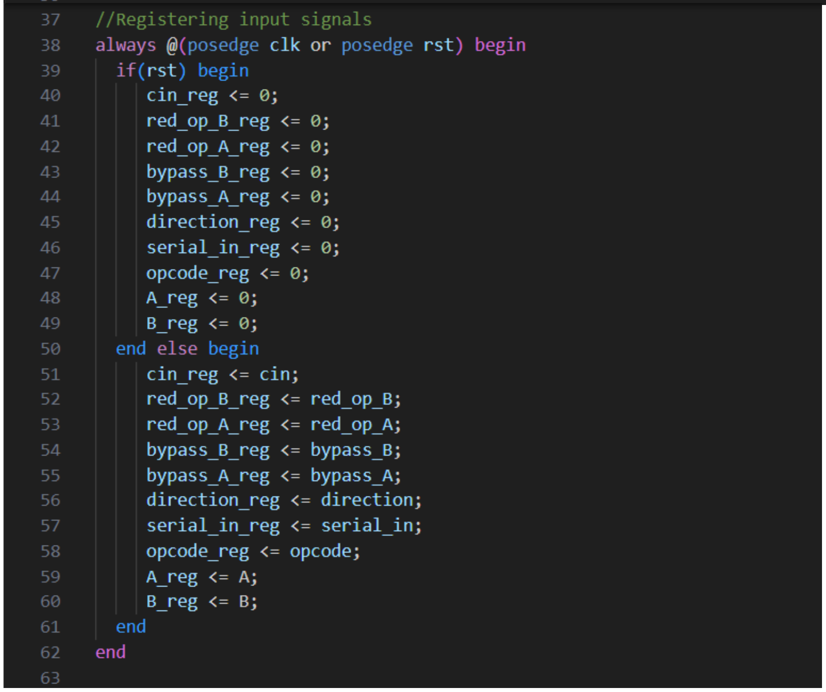
```
package ALSU_config_pkg;
    import uvm_pkg::*;
    `include "uvm_macros.svh"

    class ALSU_config extends uvm_object;

        `uvm_object_utils(ALSU_config)

        virtual ALSU_if ALSU_config_vif;

        function new (string name ="ALSU_config");
            super.new(name);
        endfunction
    endclass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Design code:

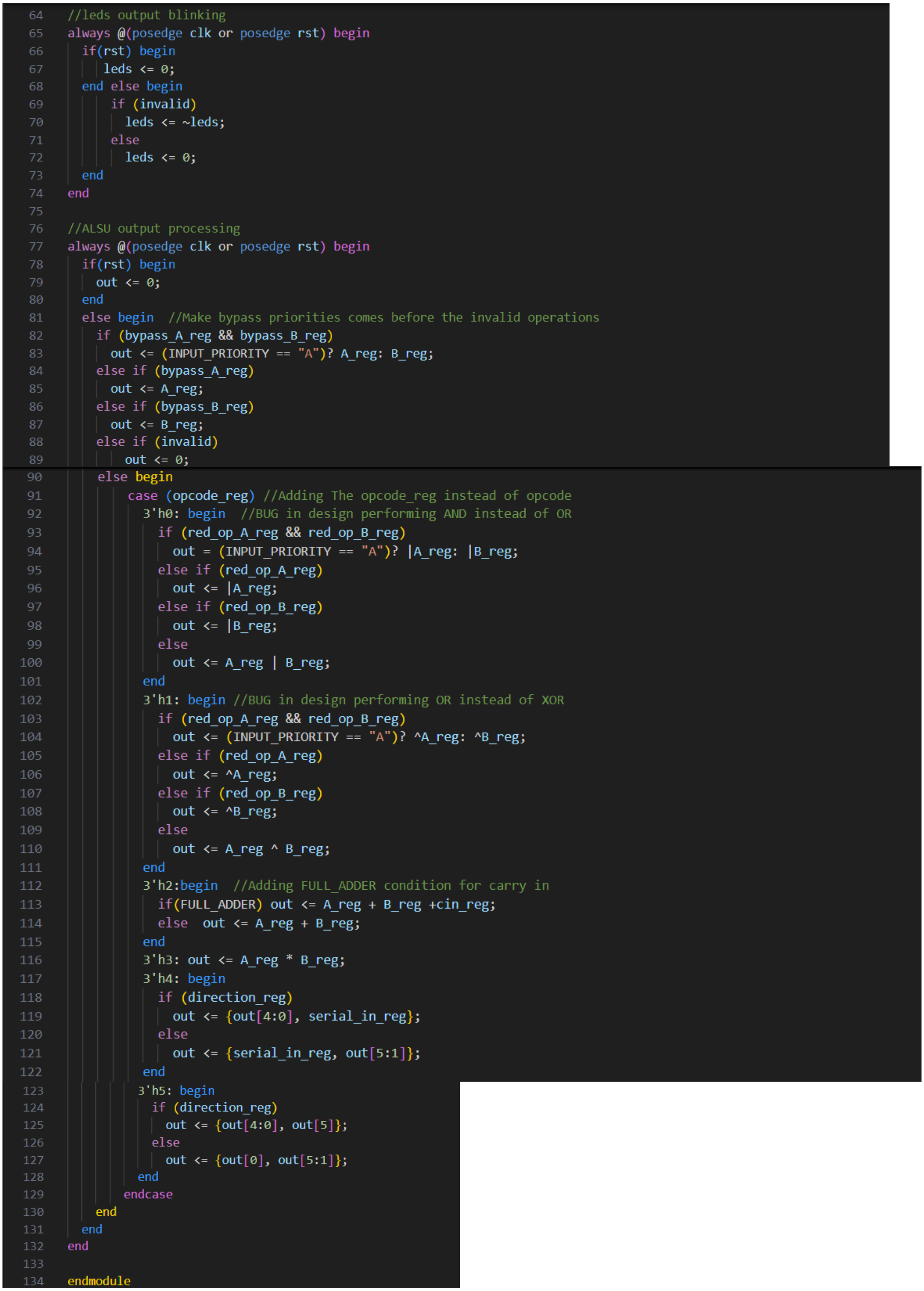


```
module ALSU(ALSU_if.DUT ALSUif);
parameter INPUT_PRIORITY = "A";
parameter FULL_ADDER = "ON";

logic clk,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  [15:0] leds;
logic signed [5:0] out;

assign A=ALSUif.A ;
assign B=ALSUif.B ;
assign cin=ALSUif.cin ;
assign serial_in=ALSUif.serial_in ;
assign red_op_A=ALSUif.red_op_A;
assign red_op_B=ALSUif.red_op_B ;
assign opcode=ALSUif.opcode ;
assign bypass_A=ALSUif.bypass_A;
assign bypass_B=ALSUif.bypass_B ;
assign clk=ALSUif.clk;
assign rst=ALSUif.rst;
assign direction=ALSUif.direction;
assign ALSUif.out=out;
assign ALSUif.leds=leds;

logic  cin_reg,red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
logic [2:0] opcode_reg;
logic signed [2:0] A_reg, B_reg;
logic invalid_red_op, invalid_opcode, invalid;

//Invalid handling
//Make leds adjustment to not blink if bypass (A or B) is high ignoring red_op and opcode
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 = ((bypass_A_reg||bypass_B_reg))?0:(invalid_red_op | invalid_opcode);

```



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

```

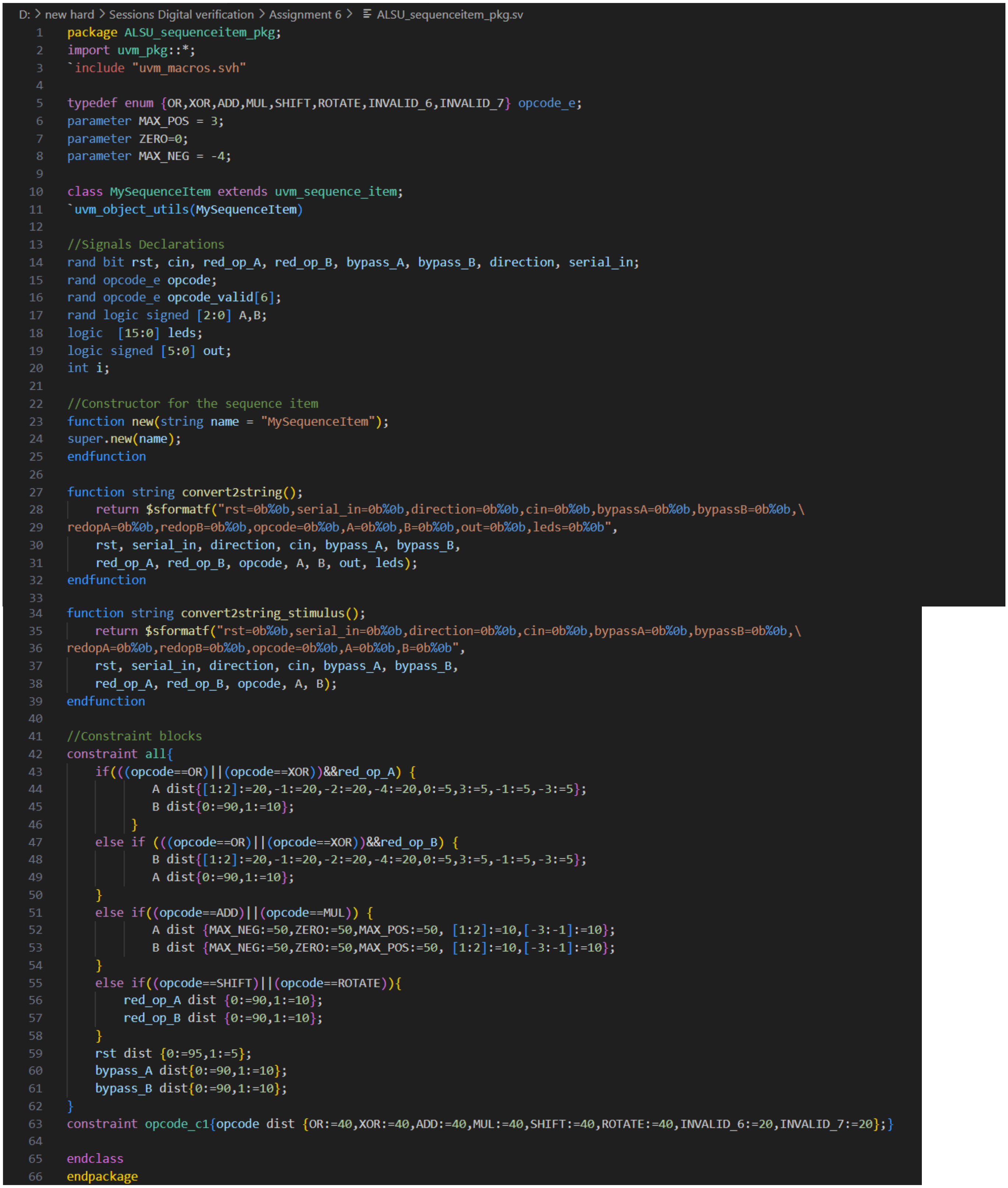
Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim



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

//ALSU output processing
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin  //Make bypass priorities comes before the invalid operations
    if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else if (invalid)
        out <= 0;
    else begin
        case (opcode_reg) //Adding The opcode_reg instead of opcode
          3'h0: begin  //BUG in design performing AND instead of OR
            if (red_op_A_reg && red_op_B_reg)
              out = (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
            else if (red_op_A_reg)
              out <= |A_reg;
            else if (red_op_B_reg)
              out <= |B_reg;
            else
              out <= A_reg | B_reg;
          end
          3'h1: begin //BUG in design performing OR instead of XOR
            if (red_op_A_reg && red_op_B_reg)
              out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
            else if (red_op_A_reg)
              out <= ^A_reg;
            else if (red_op_B_reg)
              out <= ^B_reg;
            else
              out <= A_reg ^ B_reg;
          end
          3'h2:begin  //Adding FULL_ADDER condition for carry in
            if(FULL_ADDER) out <= A_reg + B_reg +cin_reg;
            else  out <= A_reg + B_reg;
          end
          3'h3: out <= A_reg * B_reg;
          3'h4: begin
            if (direction_reg)
              out <= {out[4:0], serial_in_reg};
            else
              out <= {serial_in_reg, out[5:1]};
          end
          3'h5: begin
            if (direction_reg)
              out <= {out[4:0], out[5]};
            else
              out <= {out[0], out[5:1]};
          end
        endcase
    end
  end
end

endmodule
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Sequence Item Package code:

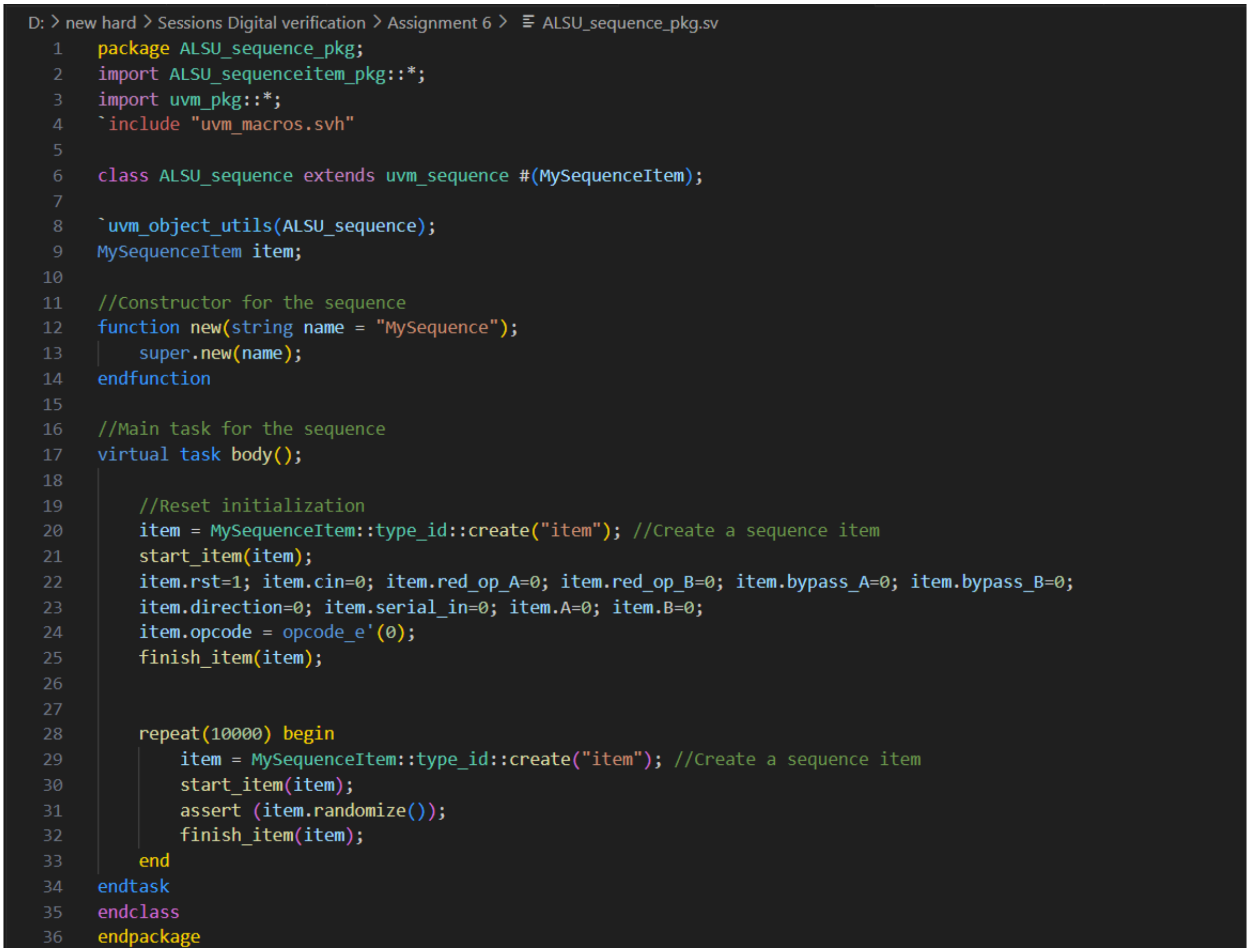


D: > new hard > Sessions Digital verification > Assignment 6 > ≡ ALSU_sequenceitem_pkg.sv

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

typedef enum {OR,XOR,ADD,MUL,SHIFT,ROTATE,INVALID_6,INVALID_7} opcode_e;
parameter MAX_POS = 3;
parameter ZERO=0;
parameter MAX_NEG = -4;

class MySequenceItem extends uvm_sequence_item;
`uvm_object_utils(MySequenceItem)

//Signals Declarations
rand bit rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
rand opcode_e opcode;
rand opcode_e opcode_valid[6];
rand logic signed [2:0] A,B;
logic  [15:0] leds;
logic signed [5:0] out;
int i;

//Constructor for the sequence item
function new(string name = "MySequenceItem");
super.new(name);
endfunction

function string convert2string();
    return $sformatf("rst=0b%0b,serial_in=0b%0b,direction=0b%0b,cin=0b%0b,bypassA=0b%0b,bypassB=0b%0b,\
redopA=0b%0b,redopB=0b%0b,opcode=0b%0b,A=0b%0b,B=0b%0b,out=0b%0b,leds=0b%0b",
    rst, serial_in, direction, cin, bypass_A, bypass_B,
    red_op_A, red_op_B, opcode, A, B, out, leds);
endfunction

function string convert2string_stimulus();
    return $sformatf("rst=0b%0b,serial_in=0b%0b,direction=0b%0b,cin=0b%0b,bypassA=0b%0b,bypassB=0b%0b,\
redopA=0b%0b,redopB=0b%0b,opcode=0b%0b,A=0b%0b,B=0b%0b",
    rst, serial_in, direction, cin, bypass_A, bypass_B,
    red_op_A, red_op_B, opcode, A, B);
endfunction

//Constraint blocks
constraint all{
    if(((opcode==OR)||(opcode==XOR))&&red_op_A) {
            A dist{[1:2]:=20,-1:=20,-2:=20,-4:=20,0:=5,3:=5,-1:=5,-3:=5};
            B dist{0:=90,1:=10};
         }
    else if (((opcode==OR)||(opcode==XOR))&&red_op_B) {
            B dist{[1:2]:=20,-1:=20,-2:=20,-4:=20,0:=5,3:=5,-1:=5,-3:=5};
            A dist{0:=90,1:=10};
    }
    else if((opcode==ADD)||(opcode==MUL)) {
            A dist {MAX_NEG:=50,ZERO:=50,MAX_POS:=50, [1:2]:=10,[-3:-1]:=10};
            B dist {MAX_NEG:=50,ZERO:=50,MAX_POS:=50, [1:2]:=10,[-3:-1]:=10};
    }
    else if((opcode==SHIFT)||(opcode==ROTATE)){
        red_op_A dist {0:=90,1:=10};
        red_op_B dist {0:=90,1:=10};
    }
    rst dist {0:=95,1:=5};
    bypass_A dist{0:=90,1:=10};
    bypass_B dist{0:=90,1:=10};
}
constraint opcode_c1{opcode dist {OR:=40,XOR:=40,ADD:=40,MUL:=40,SHIFT:=40,ROTATE:=40,INVALID_6:=20,INVALID_7:=20};}

endclass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

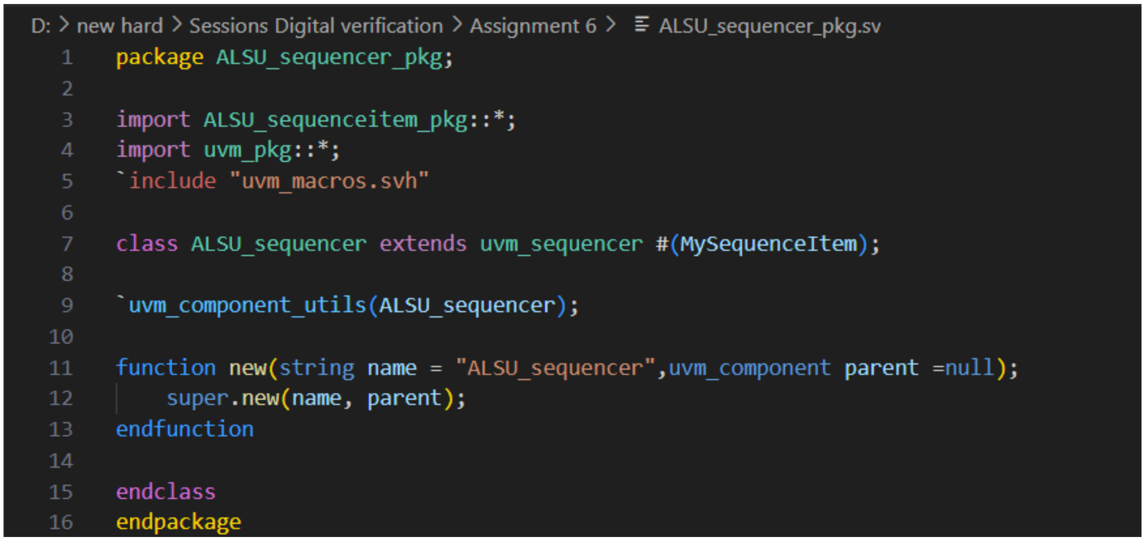
Sequence Package Code:



D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_sequence_pkg.sv

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

class ALSU_sequence extends uvm_sequence #(MySequenceItem);

`uvm_object_utils(ALSU_sequence);
MySequenceItem item;

//Constructor for the sequence
function new(string name = "MySequence");
    super.new(name);
endfunction

//Main task for the sequence
virtual task body();

    //Reset initialization
    item = MySequenceItem::type_id::create("item"); //Create a sequence item
    start_item(item);
    item.rst=1; item.cin=0; item.red_op_A=0; item.red_op_B=0; item.bypass_A=0; item.bypass_B=0;
    item.direction=0; item.serial_in=0; item.A=0; item.B=0;
    item.opcode = opcode_e'(0);
    finish_item(item);


    repeat(10000) begin
        item = MySequenceItem::type_id::create("item"); //Create a sequence item
        start_item(item);
        assert (item.randomize());
        finish_item(item);
    end
endtask
endclass
endpackage
```

Sequencer Package Code:

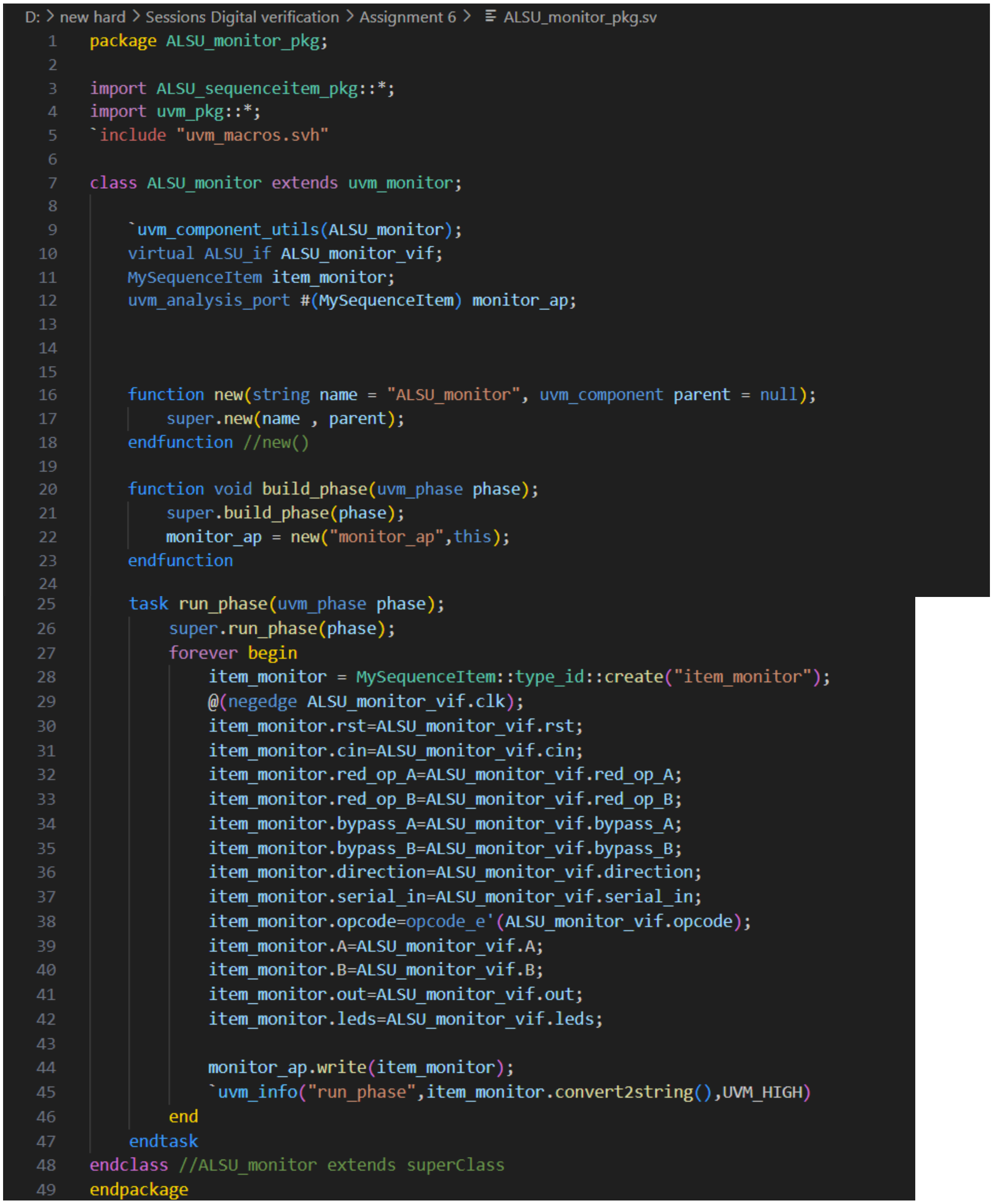


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_sequencer_pkg.sv

```
package ALSU_sequencer_pkg;

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

class ALSU_sequencer extends uvm_sequencer #(MySequenceItem);

`uvm_component_utils(ALSU_sequencer);

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

endclass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Monitor Package Code:

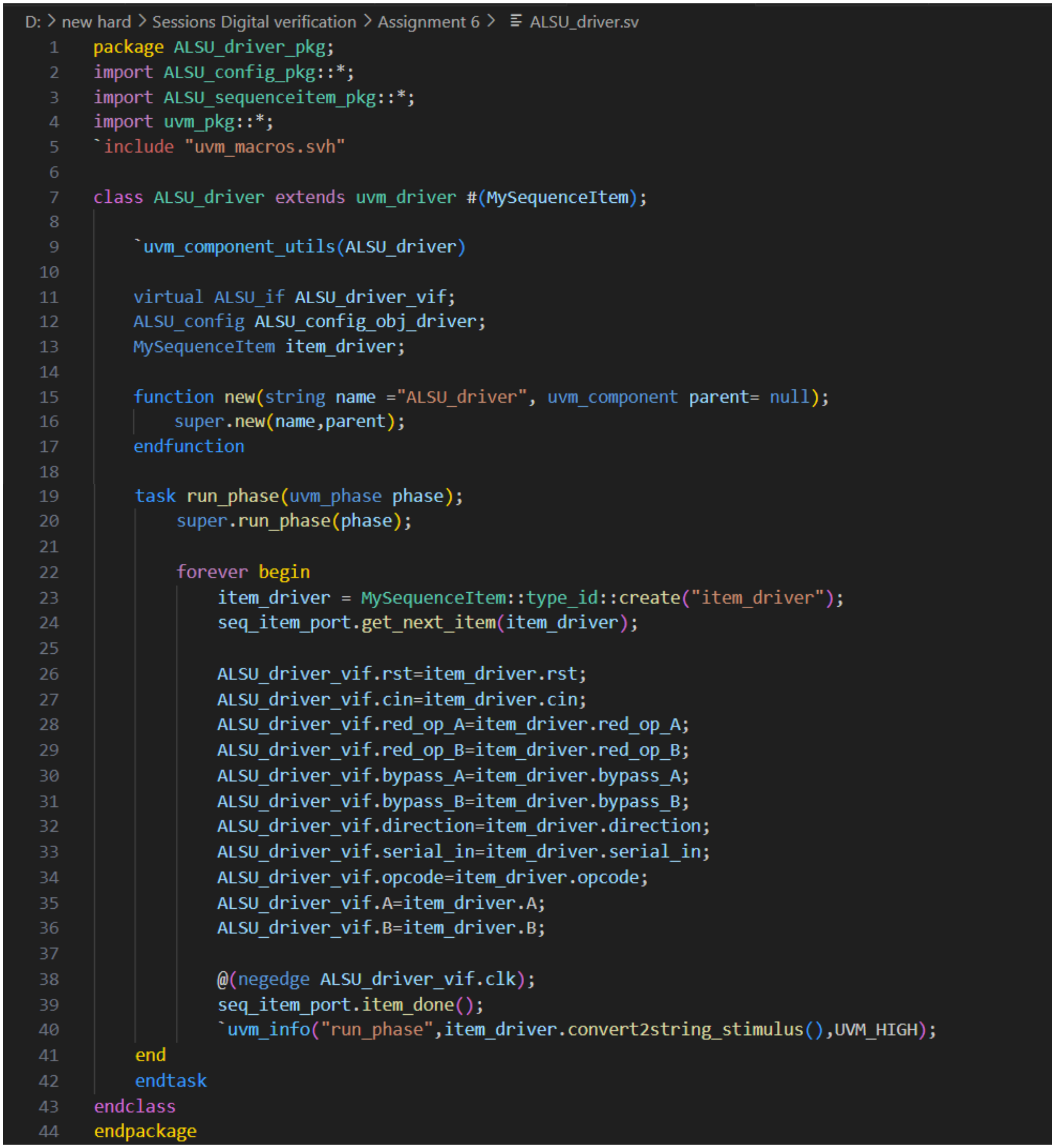


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_monitor_pkg.sv

```
package ALSU_monitor_pkg;

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

class ALSU_monitor extends uvm_monitor;

    `uvm_component_utils(ALSU_monitor);
    virtual ALSU_if ALSU_monitor_vif;
    MySequenceItem item_monitor;
    uvm_analysis_port #(MySequenceItem) monitor_ap;



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

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

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        forever begin
            item_monitor = MySequenceItem::type_id::create("item_monitor");
            @(negedge ALSU_monitor_vif.clk);
            item_monitor.rst=ALSU_monitor_vif.rst;
            item_monitor.cin=ALSU_monitor_vif.cin;
            item_monitor.red_op_A=ALSU_monitor_vif.red_op_A;
            item_monitor.red_op_B=ALSU_monitor_vif.red_op_B;
            item_monitor.bypass_A=ALSU_monitor_vif.bypass_A;
            item_monitor.bypass_B=ALSU_monitor_vif.bypass_B;
            item_monitor.direction=ALSU_monitor_vif.direction;
            item_monitor.serial_in=ALSU_monitor_vif.serial_in;
            item_monitor.opcode=opcode_e'(ALSU_monitor_vif.opcode);
            item_monitor.A=ALSU_monitor_vif.A;
            item_monitor.B=ALSU_monitor_vif.B;
            item_monitor.out=ALSU_monitor_vif.out;
            item_monitor.leds=ALSU_monitor_vif.leds;

            monitor_ap.write(item_monitor);
            `uvm_info("run_phase",item_monitor.convert2string(),UVM_HIGH)
        end
    endtask
endclass //ALSU_monitor extends superClass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Driver Package Code:

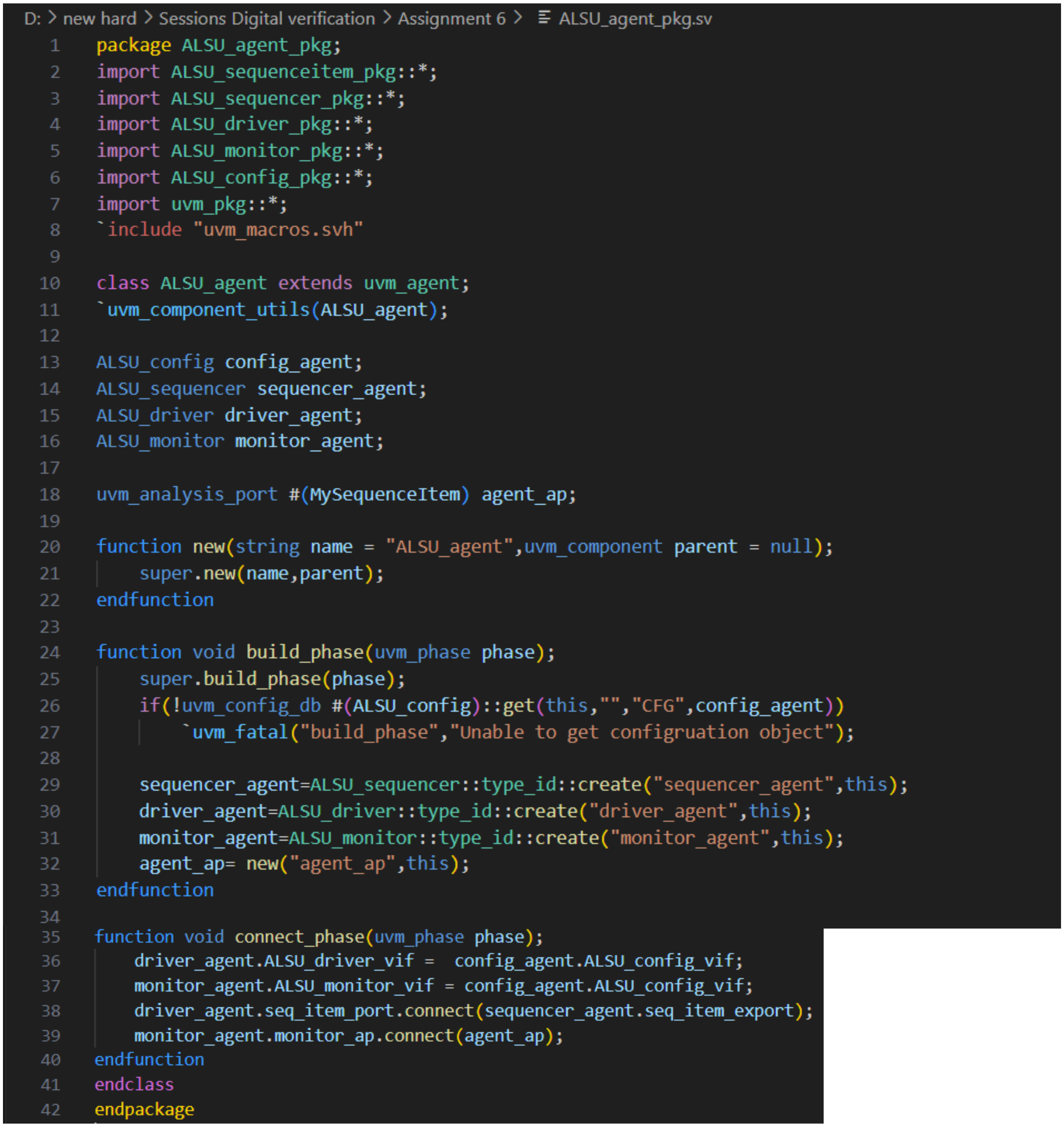


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_driver.sv

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

class ALSU_driver extends uvm_driver #(MySequenceItem);

    `uvm_component_utils(ALSU_driver)

    virtual ALSU_if ALSU_driver_vif;
    ALSU_config ALSU_config_obj_driver;
    MySequenceItem item_driver;

    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
            item_driver = MySequenceItem::type_id::create("item_driver");
            seq_item_port.get_next_item(item_driver);

            ALSU_driver_vif.rst=item_driver.rst;
            ALSU_driver_vif.cin=item_driver.cin;
            ALSU_driver_vif.red_op_A=item_driver.red_op_A;
            ALSU_driver_vif.red_op_B=item_driver.red_op_B;
            ALSU_driver_vif.bypass_A=item_driver.bypass_A;
            ALSU_driver_vif.bypass_B=item_driver.bypass_B;
            ALSU_driver_vif.direction=item_driver.direction;
            ALSU_driver_vif.serial_in=item_driver.serial_in;
            ALSU_driver_vif.opcode=item_driver.opcode;
            ALSU_driver_vif.A=item_driver.A;
            ALSU_driver_vif.B=item_driver.B;

            @(negedge ALSU_driver_vif.clk);
            seq_item_port.item_done();
            `uvm_info("run_phase",item_driver.convert2string_stimulus(),UVM_HIGH);
    end
    endtask
endclass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Agent Package Code:

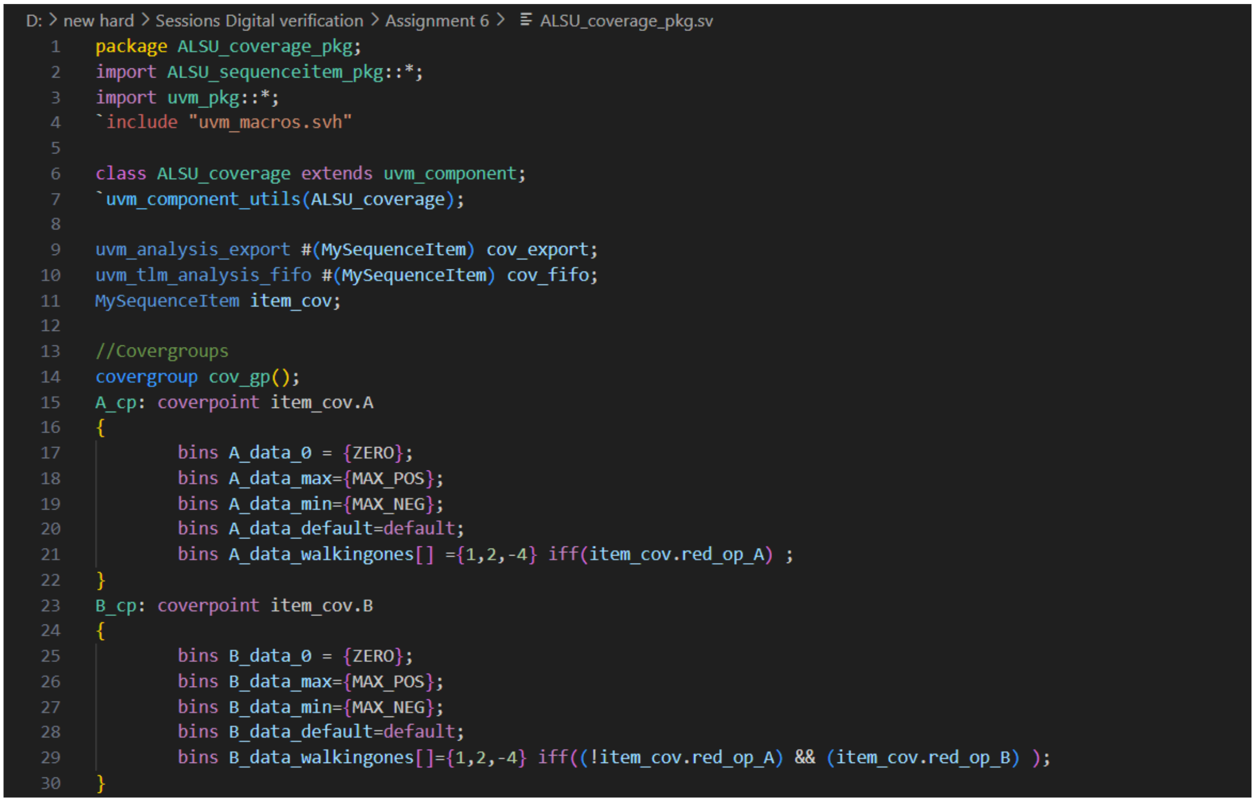


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_agent_pkg.sv

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

class ALSU_agent extends uvm_agent;
`uvm_component_utils(ALSU_agent);

ALSU_config config_agent;
ALSU_sequencer sequencer_agent;
ALSU_driver driver_agent;
ALSU_monitor monitor_agent;

uvm_analysis_port #(MySequenceItem) agent_ap;

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

function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db #(ALSU_config)::get(this,"","CFG",config_agent))
        `uvm_fatal("build_phase","Unable to get configruation object");

    sequencer_agent=ALSU_sequencer::type_id::create("sequencer_agent",this);
    driver_agent=ALSU_driver::type_id::create("driver_agent",this);
    monitor_agent=ALSU_monitor::type_id::create("monitor_agent",this);
    agent_ap= new("agent_ap",this);
endfunction

function void connect_phase(uvm_phase phase);
    driver_agent.ALSU_driver_vif =  config_agent.ALSU_config_vif;
    monitor_agent.ALSU_monitor_vif = config_agent.ALSU_config_vif;
    driver_agent.seq_item_port.connect(sequencer_agent.seq_item_export);
    monitor_agent.monitor_ap.connect(agent_ap);
endfunction
endclass
endpackage
```

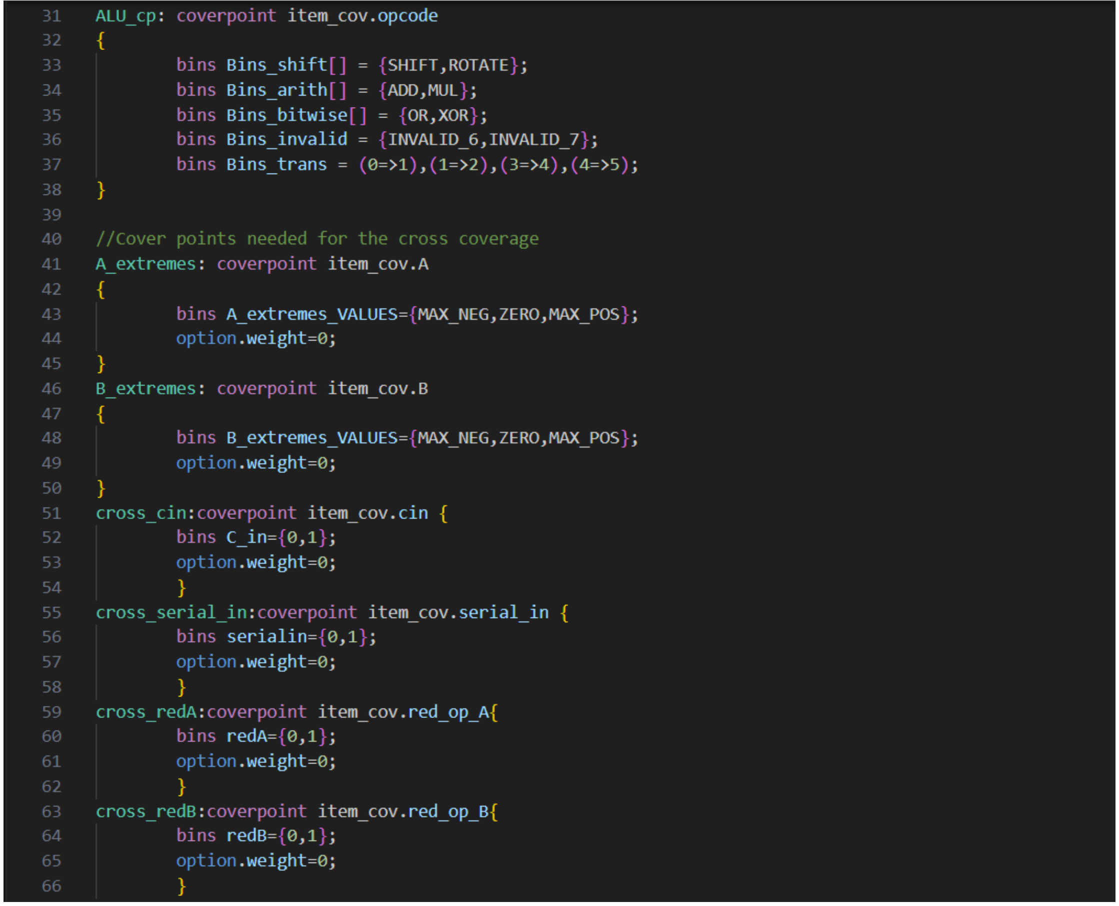
Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

Coverage Package Code:



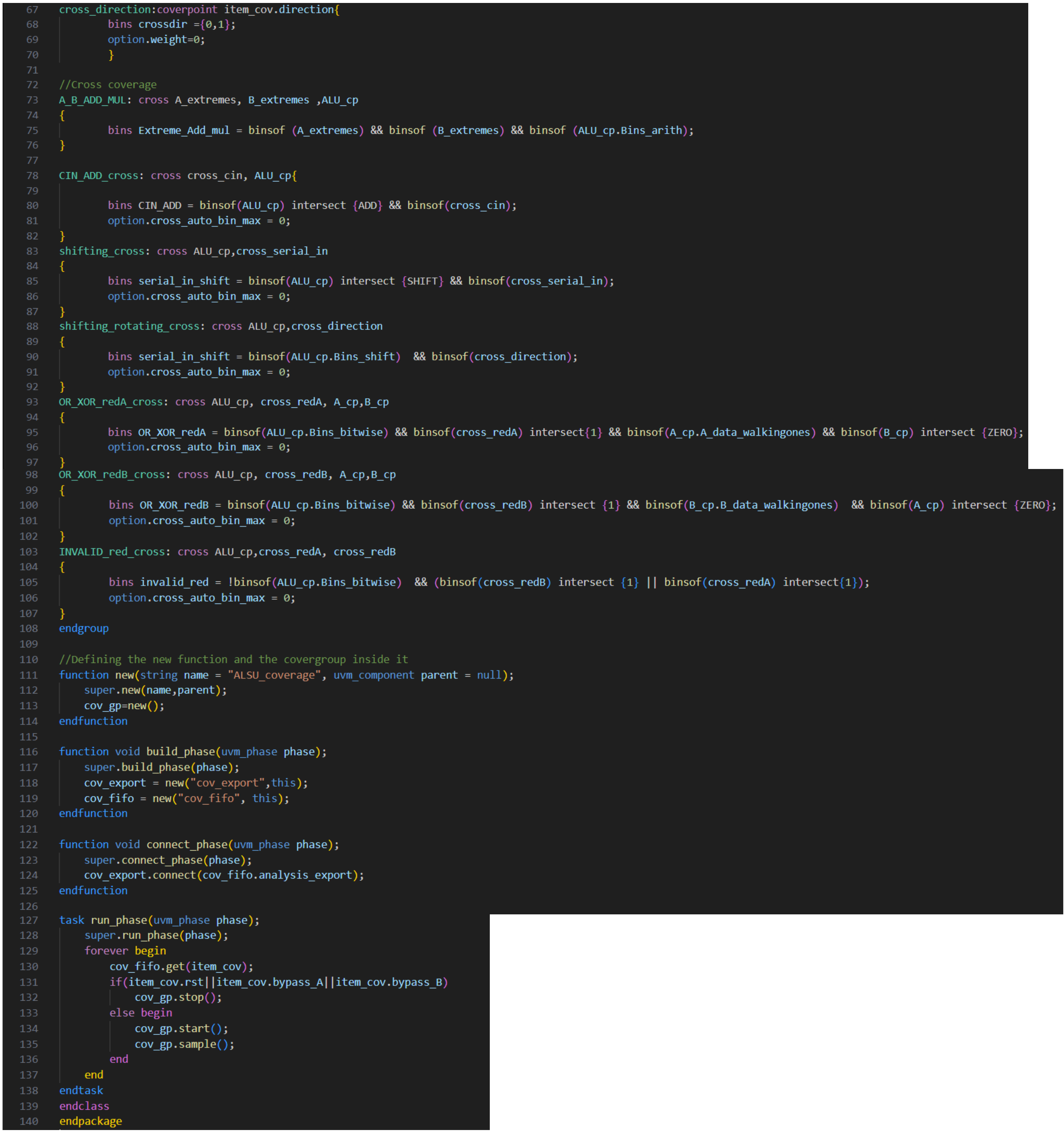
D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_coverage_pkg.sv

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

class ALSU_coverage extends uvm_component;
`uvm_component_utils(ALSU_coverage);

uvm_analysis_export #(MySequenceItem) cov_export;
uvm_tlm_analysis_fifo #(MySequenceItem) cov_fifo;
MySequenceItem item_cov;

//Covergroups
covergroup cov_gp();
A_cp: coverpoint item_cov.A
{
        bins A_data_0 = {ZERO};
        bins A_data_max={MAX_POS};
        bins A_data_min={MAX_NEG};
        bins A_data_default=default;
        bins A_data_walkingones[] ={1,2,-4} iff(item_cov.red_op_A) ;
}
B_cp: coverpoint item_cov.B
{
        bins B_data_0 = {ZERO};
        bins B_data_max={MAX_POS};
        bins B_data_min={MAX_NEG};
        bins B_data_default=default;
        bins B_data_walkingones[]={1,2,-4} iff((!item_cov.red_op_A) && (item_cov.red_op_B) );
}
```



```
ALU_cp: coverpoint item_cov.opcode
{
        bins Bins_shift[] = {SHIFT,ROTATE};
        bins Bins_arith[] = {ADD,MUL};
        bins Bins_bitwise[] = {OR,XOR};
        bins Bins_invalid = {INVALID_6,INVALID_7};
        bins Bins_trans = (0=>1),(1=>2),(3=>4),(4=>5);
}

//Cover points needed for the cross coverage
A_extremes: coverpoint item_cov.A
{
        bins A_extremes_VALUES={MAX_NEG,ZERO,MAX_POS};
        option.weight=0;
}
B_extremes: coverpoint item_cov.B
{
        bins B_extremes_VALUES={MAX_NEG,ZERO,MAX_POS};
        option.weight=0;
}
cross_cin:coverpoint item_cov.cin {
        bins C_in={0,1};
        option.weight=0;
        }
cross_serial_in:coverpoint item_cov.serial_in {
        bins serialin={0,1};
        option.weight=0;
        }
cross_redA:coverpoint item_cov.red_op_A{
        bins redA={0,1};
        option.weight=0;
        }
cross_redB:coverpoint item_cov.red_op_B{
        bins redB={0,1};
        option.weight=0;
        }
```

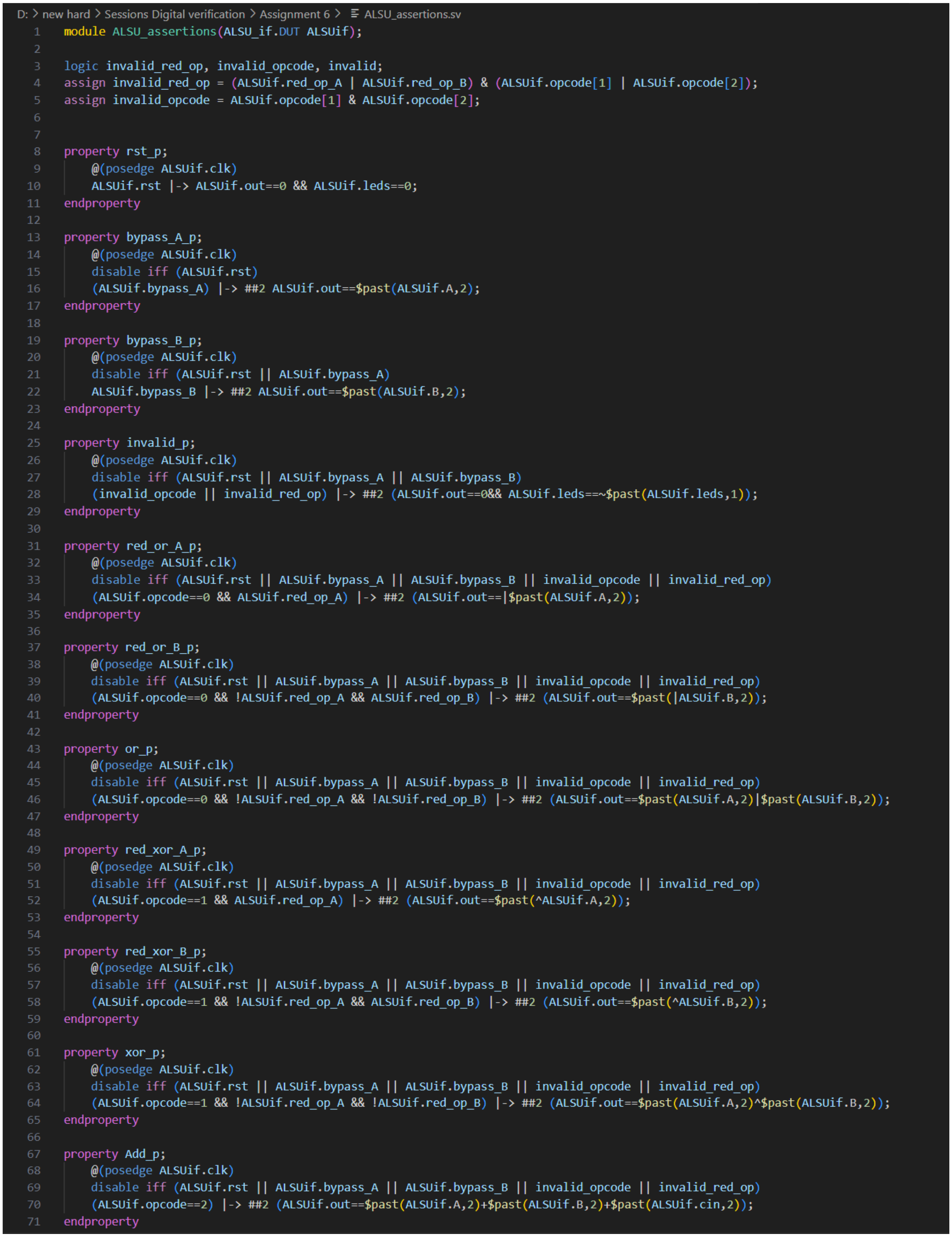
Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim



```
cross_direction:coverpoint item_cov.direction{
        bins crossdir ={0,1};
        option.weight=0;
        }

//Cross coverage
A_B_ADD_MUL: cross A_extremes, B_extremes ,ALU_cp
{
        bins Extreme_Add_mul = binsof (A_extremes) && binsof (B_extremes) && binsof (ALU_cp.Bins_arith);
}

CIN_ADD_cross: cross cross_cin, ALU_cp{

        bins CIN_ADD = binsof(ALU_cp) intersect {ADD} && binsof(cross_cin);
        option.cross_auto_bin_max = 0;
}
shifting_cross: cross ALU_cp,cross_serial_in
{
        bins serial_in_shift = binsof(ALU_cp) intersect {SHIFT} && binsof(cross_serial_in);
        option.cross_auto_bin_max = 0;
}
shifting_rotating_cross: cross ALU_cp,cross_direction
{
        bins serial_in_shift = binsof(ALU_cp.Bins_shift)  && binsof(cross_direction);
        option.cross_auto_bin_max = 0;
}
OR_XOR_redA_cross: cross ALU_cp, cross_redA, A_cp,B_cp
{
        bins OR_XOR_redA = binsof(ALU_cp.Bins_bitwise) && binsof(cross_redA) intersect{1} && binsof(A_cp.A_data_walkingones) && binsof(B_cp) intersect {ZERO};
        option.cross_auto_bin_max = 0;
}
OR_XOR_redB_cross: cross ALU_cp, cross_redB, A_cp,B_cp
{
        bins OR_XOR_redB = binsof(ALU_cp.Bins_bitwise) && binsof(cross_redB) intersect {1} && binsof(B_cp.B_data_walkingones)  && binsof(A_cp) intersect {ZERO};
        option.cross_auto_bin_max = 0;
}
INVALID_red_cross: cross ALU_cp,cross_redA, cross_redB
{
        bins invalid_red = !binsof(ALU_cp.Bins_bitwise)  && (binsof(cross_redB) intersect {1} || binsof(cross_redA) intersect{1});
        option.cross_auto_bin_max = 0;
}
endgroup

//Defining the new function and the covergroup inside it
function new(string name = "ALSU_coverage", uvm_component parent = null);
    super.new(name,parent);
    cov_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

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(item_cov);
        if(item_cov.rst||item_cov.bypass_A||item_cov.bypass_B)
            cov_gp.stop();
        else begin
            cov_gp.start();
            cov_gp.sample();
        end
    end
endtask
endclass
endpackage
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

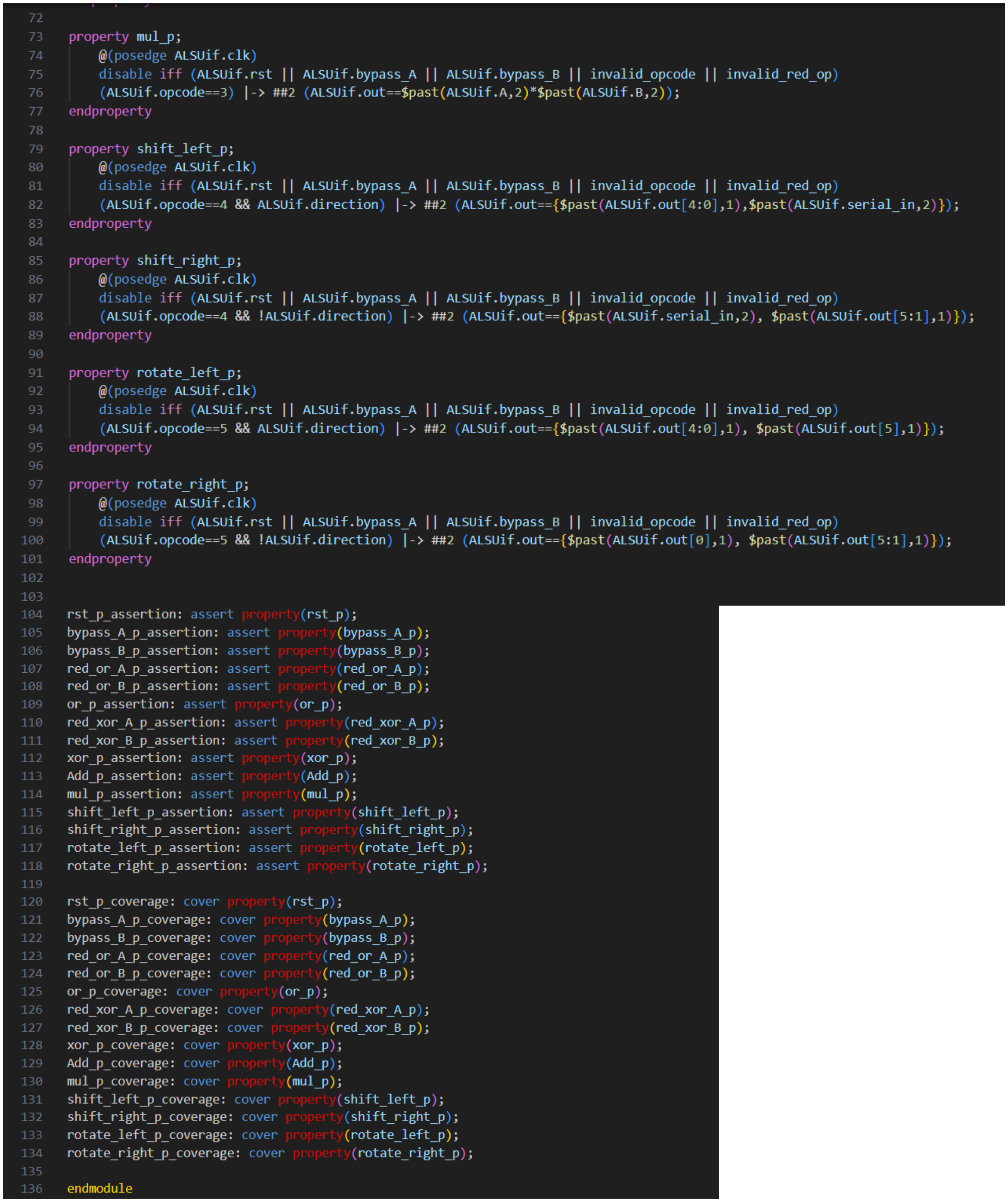
Assertions Code:



D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_assertions.sv

```
module ALSU_assertions(ALSU_if.DUT ALSUif);

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


property rst_p;
    @(posedge ALSUif.clk)
    ALSUif.rst |-> ALSUif.out==0 && ALSUif.leds==0;
endproperty

property bypass_A_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst)
    (ALSUif.bypass_A) |-> ##2 ALSUif.out==$past(ALSUif.A,2);
endproperty

property bypass_B_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A)
    ALSUif.bypass_B |-> ##2 ALSUif.out==$past(ALSUif.B,2);
endproperty

property invalid_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B)
    (invalid_opcode || invalid_red_op) |-> ##2 (ALSUif.out==0&& ALSUif.leds==~$past(ALSUif.leds,1));
endproperty

property red_or_A_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==0 && ALSUif.red_op_A) |-> ##2 (ALSUif.out==|$past(ALSUif.A,2));
endproperty

property red_or_B_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==0 && !ALSUif.red_op_A && ALSUif.red_op_B) |-> ##2 (ALSUif.out==$past(|ALSUif.B,2));
endproperty

property or_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==0 && !ALSUif.red_op_A && !ALSUif.red_op_B) |-> ##2 (ALSUif.out==$past(ALSUif.A,2)|$past(ALSUif.B,2));
endproperty

property red_xor_A_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==1 && ALSUif.red_op_A) |-> ##2 (ALSUif.out==$past(^ALSUif.A,2));
endproperty

property red_xor_B_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==1 && !ALSUif.red_op_A && ALSUif.red_op_B) |-> ##2 (ALSUif.out==$past(^ALSUif.B,2));
endproperty

property xor_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==1 && !ALSUif.red_op_A && !ALSUif.red_op_B) |-> ##2 (ALSUif.out==$past(ALSUif.A,2)^$past(ALSUif.B,2));
endproperty

property Add_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==2) |-> ##2 (ALSUif.out==$past(ALSUif.A,2)+$past(ALSUif.B,2)+$past(ALSUif.cin,2));
endproperty
```

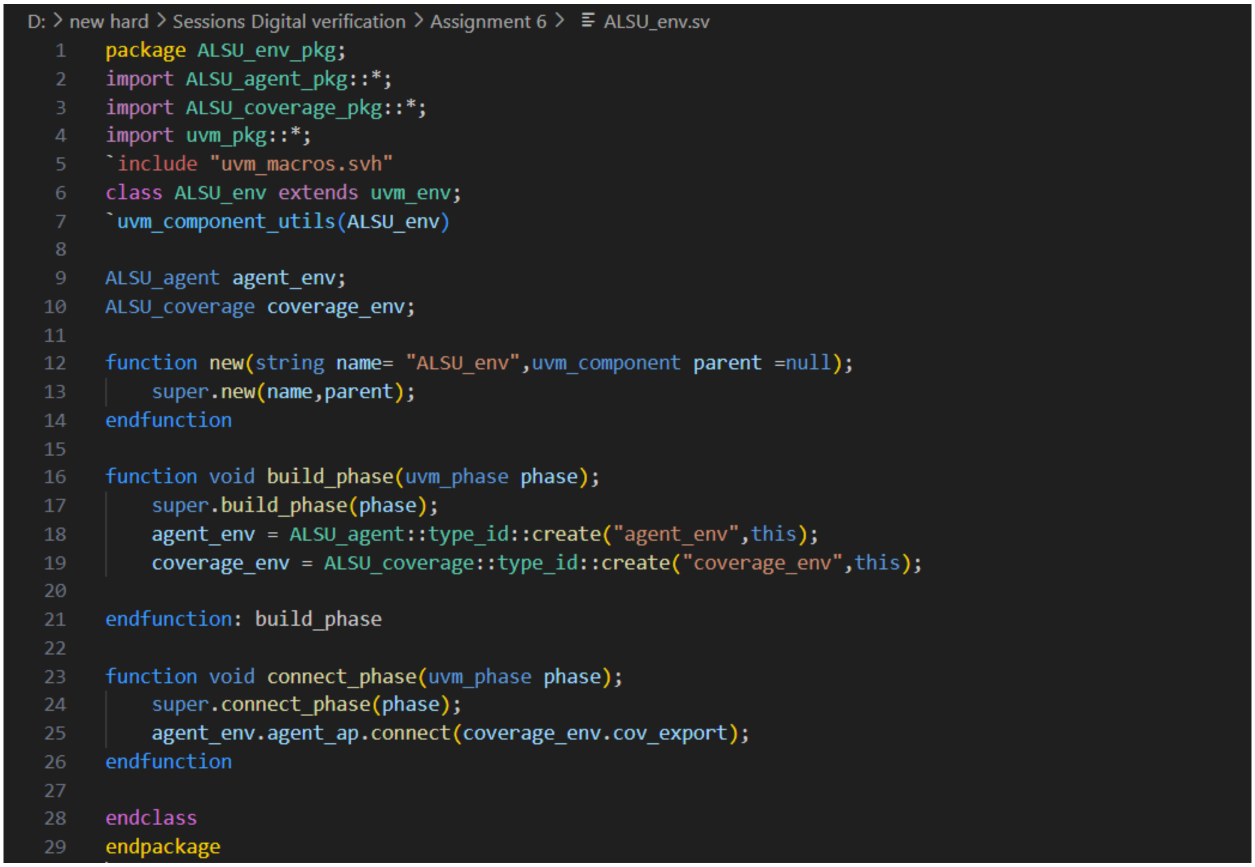
Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim



```
property mul_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==3) |-> ##2 (ALSUif.out==$past(ALSUif.A,2)*$past(ALSUif.B,2));
endproperty

property shift_left_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==4 && ALSUif.direction) |-> ##2 (ALSUif.out=={$past(ALSUif.out[4:0],1),$past(ALSUif.serial_in,2)});
endproperty

property shift_right_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==4 && !ALSUif.direction) |-> ##2 (ALSUif.out=={$past(ALSUif.serial_in,2), $past(ALSUif.out[5:1],1)});
endproperty

property rotate_left_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==5 && ALSUif.direction) |-> ##2 (ALSUif.out=={$past(ALSUif.out[4:0],1), $past(ALSUif.out[5],1)});
endproperty

property rotate_right_p;
    @(posedge ALSUif.clk)
    disable iff (ALSUif.rst || ALSUif.bypass_A || ALSUif.bypass_B || invalid_opcode || invalid_red_op)
    (ALSUif.opcode==5 && !ALSUif.direction) |-> ##2 (ALSUif.out=={$past(ALSUif.out[0],1), $past(ALSUif.out[5:1],1)});
endproperty


rst_p_assertion: assert property(rst_p);
bypass_A_p_assertion: assert property(bypass_A_p);
bypass_B_p_assertion: assert property(bypass_B_p);
red_or_A_p_assertion: assert property(red_or_A_p);
red_or_B_p_assertion: assert property(red_or_B_p);
or_p_assertion: assert property(or_p);
red_xor_A_p_assertion: assert property(red_xor_A_p);
red_xor_B_p_assertion: assert property(red_xor_B_p);
xor_p_assertion: assert property(xor_p);
Add_p_assertion: assert property(Add_p);
mul_p_assertion: assert property(mul_p);
shift_left_p_assertion: assert property(shift_left_p);
shift_right_p_assertion: assert property(shift_right_p);
rotate_left_p_assertion: assert property(rotate_left_p);
rotate_right_p_assertion: assert property(rotate_right_p);

rst_p_coverage: cover property(rst_p);
bypass_A_p_coverage: cover property(bypass_A_p);
bypass_B_p_coverage: cover property(bypass_B_p);
red_or_A_p_coverage: cover property(red_or_A_p);
red_or_B_p_coverage: cover property(red_or_B_p);
or_p_coverage: cover property(or_p);
red_xor_A_p_coverage: cover property(red_xor_A_p);
red_xor_B_p_coverage: cover property(red_xor_B_p);
xor_p_coverage: cover property(xor_p);
Add_p_coverage: cover property(Add_p);
mul_p_coverage: cover property(mul_p);
shift_left_p_coverage: cover property(shift_left_p);
shift_right_p_coverage: cover property(shift_right_p);
rotate_left_p_coverage: cover property(rotate_left_p);
rotate_right_p_coverage: cover property(rotate_right_p);

endmodule
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

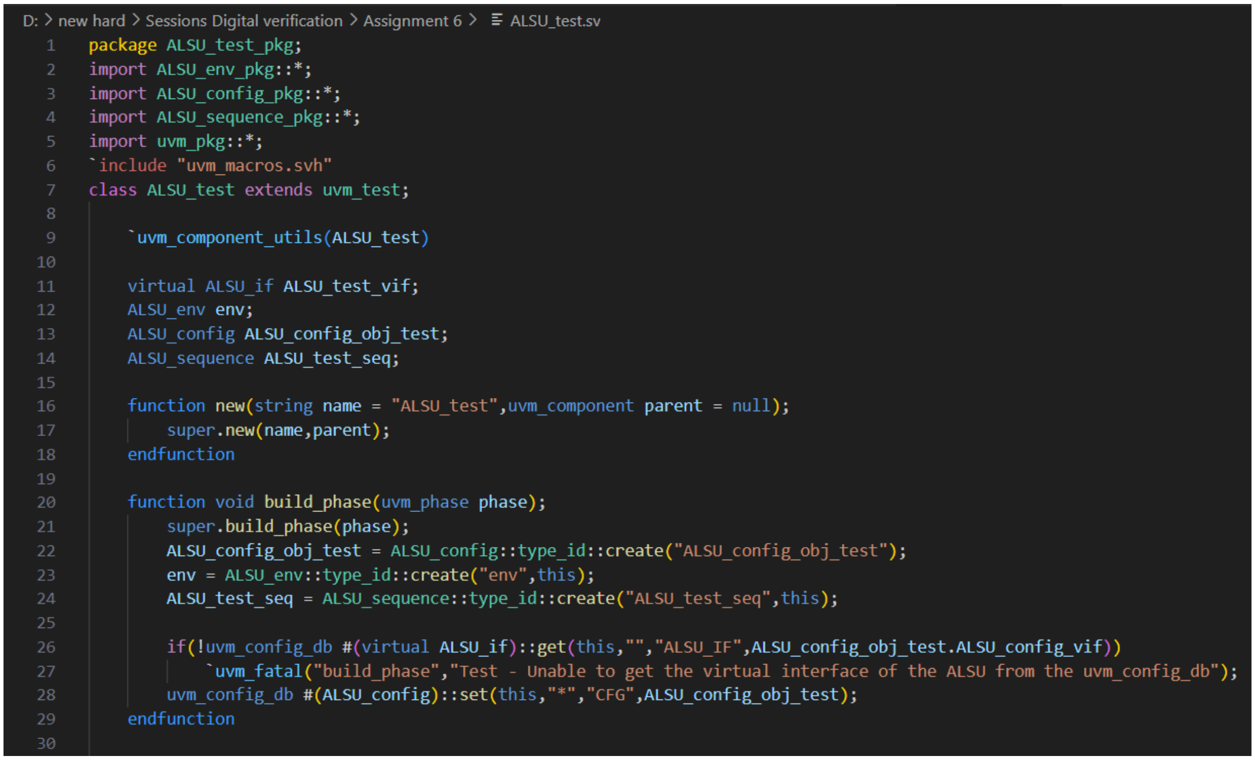
Environment Package Code:



D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_env.sv

```
package ALSU_env_pkg;
import ALSU_agent_pkg::*;
import ALSU_coverage_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ALSU_env extends uvm_env;
`uvm_component_utils(ALSU_env)

ALSU_agent agent_env;
ALSU_coverage coverage_env;

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);
    agent_env = ALSU_agent::type_id::create("agent_env",this);
    coverage_env = ALSU_coverage::type_id::create("coverage_env",this);

endfunction: build_phase

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    agent_env.agent_ap.connect(coverage_env.cov_export);
endfunction

endclass
endpackage
```

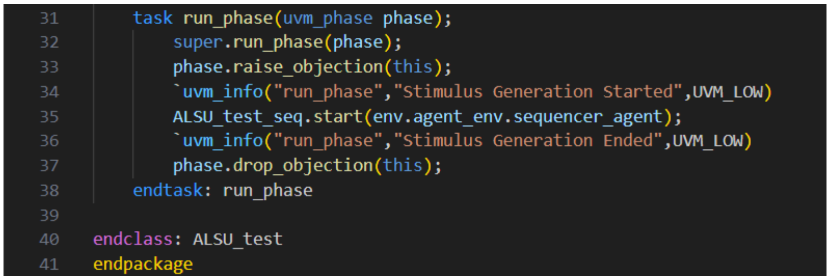
Test Package Code:



D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_test.sv

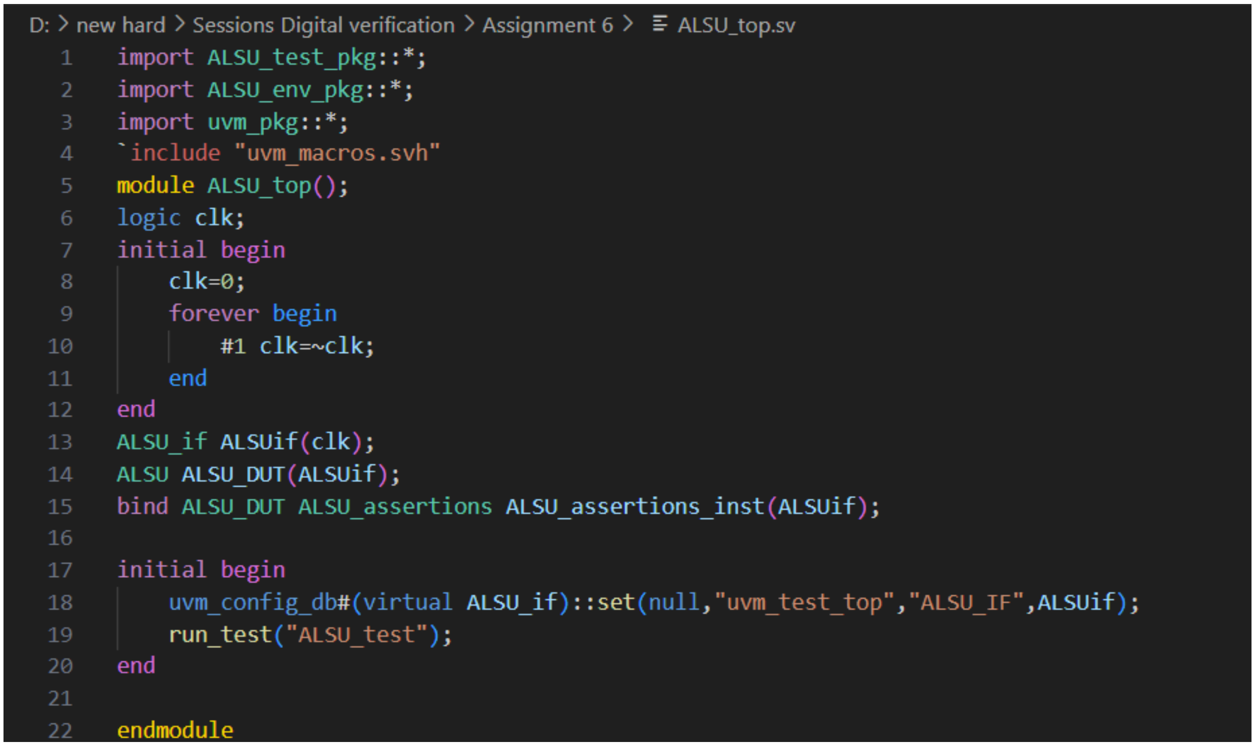
```
package ALSU_test_pkg;
import ALSU_env_pkg::*;
import ALSU_config_pkg::*;
import ALSU_sequence_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
class ALSU_test extends uvm_test;

    `uvm_component_utils(ALSU_test)

    virtual ALSU_if ALSU_test_vif;
    ALSU_env env;
    ALSU_config ALSU_config_obj_test;
    ALSU_sequence ALSU_test_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);
        ALSU_config_obj_test = ALSU_config::type_id::create("ALSU_config_obj_test");
        env = ALSU_env::type_id::create("env",this);
        ALSU_test_seq = ALSU_sequence::type_id::create("ALSU_test_seq",this);

        if(!uvm_config_db #(virtual ALSU_if)::get(this,"","ALSU_IF",ALSU_config_obj_test.ALSU_config_vif))
            `uvm_fatal("build_phase","Test - Unable to get the virtual interface of the ALSU from the uvm_config_db");
        uvm_config_db #(ALSU_config)::set(this,"*","CFG",ALSU_config_obj_test);
    endfunction

```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim



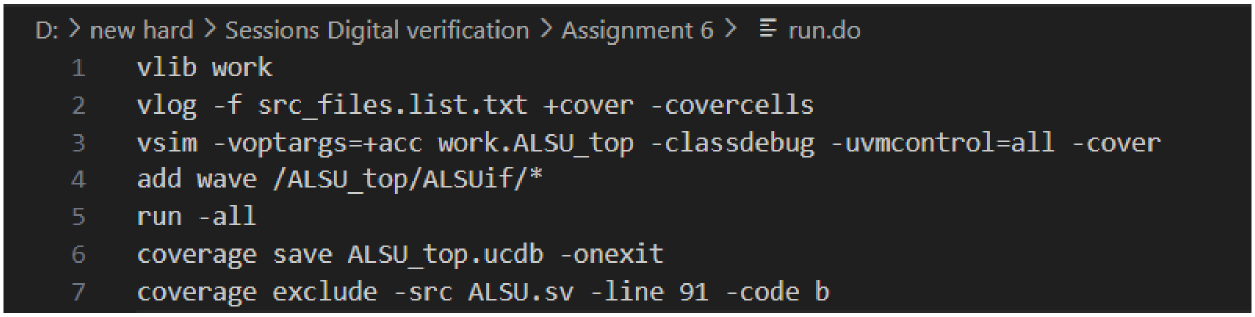
```
    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        `uvm_info("run_phase","Stimulus Generation Started",UVM_LOW)
        ALSU_test_seq.start(env.agent_env.sequencer_agent);
        `uvm_info("run_phase","Stimulus Generation Ended",UVM_LOW)
        phase.drop_objection(this);
    endtask: run_phase

endclass: ALSU_test
endpackage
```

Top Code:


D: > new hard > Sessions Digital verification > Assignment 6 > ALSU_top.sv

```
import ALSU_test_pkg::*;
import ALSU_env_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"
module ALSU_top();
logic clk;
initial begin
    clk=0;
    forever begin
        #1 clk=~clk;
    end
end
ALSU_if ALSUif(clk);
ALSU ALSU_DUT(ALSUif);
bind ALSU_DUT ALSU_assertions ALSU_assertions_inst(ALSUif);

initial begin
    uvm_config_db#(virtual ALSU_if)::set(null,"uvm_test_top","ALSU_IF",ALSUif);
    run_test("ALSU_test");
end

endmodule
```

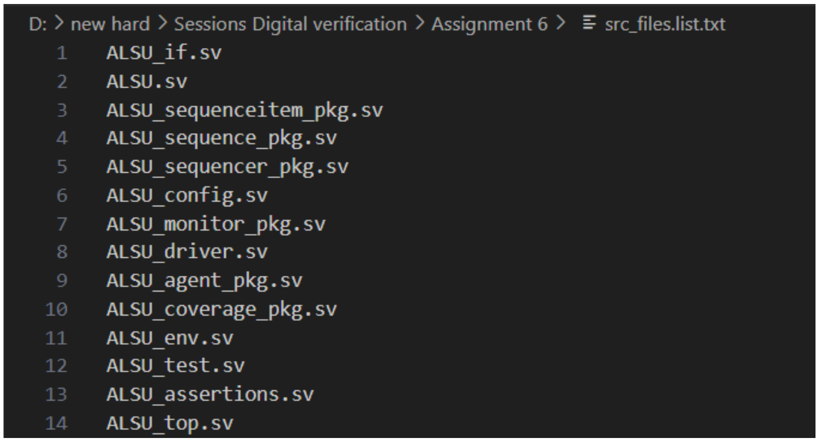
Do file:


D: > new hard > Sessions Digital verification > Assignment 6 > run.do

```
vlib work
vlog -f src_files.list.txt +cover -covercells
vsim -voptargs=+acc work.ALSU_top -classdebug -uvmcontrol=all -cover
add wave /ALSU_top/ALSUif/*
run -all
coverage save ALSU_top.ucdb -onexit
coverage exclude -src ALSU.sv -line 91 -code b
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

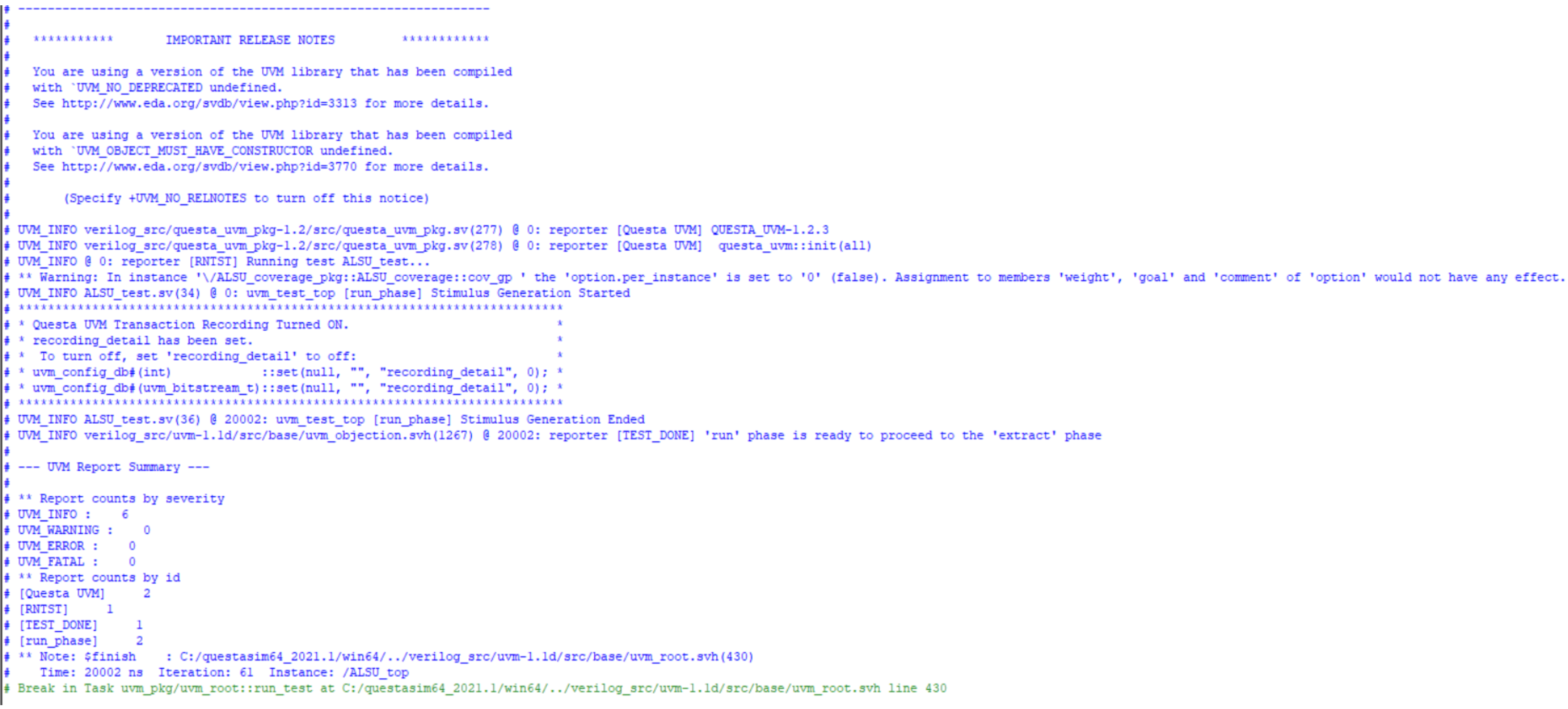
Src-files list:



D: > new hard > Sessions Digital verification > Assignment 6 > src_files.list.txt

```
ALSU_if.sv
ALSU.sv
ALSU_sequenceitem_pkg.sv
ALSU_sequence_pkg.sv
ALSU_sequencer_pkg.sv
ALSU_config.sv
ALSU_monitor_pkg.sv
ALSU_driver.sv
ALSU_agent_pkg.sv
ALSU_coverage_pkg.sv
ALSU_env.sv
ALSU_test.sv
ALSU_assertions.sv
ALSU_top.sv
```

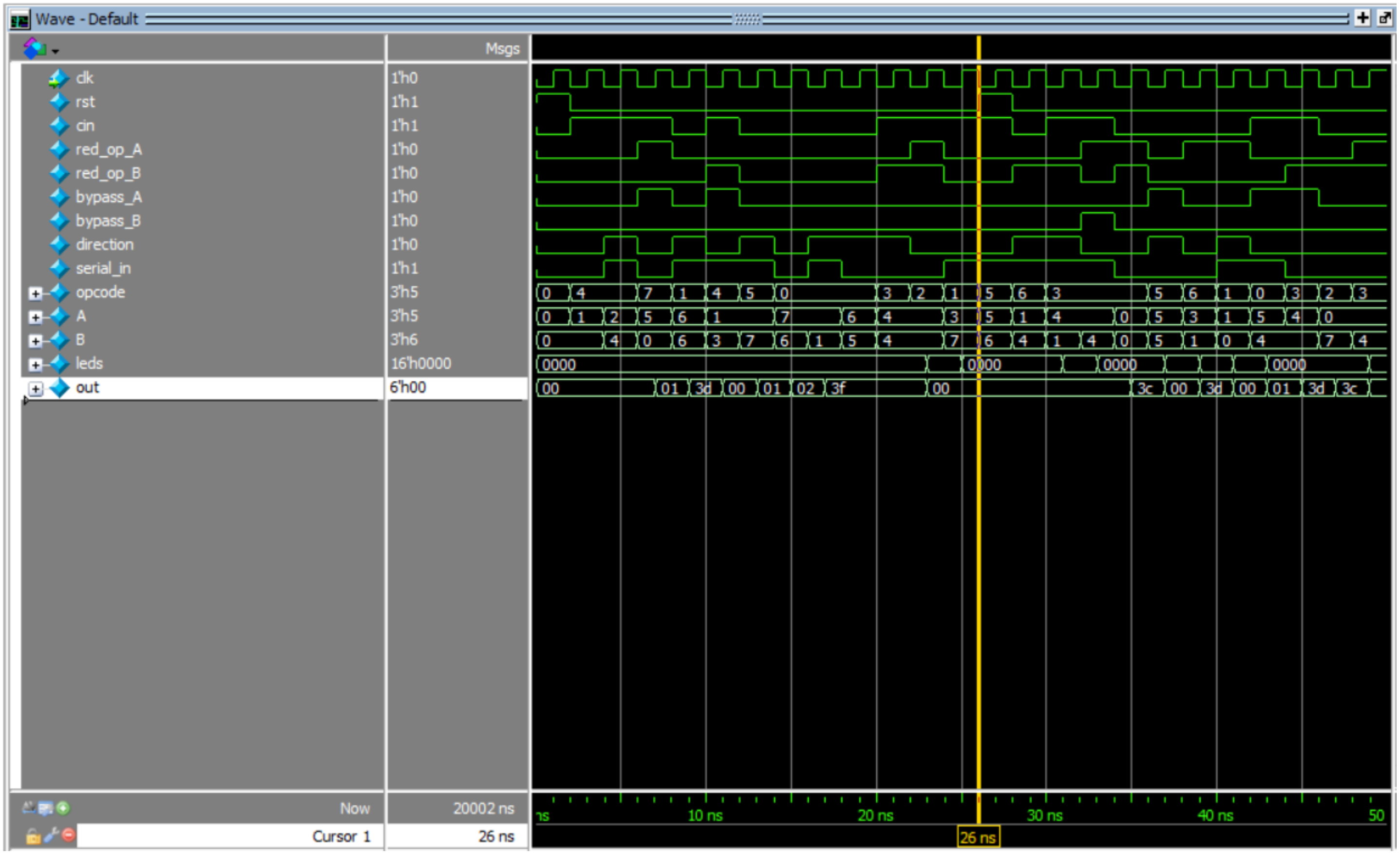
Transcript:

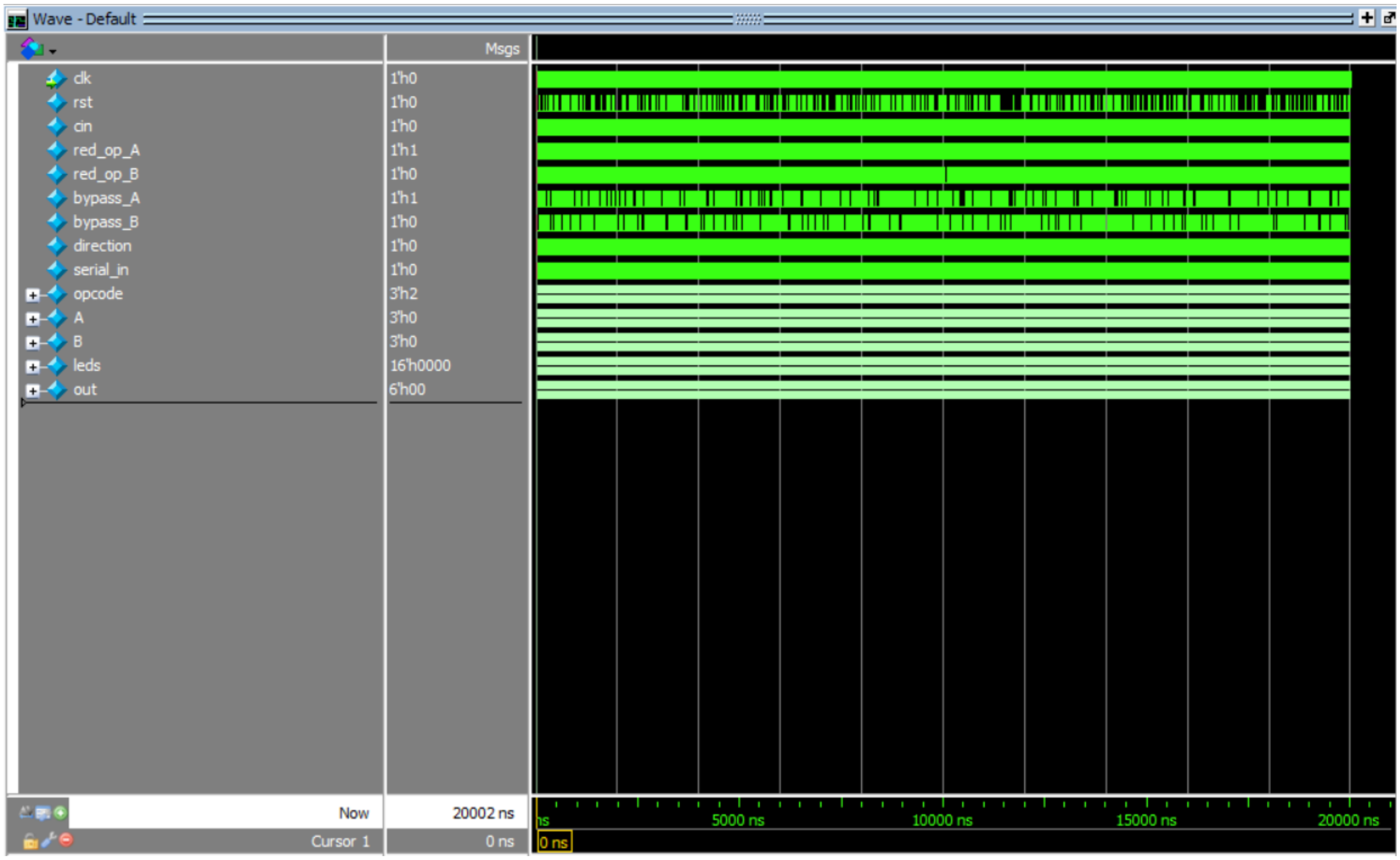


```
# ----------------------------------------------------------------
#
#   ***********       IMPORTANT RELEASE NOTES         ************
#
#   You are using a version of the UVM library that has been compiled
#   with `UVM_NO_DEPRECATED undefined.
#   See http://www.eda.org/svdb/view.php?id=3313 for more details.
#
#   You are using a version of the UVM library that has been compiled
#   with `UVM_OBJECT_MUST_HAVE_CONSTRUCTOR undefined.
#   See http://www.eda.org/svdb/view.php?id=3770 for more details.
#
#       (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test ALSU_test...
# ** Warning: In instance '\/ALSU_coverage_pkg::ALSU_coverage::cov_gp ' the 'option.per_instance' is set to '0' (false). Assignment to members 'weight', 'goal' and 'comment' of 'option' would not have any effect.
# UVM_INFO ALSU_test.sv(34) @ 0: uvm_test_top [run_phase] Stimulus Generation Started
# ***************************************************************************
# * Questa UVM Transaction Recording Turned ON.                             *
# * recording_detail has been set.                                          *
# *  To turn off, set 'recording_detail' to off:                            *
# * uvm_config_db#(int)            ::set(null, "", "recording_detail", 0); *
# * uvm_config_db#(uvm_bitstream_t)::set(null, "", "recording_detail", 0); *
# ***************************************************************************
# UVM_INFO ALSU_test.sv(36) @ 20002: uvm_test_top [run_phase] Stimulus Generation Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 20002: 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: 20002 ns  Iteration: 61  Instance: /ALSU_top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

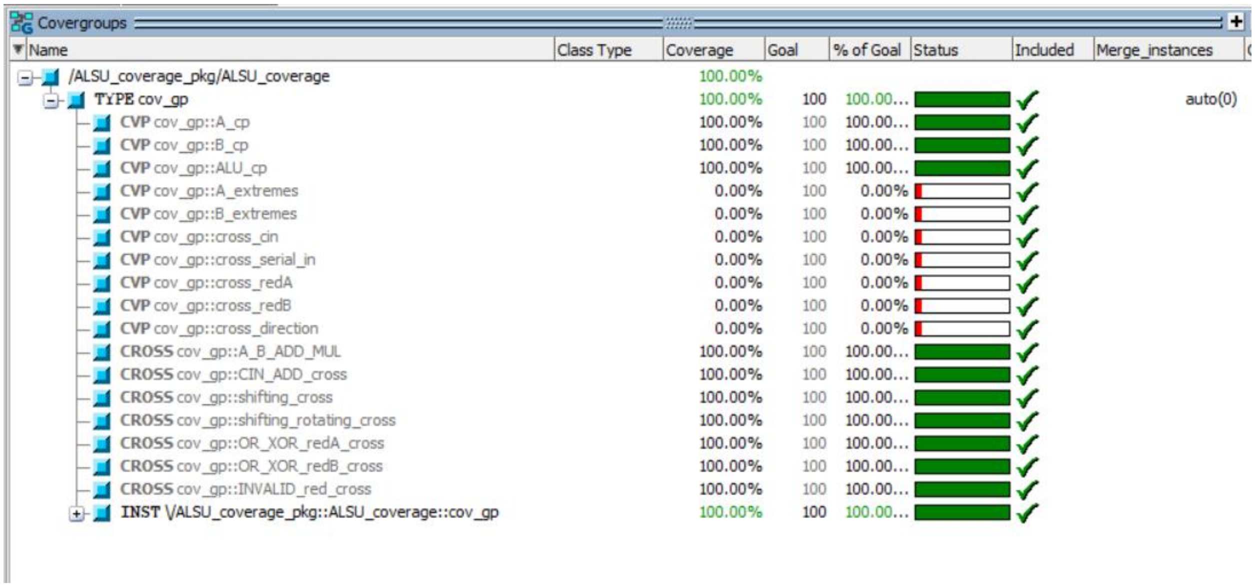
Test cases:





Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

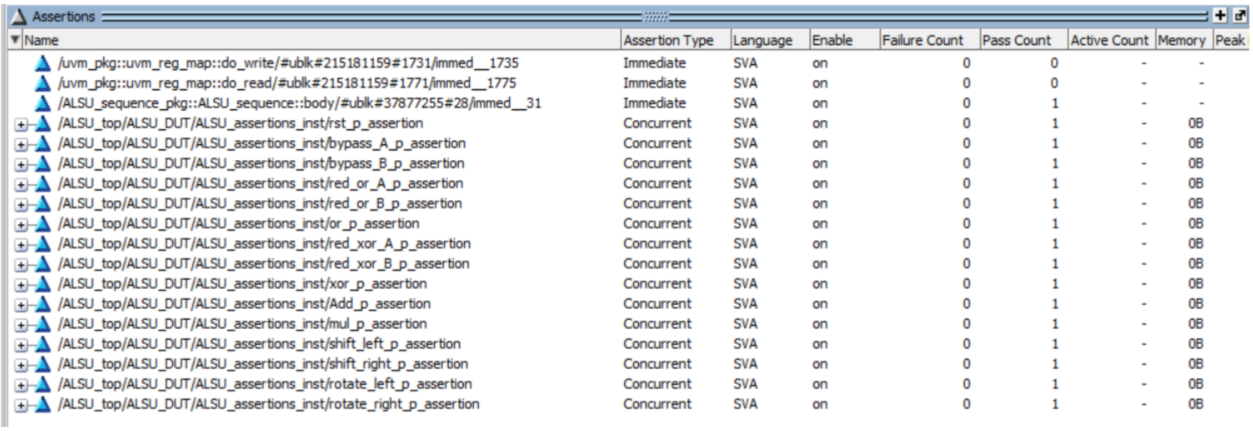
Functional Coverage:



Covergroups

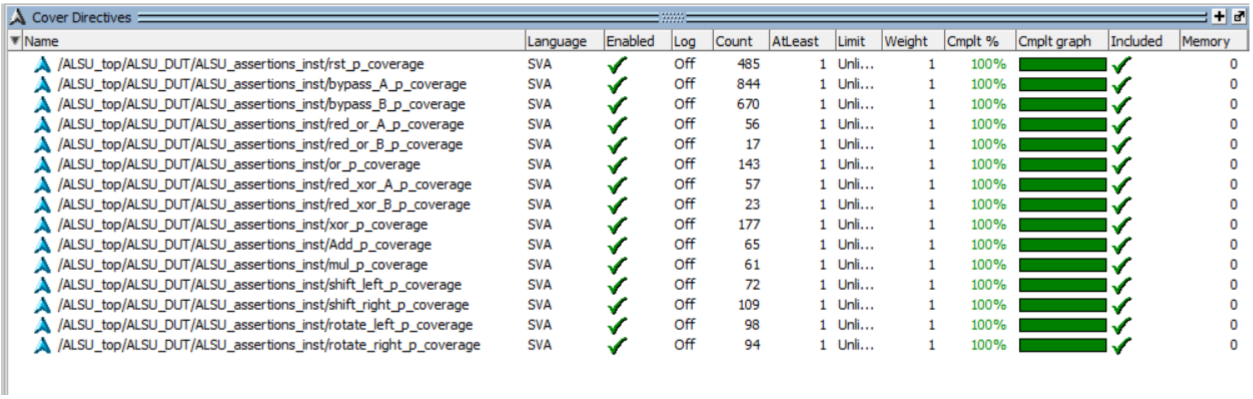
Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances
/ALSU_coverage_pkg/ALSU_coverage		100.00%					
TYPE cov_gp		100.00%	100	100.00...	🟩	✓	auto(0)
CVP cov_gp::A_cp		100.00%	100	100.00...	🟩	✓	
CVP cov_gp::B_cp		100.00%	100	100.00...	🟩	✓	
CVP cov_gp::ALU_cp		100.00%	100	100.00...	🟩	✓	
CVP cov_gp::A_extremes		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::B_extremes		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::cross_cin		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::cross_serial_in		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::cross_redA		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::cross_redB		0.00%	100	0.00%	🟥	✓	
CVP cov_gp::cross_direction		0.00%	100	0.00%	🟥	✓	
CROSS cov_gp::A_B_ADD_MUL		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::CIN_ADD_cross		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::shifting_cross		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::shifting_rotating_cross		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::OR_XOR_redA_cross		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::OR_XOR_redB_cross		100.00%	100	100.00...	🟩	✓	
CROSS cov_gp::INVALID_red_cross		100.00%	100	100.00...	🟩	✓	
INST \/ALSU_coverage_pkg::ALSU_coverage::cov_gp		100.00%	100	100.00...	🟩	✓	

Assertion Coverage & directive coverage:



Assertions

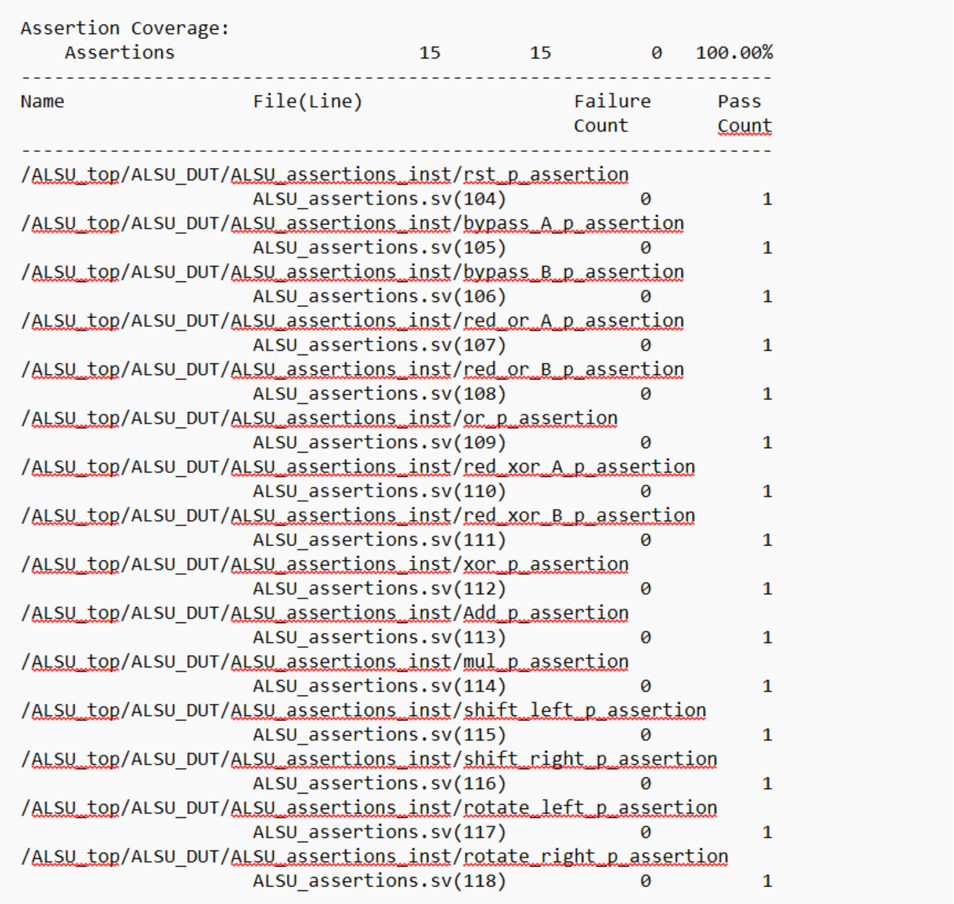
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-	-	
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-	-	
/ALSU_sequence_pkg::ALSU_sequence::body/#ublk#37877255#28/immed__31	Immediate	SVA	on	0	1	-	-	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rst_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_A_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_B_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_A_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_B_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/or_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_A_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_B_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/xor_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/Add_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/mul_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_left_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_right_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_left_p_assertion	Concurrent	SVA	on	0	1	-	0B	
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_right_p_assertion	Concurrent	SVA	on	0	1	-	0B	



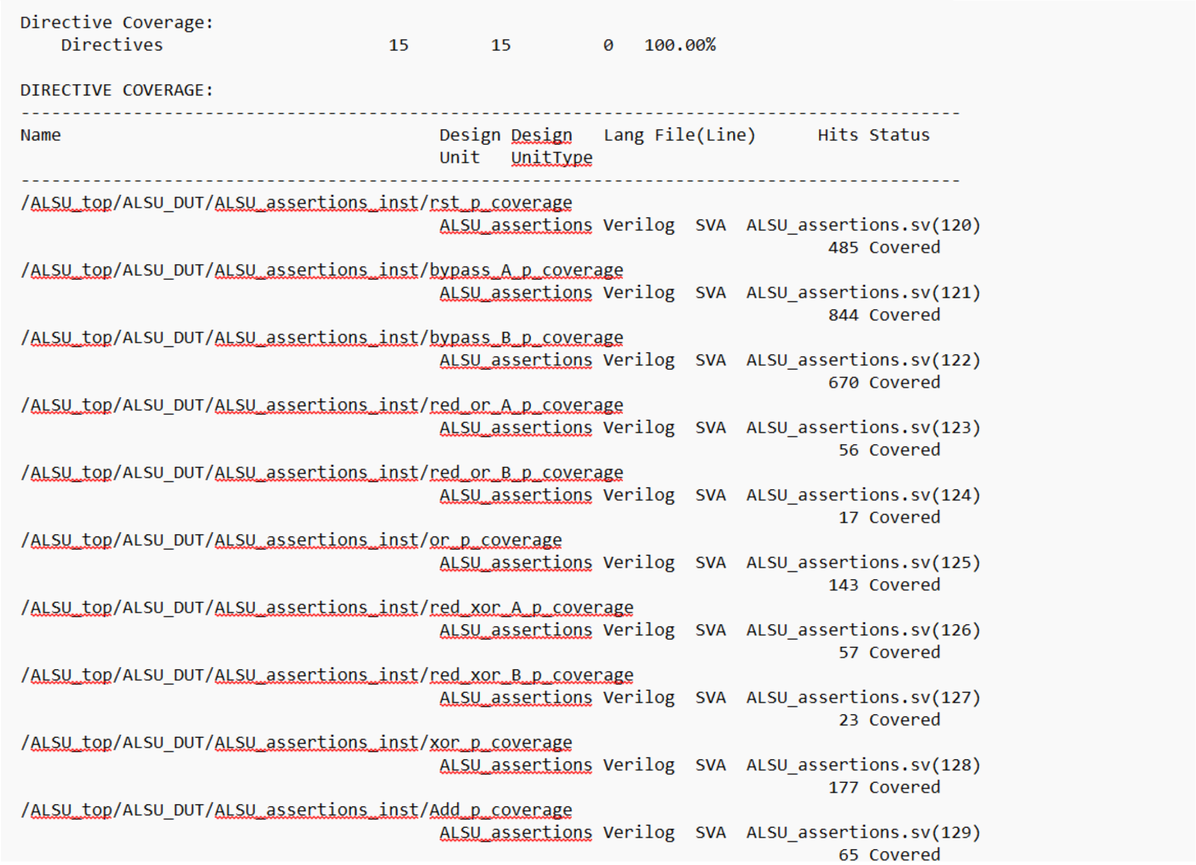
Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rst_p_coverage	SVA	✓	Off	485	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_A_p_coverage	SVA	✓	Off	844	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_B_p_coverage	SVA	✓	Off	670	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_A_p_coverage	SVA	✓	Off	56	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_B_p_coverage	SVA	✓	Off	17	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/or_p_coverage	SVA	✓	Off	143	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_A_p_coverage	SVA	✓	Off	57	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_B_p_coverage	SVA	✓	Off	23	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/xor_p_coverage	SVA	✓	Off	177	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/Add_p_coverage	SVA	✓	Off	65	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/mul_p_coverage	SVA	✓	Off	61	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_left_p_coverage	SVA	✓	Off	72	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_right_p_coverage	SVA	✓	Off	109	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_left_p_coverage	SVA	✓	Off	98	1	Unli...	1	100%	🟩	✓	0
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_right_p_coverage	SVA	✓	Off	94	1	Unli...	1	100%	🟩	✓	0

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim

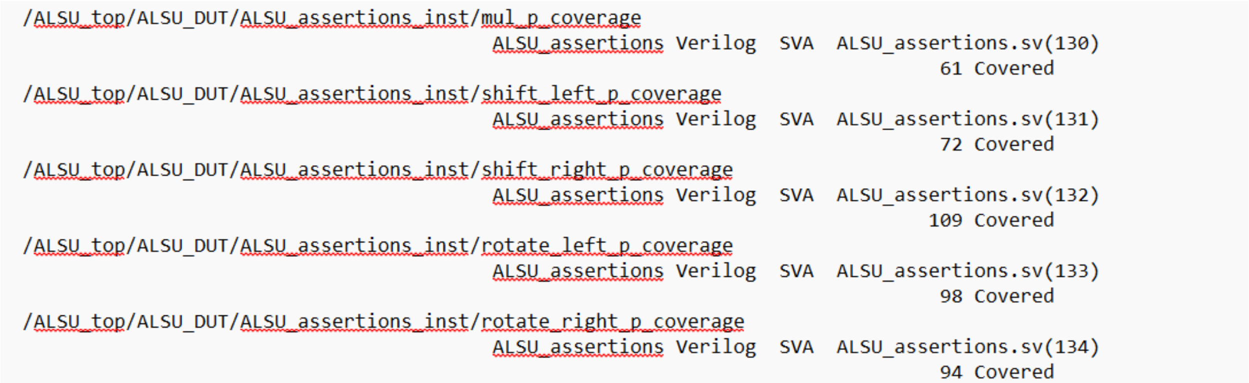


```
Assertion Coverage:
    Assertions                      15        15         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                    Failure      Pass
                                                   Count        Count
--------------------------------------------------------------------
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rst_p_assertion
                     ALSU_assertions.sv(104)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_A_p_assertion
                     ALSU_assertions.sv(105)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_B_p_assertion
                     ALSU_assertions.sv(106)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_A_p_assertion
                     ALSU_assertions.sv(107)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_B_p_assertion
                     ALSU_assertions.sv(108)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/or_p_assertion
                     ALSU_assertions.sv(109)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_A_p_assertion
                     ALSU_assertions.sv(110)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_B_p_assertion
                     ALSU_assertions.sv(111)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/xor_p_assertion
                     ALSU_assertions.sv(112)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/Add_p_assertion
                     ALSU_assertions.sv(113)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/mul_p_assertion
                     ALSU_assertions.sv(114)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_left_p_assertion
                     ALSU_assertions.sv(115)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_right_p_assertion
                     ALSU_assertions.sv(116)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_left_p_assertion
                     ALSU_assertions.sv(117)             0          1
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_right_p_assertion
                     ALSU_assertions.sv(118)             0          1
```



```
Directive Coverage:
    Directives                     15         15          0   100.00%

DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                                         Design Design    Lang File(Line)      Hits Status
                                             Unit   UnitType
--------------------------------------------------------------------------------------------
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/rst_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(120)
                                                                                    485 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_A_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(121)
                                                                                    844 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/bypass_B_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(122)
                                                                                    670 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_A_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(123)
                                                                                     56 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_or_B_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(124)
                                                                                     17 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/or_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(125)
                                                                                    143 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_A_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(126)
                                                                                     57 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/red_xor_B_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(127)
                                                                                     23 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/xor_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(128)
                                                                                    177 Covered
/ALSU_top/ALSU_DUT/ALSU_assertions_inst/Add_p_coverage
                                             ALSU_assertions Verilog  SVA  ALSU_assertions.sv(129)
                                                                                     65 Covered
```

Assignment 6 ---> Abdelrahman Khaled Fouad Abdelrahim



```
 /ALSU_top/ALSU_DUT/ALSU_assertions_inst/mul_p_coverage
                                         ALSU_assertions Verilog  SVA  ALSU_assertions.sv(130)
                                                                                 61 Covered
 /ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_left_p_coverage
                                         ALSU_assertions Verilog  SVA  ALSU_assertions.sv(131)
                                                                                 72 Covered
 /ALSU_top/ALSU_DUT/ALSU_assertions_inst/shift_right_p_coverage
                                         ALSU_assertions Verilog  SVA  ALSU_assertions.sv(132)
                                                                                109 Covered
 /ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_left_p_coverage
                                         ALSU_assertions Verilog  SVA  ALSU_assertions.sv(133)
                                                                                 98 Covered
 /ALSU_top/ALSU_DUT/ALSU_assertions_inst/rotate_right_p_coverage
                                         ALSU_assertions Verilog  SVA  ALSU_assertions.sv(134)
                                                                                 94 Covered
```